Lecture Notes in Computer Science 2669
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Vipin Kumar Marina L. Gavrilova
Chih Jeng Kenneth Tan Pierre L'Ecuyer (Eds.)

Computational Science and Its Applications – ICCSA 2003

International Conference
Montreal, Canada, May 18-21, 2003
Proceedings, Part III



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Vipin Kumar
Army High Performance Computing Research Center, USA
University of Minessota
Department of Computer Science and Engineering, MN 55455, USA
E-mail: kumar@cs.umn.edu

Marina L. Gavrilova
University of Calgary
Department of Computer Science, Calgary, AB, T2N1N4, Canada
E-mail: marina@cpsc.ucalgary.ca

Chih Jeng Kenneth Tan
Heuchera Technologies Inc.
122 9251-8 Yonge Street, Richmond Hill, ON, Canada L4C 9T3
The Queen's University of Belfast
School of Computer Science, Belfast BT7 1NN, Northern Ireland, UK
E-mail: cjtan@OptimaNumerics.com

Pierre L'Ecuyer
Université de Montréal
Département d'informatique et de recherche opérationelle
Montréal, Québec, H3C 3J7, Canada
E-mail: lecuyer@iro.umontreal.ca

Cataloging-in-Publication Data applied for
A catalog record for this book is available from the Library of Congress.
Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): D, F, G, H, I, J, C.2-3
ISSN 0302-9743
ISBN 3-540-40156-3 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2003
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10927441 06/3142 5 4 3 2 1 0

Preface

Computational Science is becoming a vital part of many scientific investigations, impacting researchers and practitioners in areas ranging from aerospace and automotive to chemistry, electronics, geosciences, finance, mathematics, and physics. Due to the sheer size of many challenges in computational science, the use of supercomputing, parallel processing, and sophisticated algorithms is inevitable.

This volume contains the proceedings of The 2003 International Conference on Computational Science and Its Applications (ICCSA 2003), held in Montreal, Canada in May 2003. The papers presented here reflect the aim of the program committee to bring together researchers and scientists from mathematics and computer science, the basic computing disciplines, and researchers from various application areas who are pioneering advanced application of computational methods to sciences such as physics, chemistry, life sciences, engineering, arts, and the humanities, along with software developers and vendors, to discuss problems and solutions in the area, identify new issues, and shape future directions for research, as well as help industrial users apply various advanced computational techniques.

Our outstanding invited speakers outlined the theme for the Conference, considering sophisticated numerical computational methods, emerging computational solutions, and problems and applications pushing the bounds of advanced computing techniques and hardware.

This Conference was held as the first in its series, with an emphasis on Computational Science and the application of computational science. Interest from researchers and practitioners in this interdisciplinary area has been overwhelming, and we are looking forward to future events in this Conference series.

The ICCSA 2003 Conference was jointly organized by (in alphabetical order): Heuchera Technologies, Canada, The Queen's University of Belfast, UK, SHARCNET, Canada, University of Calgary, Canada, University of Minnesota, USA, and University of Montreal, Canada.

ICCSA 2003 would not have been possible without the support of our sponsors (in alphabetical order):

- CERCA, Canada
- IBM Canada, Canada
- IBM, USA
- Heuchera Technologies, Canada
- Pallas, Germany
- The Queen's University of Belfast, UK
- SHARCNET, Canada
- Society for Industrial and Applied Mathematics, USA
- Springer-Verlag, Germany

– School of Computer Science, The Queen's University of Belfast, UK
– Department of Computer Science, University of Calgary, Canada

The support provided by these sponsors was beyond our expectations.

We would also like to mention the following individuals who have greatly supported us (in alphabetical order):

– Hamid Arabnia (University of Georgia, USA)
– Michael Bauer (SHARCNET, Canada)
– Matthew Dixon (Heuchera Technologies, Canada)
– Anna Kramer (Springer-Verlag, Germany)
– Dominic Lam (IBM Canada, Canada)
– Alfred Hofmann (Springer-Verlag, Germany)
– Andres Iglesias (University de Cantabria, Spain)
– Allan MacIsaac (SHARCNET, Canada)
– Youngsong Mun (Soongsil University, Korea)
– Colette Tipping (The Queen's University of Belfast, UK)
– Birgit Wissen (Pallas, Germany)

We are deeply indebted to the members of the program committee and all people in the community who have helped us to form a successful program. Many individuals have kindly offered us their assistance and support in the organization of this Conference. If we have failed to thank them, we would like to ask them to accept our sincere apologies.

May 2003

Vipin Kumar
Marina L. Gavrilova
C.J. Kenneth Tan
Pierre L'Ecuyer

Conference Organization

Conference Chairs

Honorary Chair:
Vipin Kumar (Army High Performance Computing Center, USA, and University of Minnesota, USA)

Conference Chairs:
Marina Gavrilova (University of Calgary, Canada)
C.J. Kenneth Tan (Heuchera Technologies, Canada, and The Queen's University of Belfast, UK)

Local Organizing Chair:
Pierre L'Ecuyer (University of Montreal, Canada)

International Steering Committee

Michael Bauer (SHARCNET, Canada)
J.A. Rod Blais (University of Calgary, Canada)
Alexander V. Bogdanov (Institute for High Performance Computing and Data Bases, Russia)
Marina L. Gavrilova (University of Calgary, Canada)
Andres Iglesias (University de Cantabria, Spain)
Benjoe A. Juliano (California State University at Chico, USA)
Vipin Kumar (Army High Performance Computing Center, USA, and University of Minnesota, USA)
Reneé S. Renner (California State University at Chico, USA)
C.J. Kenneth Tan (Heuchera Technologies, Canada, and The Queen's University of Belfast, UK)
Jerzy Wasniewski (Danish Computing Center for Research and Education, Denmark)

Program Committee

Sergei Bespamyatnikh (Duke University, USA)
J.A. Rod Blais (University of Calgary, Canada)
Alexander V. Bogdanov (Institute for High Performance Computing and Data Bases, Russia)
Marian Bubak (AGH, Poland)
Toni Cortes (Universidad de Catalunya, Barcelona, Spain)
Brian J. d'Auriol (University of Texas at El Paso, USA)

Ovidiu Daescu (University of Texas at Dallas, USA)
Frank Dehne (Carleton University, Canada)
Tamal Dey (Ohio State University, USA)
Ivan Dimov (Bulgarian Academy of Sciences, Bulgaria)
Matthew F. Dixon (Heuchera Technologies, Canada)
Geoffrey Fox (Indiana University, USA)
Marina L. Gavrilova (University of Calgary, Canada)
Christopher Gold (Hong Kong Polytechnic University)
Bob Hertzberger (Universiteit van Amsterdam, The Netherlands)
Andres Iglesias (University de Cantabria, Spain)
Chris Johnson (University of Utah, USA)
Benjoe A. Juliano (California State University at Chico, USA)
Deok-Soo Kim (Hanyang University, Korea)
Vipin Kumar (Army High Performance Computing Center, USA, and University of Minnesota, USA)
Antonio Lagana (Università degli Studi di Perugia, Italy)
Michael Mascagni (Florida State University, USA)
Cathy McDonald (Department of Defense HPC Modernization Program, USA)
Graham Megson (University of Reading, UK)
Jiri Nedoma (Academy of Sciences of the Czech Republic, Czech Republic)
Robert Panoff (Shodor Education Foundation, USA)
Ron Perrott (The Queen's University of Belfast, UK)
Reneé S. Renner (California State University at Chico, USA)
Vaclav Skala (University of West Bohemia, Czech Republic)
Kokichi Sugihara (University of Tokyo, Japan)
David Taniar (Monash University, Australia)
Ruppa K. Thulasiram (University of Manitoba, Canada)
Koichi Wada (University of Tsukuba, Japan)
Jerzy Wasniewski (Danish Computing Center for Research and Education, Denmark)
Roy Williams (California Institute of Technology, USA)
Stephen Wismath (University of Lethbridge, Canada)
Osman Yasar (SUNY at Brockport, USA)
Zahari Zlatev (Danish Environmental Research Institute, Denmark)

Table of Contents, Part III

Computational and Methodological Developments in Spatial Analysis within GIS

A Variable Resolution Approach to Cluster Discovery in Spatial Data Mining 1
A.J. Brimicombe

PGIS, a Software for Paleogeographic Reconstruction in ArcGIS 12
S. Mei

Spatial and Temporal Autocorrelation in Innovation Diffusion Analysis ... 23
S. Bertazzon

Testing Differences between Case and Control Point Patterns Using Nearest Neighbour Distances and Bootstrapping 33
K.A. Henry, L.M. Burge, D. Nguyen

Assessment of Different Link Functions for Modeling Binary Data to Derive Sound Inferences and Predictions 43
F. Huettmann, J. Linke

Partitioning Mesh-Based Applications for Computational Grids

Improvements to the Helpful-Set Algorithm and a New Evaluation Scheme for Graph-Partitioners 49
S. Schamberger

Mesh Partitioners for Computational Grids: A Comparison 60
S. Huang, E. Aubanel, V.C. Bhavsar

CONFIIT: A Middleware for Peer to Peer Computing 69
O. Flauzac, M. Krajecki, J. Fugère

Internet Communications Security

On MARS's s-boxes Strength against Linear Cryptanalysis 79
C.J. Hernández Castro, L.J. García Villalba, J.C. Hernández Castro, J.M. Sierra Cámara

A Secret Sharing Scheme Using Matrices 84
C. Hernández-Goya, P. Caballero-Gil, C. Bruno-Castañeda

A Modular Architecture for Distributed IDS in MANET 91
R.S. Puttini, J.-M. Percher, L. Mé, O. Camp, R. de Sousa Jr., C.J. Barenco Abbas, L.J. García-Villalba

Characterization of the Security Implications of Alternative E-procurement Topologies 114
E. Ponce, A. Durán

Protection of Multiagent Systems 123
J.M. Sierra, J.C. Hernández, E. Ponce, A. Ribagorda

On the Number of Equivalence Classes in Certain Stream Ciphers 129
L.J. García-Villalba

Computational Finance

Parallelization and Vectorization of Simulation Based Option Pricing Methods 139
J. Schumacher, U. Jaekel, A. Basermann

An Algorithm for Determining the Controllers of Supervised Entities at the First and Second Levels: A Case Study with the Brazilian Central Bank 148
V.G. Fracari Branco, L. Weigang, M.P. Estrela Abad, J. Denzinger

Two Factor Option Pricing with Uncertain Volatility 158
D.M. Pooley, P.A. Forsyth, K.R. Vetzal

Improved Monte Carlo Linear Solvers Through Non-diagonal Splitting 168
A. Srinivasan, V. Aggarwal

Valuation of American Options Using Direct, Linear Complementarity-Based Methods 178
M.D. Koulisianis, T.S. Papatheodorou

Testing the Expectations Hypothesis for Interest Rate Term Structure: Some Australian Evidence 189
V. Fang, V.C.S. Lee

A Fourth Order L-stable Method for the Black-Scholes Model with Barrier Options 199
D.A. Voss, A.Q.M. Khaliq, S.H.K. Kazmi, H. He

Ranking Decision Variants by Subjective Paired Comparisons in Cases with Incomplete Data 208
M. Kwiesielewicz, E. van Uden

Quantum Games and Minimum Entropy 216
E. Jiménez

Distributed Solution of High-Order Compact Difference Schemes for Multidimensional Convection-Diffusion Equations 226
M.F. Dixon, K. Tan

Neural Network for Modeling Financial Time Series: A New Approach 236
C. Slim, A. Trabelsi

Theme: Scientific Visualization and Image Processing

Illumination, Visualization, and Rendering

Camera Models and Optical Systems Used in Computer Graphics: Part I, Object-Based Techniques 246
B.A. Barsky, D.R. Horn, S.A. Klein, J.A. Pang, M. Yu

Camera Models and Optical Systems Used in Computer Graphics: Part II, Image-Based Techniques 256
B.A. Barsky, D.R. Horn, S.A. Klein, J.A. Pang, M. Yu

Moment Based Transfer Function Design for Volume Rendering 266
Z. Jiawan, S. Jizhou, S. Zhigang, W. Zunce

Information-Theory-Based Oracles for Hierarchical Radiosity 275
J. Rigau, M. Feixas, M. Sbert

A Hybrid Scheme for Interactive Rendering a Virtual Environment 285
T.-Y. Lee, P.-H. Lin, T.-H. Yang

Fast Adaptive Selection of Best Views 295
P.-P. Vázquez, M. Sbert

Perception-Based Illumination Information Measurement and Light Source Placement 306
P.-P. Vázquez, M. Sbert

A Study on the Dynamic Painterly Stroke Generation for 3D Animation . . 317
H.K. Lee, Y.S. Park, K.H. Yoon

A Directional Stroke Generation Algorithm for Painterly Rendering 326
J.S. Cha, Y.S. Park, K.H. Yoon

A Simple Method for Ray Tracing Diffraction 336
E. Agu, F.S. Hill Jr.

A Multiple Depth Buffer Implementation for Radiosity 346
R. Martínez, L. Szirmay-Kalos, M. Sbert

Geometry, Modeling, and Computer-Aided Geometric Design

Solving Geometric Constraints Invariant Modulo the Similarity Group 356
É. Schramm, P. Schreck

Triangular Bézier Surfaces of Minimal Area 366
A. Arnal, A. Lluch, J. Monterde

Projection Filling Based on Contour Structural Points 376
D. Yu, H. Dong, W. Lai, Y. Yang

A Graph Based Algorithm for Intersection of Subdivision Surfaces 387
S. Lanquetin, S. Foufou, H. Kheddouci, M. Neveu

Implicit Polynomial Support Optimized for Sparseness 397
I.Z. Emiris, I.S. Kotsireas

"CONTOUR" Algorithm for Finding and Visualizing Flat Sections of 3D-Objects 407
D.V. Mogilenskikh

A Non-uniform Binary Space Partition Algorithm for 2D Implicit Curves . 418
F. Morgado, A. Gomes

Navigation, Compression, Meshes, Data Management

Web Visualization of Function-Defined Shapes 428
A. Sourin, L.F. Min, K. Levinski

Planar Subdivisions by Radical Axes Applied to Structural Morphology ... 438
R. Togores, C. Otero

Distribution of Vertex Indices in Edgebreaker 448
Y. Cho, D.-S. Kim, H. Kim, H.C. Lee, J.Y. Park

A Comparison Study of Modern Heuristics for Solving the Partitioning Problem in Distributed Virtual Environment Systems 458
P. Morillo, M. Fernández, J.M. Orduña

Optimal Exploitation of Client Texture Hardware Capabilities on a Client-Server Remote Visualization Framework 468
I. Boada, I. Navazo

AIF - A Data Structure for Polygonal Meshes 478
F.G.M. Silva, A.J.P. Gomes

High Dynamic Range Image Texture Mapping Based on VRML 488
S.-Y. Kim, B.-T. Choi

CW Complexes: Topological Mainframe for Numerical Representations of Objects 498
P. Desbarats, S. Gueorguieva

Reconstruction, Approximation, and Morphing

Approximating 3D General Sweep Boundary Using Depth-Buffer 508
J. Ahn, S.J. Hong

Morphology-Based 3D Volume Metamorphosis 518
T.-Y. Lee, C.-H. Lin, W.-H. Wang

Self-correctional 3D Shape Reconstruction from a Single Freehand Line Drawing 528
B.S. Oh, C.H. Kim

3D Primitive Reconstruction Using the Line Segment with Single Image . . 539
S.-D. Cho, S.-T. Ryoo, T.-H. Choi, K.-H. Yoon

CIGRO: A Minimal Instruction Set Calligraphic Interface for Sketch-Based Modeling 549
M. Contero, F. Naya, J. Jorge, J. Conesa

Computer Art: Computer Graphics Applications

MOTRICO Project: Geometric Construction and Mesh Generation of Blood Vessels in Coronary Bifurcation 559
F.J. Seron, E. Garcia, J. del Pico

Analysis Tool for Cataloguing Textile and Tile Pattern Designs 569
M. Valor, F. Albert, J.M. Gomis, M. Contero

Animation and Motion

Urban Site Modeling from LiDAR 579
S. You, J. Hu, U. Neumann, P. Fox

An Automatic Outline Fitting Algorithm for Arabic Characters 589
M. Sarfraz, M.A. Khan

Hardware-Accelerated Real-Time Rendering for 3D Sumi-e Painting 599
S.-J. Kang, S.-J. Kim, C.-H. Kim

Generation of Symmetrical Patterns Based on Text and Colour 609
A. Salman, R. Salman, E. Love

Reuse of Motion Capture Data in Animation: A Review 620
W. Geng, G. Yu

A Real-Time Natural Motion Edit by the Uniform Posture Map Algorithm 630
J.O. Kim, B.R. Lee, C.H. Chung

A Feature-Preserved Simplification for Autonomous Facial Animation from 3D Scan Data 640
S.-K. Kim, S.-J. Kim, C.-H. Kim

Human Motion Tracking by Combining View-Based and Model-Based Methods for Monocular Video Sequences 650
J. Park, S. Park, J.K. Aggarwal

Animating Behavior of Virtual Agents: The Virtual Park 660
F. Luengo, A. Iglesias

Computer Aided Design and Computer Aided Manufacturing

A Curve Design Method with Shape Control 670
M. Sarfraz, M. Balah

Determination of Cutting Direction for Minimization of Tool Retraction Length in Zigzag Pocket Machining 680
B.K. Kim, J.Y. Park, H.C. Lee, D.-S. Kim

Digital Styling for Designers: Sketch Emulation in Computer Environment 690
S.-H. Bae, W.-S. Kim, E.-S. Kwon

Digital Styling for Designers: 3D Plane-Symmetric Freeform Curve Creation Using Sketch Interface 701
S.-H. Bae, R. Kijima, W.-S. Kim

Directional Offset of a Spatial Curve for Practical Engineering Design 711
H. Shin, S.K. Yoo, S.K. Cho, W.H. Chung

Task-Level Assembly Modeling in Virtual Environments 721
B. Jung

Design of a New Test Part for Benchmarking the Accuracy and Surface Finish of Rapid Prototyping Processes 731
H.-S. Byun, K.H. Lee

Automated Scan Plan Generation Using STL Meshes for 3D Stripe-Type Laser Scanner 741
S. Son, K.H. Lee

An Efficient Algorithm for Real-Time 3D Terrain Walkthrough 751
M. Hesse, M.L. Gavrilova

Computational Geometry and Applications

Geometric Graphs for Improving Nearest Neighbor Decision Rules 762
G. Toussaint

Red-Blue Separability Problems in 3D 766
F. Hurtado, C. Seara, S. Sethia

Two-Dimensional Range Search Based on the Voronoi Diagram 776
T. Kanda, K. Sugihara

Computing a Closest Point to a Query Hyperplane in Three and Higher Dimensions 787
P. Mitra, A. Mukhopadhyay

Computing a Largest Empty Arbitrary Oriented Rectangle: Theory and Implementation 797
A. Mukhopadhyay, S.V. Rao

An Approximate Morphing between Polylines 807
S. Bespamyatnikh

Efficient Proximity Search for 3-D Cuboids 817
J. Gao, R. Gupta

An Explicit Solution for Computing the Euclidean d-dimensional Voronoi Diagram of Spheres in a Floating-Point Arithmetic 827
M.L. Gavrilova

Dynamically Maintaining a Hierarchical Planar Voronoi Diagram Approximation 836
I. Boada, N. Coll, J.A. Sellarès

Voronoi Diagram of Circles in a Large Circle 847
D.-S. Kim, D. Kim, K. Sugihara

On Estimating Result Sizes of Multi-way Spatial Joins 856
H.-H. Park

The Integer Hull of a Convex Rational Polytope 866
J.B. Lasserre

Straight-Line Drawings of General Trees with Linear Area and Arbitrary Aspect Ratio 876
A. Garg, A. Rusu

Connected Guards in Orthogonal Art Galleries 886
V. Pinciu

Four Colouring the Vertices of the Triangulation of a Polygon Containing a Hole 894
G.M. Seed, D.E.R. Clark, R. Ocone, X.Y. Yang

Partitioning Polygons into Tree Monotone and Y-monotone Subpolygons . 903
R.P. Boland, J. Urrutia

Finding Coarse Grained Parallelism in Computational Geometry Algorithms 913
V. Beletskyy

On the Reliability of Triangle Intersection in 3D 923
S. Robbins, S. Whitesides

A Parametrically-Optimized Morphing Scheme of Polygonal Meshes 931
J. Shen

Author Index 941

Table of Contents, Part I

Invited Paper

Random Number Generators with Period Divisible by a Mersenne Prime .. 1
R.P. Brent, P. Zimmermann

Scientific Visualization

Visualizing Large Relational Datasets by Combining Grand Tour with Footprint Splatting of High Dimensional Data Cubes 11
L. Yang

Visualizing Frequent Itemsets, Association Rules, and Sequential Patterns in Parallel Coordinates 21
L. Yang

Information Extraction to Generate Visual Simulations of Car Accidents from Written Descriptions 31
P. Nugues, S. Dupuy, A. Egges

Interactive Stereoscopic Visualization of Very Large Visible Men on CAVE 41
I. Ihm, B. Lee, J.-Y. Lee, M. Joh, S. Park

Panoramic Spherical Video – The Space Ball 51
J.E. Coleshill, A. Ferworn

Human Expressions Interaction Between Avatar and Virtual World 59
R. Liang, C. Chen, Z. Pan, J. Bu

Image Processing

Progressive Curve Representation Based on Reverse Subdivision 67
F.F. Samavati, M.A. Nur, R. Bartels, B. Wyvill

Concept-Based Image Retrieval Using the New Semantic Similarity Measurement 79
J. Choi, M. Cho, S.H. Park, P. Kim

Case- and Rule-Based Algorithms for the Contextual Pattern Recognition Problem 89
M. Wozniak

A Simple Algorithm for Object Location from a Single Image without Camera Calibration 99
M. Bénallal, J. Meunier

Spatial Enhancement of Digital Terrain Model Using Shape from Shading with Single Satellite Imagery 105
M.A. Rajabi, J.A.R. Blais

Robust Real-Time Face Tracking and Modeling from Video 114
R. Liang, C. Chen, Z. Pan, J. Bu

Computer, Mobile, and Data Networks

Prediction Mechanism for Accessing Web Sites for Mobile Clients 124
G. González S., R.A. Pazos R., V.J. Sosa S.

Adaptive and Scalable Allocation of Data-Objects in the Web 134
J. Pérez O., R.A. Pazos R., D. Romero, R. Santaolaya S., G. Rodriguez O., V. Sosa S.

Intelligent Fault-Tolerant Web Caching Service on Application Level Active Networks 144
S. Hong, K. Kim, S. Han

Dynamic Clustering of Web Search Results 153
L. Yang, A. Rahi

A Resilient Web Caching Architecture 160
V.J. Sosa Sosa, G. González S., L. Navarro, J. Pérez O.

A Context-Based Information Agent for Supporting Education on the Web 170
M. Abdel Razek, C. Frasson, M. Kaltenbach

The Two-Criteria Topological Design Problem in WAN with Delay Constraint: An Algorithm and Computational Results 180
M. Gola, A. Kasprzak

Mobile IPv6 Fast Handover Mechanism in Wireless LAN with Several Access Routers 190
P.S. Kim, J.H. Lee

Symmetrical Declustering: A Load Balancing and Fault Tolerant Strategy for Clustered Video Servers 199
S. Wu, H. Jin, G. Tan

Efficient and Fast Discovery of Slave Home Agent's Address in Mobile IPv6 209
P.S. Kim, Y.K. Kim

Adaptive Information Exchange with Field Bus Systems 217
S. Eberle, P. Göhner

A Replication Protocol with Composite Topology for High Adaptability . . 227
S.C. Choi, H.Y. Youn, J.K. Park, J.S. Choi

How Secure Personal Mobility Can Be? 238
S. Vincze

Algorithms and Applications

Generation of Valid Labeled Binary Trees 245
B. Effantin

Scheduling Intervals Using Independent Sets in Claw-Free Graphs 254
D.R. Gaur, R. Krishnamurti

A Novel Approach to Optimal Packing Using Composite Object Geometry 263
M.M. Atiqullah, E.M. Crespo

A Discrete Farkas Lemma 273
J.B. Lasserre

Elementary Encoding by Evolutionary Approach 282
I. Vasyltsov

An Efficient Algorithm for the Compression of Time-Dependent Data from Scientific Simulations 291
C. Li, P.E. Plassmann

Rate Control for Low Bit Rate Video via Enhanced Frame Skipping 302
J.-Y. Pyun, Y. Kim, S.-J. Ko

Automatic Object-Based Video Segmentation Using Distributed Genetic Algorithms 312
E.Y. Kim, S.H. Park

3D Protein Peptide Chain Search Using an Improved Genetic Algorithm . . 322
Z. Yang, G. Liu

Efficient Inversion Algorithm for Optimal Normal Bases Type II 330
H.S. Yoo, E.S. Kim

Dual Space Drawing Methods of Cylinders 338
R. Ding

Implementation of Smartcard Using Multi Authentication Method 348
S.-E. Lee, H.-J. Jang, S.-Y. Han

Textual CBR for Incident Report Retrieval 358
D.C. Wilson, J. Carthy, K. Abbey, J. Sheppard, R. Wang, J. Dunnion, A. Drummond

The Integral Dictionary: A Lexical Network Based on Componential Semantics 368
D. Dutoit, P. Nugues, P. de Torcy

Classification Methods Using Neural Networks and Partial Precedence Algorithms for Differential Medical Diagnosis: A Case Study 378
A.F. Kuri-Morales, M.R. Ortiz-Posadas

Consistent and Efficient Density Estimation 388
H. Hendriks, P.T. Kim

Investigation of Geometric Shapes of Hydrodynamic Structures for Identification of Dynamical States of Convective Liquid 398
A.M. Krot, P.P. Tkachova

Restoration of Dynamical Systems Attractors and Estimation of Their Geometric Characteristics into State-Space 407
A.M. Krot, H.B. Minervina

Parallel Genetic Algorithms for the Tuning of a Fuzzy AQM Controller . . 417
G. Di Fatta, G. Lo Re, A. Urso

New Robust Algorithm for Nonlinear Robot Systems 427
J.-G. Kim, K.h. Park

Knowledge Representation and Artificial Intelligence

Scalable Knowledge Discovery in Point-to-Multipoint Environments 437
S. Cho

Knowledge Grid Support for Treatment of Traumatic Brain Injury Victims 446
P. Brezany, A.M. Tjoa, M. Rusnak, J. Brezanyova, I. Janciak

Neural Based Approach to Keyword Extraction from Documents 456
T. Jo

Intelligent Crowd Simulation 462
L. Feng, R. Liang

Simulation and Modeling

Parallel Coupled Thermomechanical Simulation Using Hybrid Domain Decomposition 472
P.A. Adamidis, M.M. Resch

Three Dimensional Modelling of Combustion in a Direct Injection Diesel Engine Using a New Unstructured Parallel Solver 483
J. Bohbot, M. Zolver, D. Klahr, A. Torres

Computational Modelling of Particle Degradation in Dilute Phase Pneumatic Conveyors 493
P. Chapelle, N. Christakis, H. Abou-Chakra, U. Tuzun, I. Bridle, M.S.A. Bradley, M.K. Patel, M. Cross

Simulation of Chemical Reaction Fronts in Anaerobic Digestion of Solid Waste 503
H.J. Eberl

Numerical Studies of Turbulent Jet Hydrogen/Air Diffusion Flames with Local Quenching Using Stiff Complex Chemistry 513
X. Zhou, D.L. Hitt

Computational Modeling of Geophysical Systems 523
S. Davenport, P. Seshaiyer

Modeling and Simulation of an Experimental Manufacturing Cell 533
N. Rakoto-Ravalontsalama, J.L. Villa

A Study of Three-Dimensional Preferential Flow Using a Web-Based Simulation System 539
V.J. Alarcon, H. Zeng, W. Kingery, H.M. Selim, J. Zhu

Mathematical Modeling of Crown Forest Fires Initiation 549
V. Perminov

Numerical Simulation of Free Surface MHD Flows: Richtmyer - Meshkov Instability and Applications 558
R. Samulyak, J. Glimm, W. Oh, H. Kirk, K. McDonald

Parallel Computing for Semiquantal Few-Body Systems in Atomic Physics 568
R.A. Sultanov, D. Guster

Modelling Traffic Flow at a Multilane Intersection 577
R. Wang, H.J. Ruskin

Some Recent Results on a Domain Decomposition Method in Biomechanics of Human Joints 587
J. Nedoma, I. Hlaváček, J. Daněk, P. Vavřík, J. Stehlík, F. Denk

The Design and Implementation of Chinese Question and Answering System 601
I-H. Meng, W.-P. Yang

Computational Aspects of Data Mining 614
F.A. Mărginean

Dynamic System Design for the Scrap-Processing Problem under JIT 623
J. Ding, R. Karunamuni

Extracting Motion Model Parameters with Robust Estimation 633
S.-W. Jang, M. Pomplun, G.-Y. Kim, H.-I. Choi

A Scientific On-line Database for Efficient Function Approximation 643
I. Veljkovic, P.E. Plassmann, D.C. Haworth

Programming Methods in Computer Simulation

Return of an Object in Two-Stack Dynamic Memory 654
G. Bassen, I. Garber

Programming in a High Level Approach for Scientific Computing 663
B. Ge

A Method to Evaluate Computational Programs Using the Logical Combinatorial Approach 674
M.R. Ortiz-Posadas, R.L.S. Dazzi, M. Nazaré M.A. Hahne

The Efficient Debugging System for Locating Logical Errors in Java Programs 684
H.-J. Kouh, W.-H. Yoo

A New Definitional Specification Language 694
A. Redouane

Restructuring Conditional Code Structures Using Object Oriented Design Patterns 704
R. Santaolaya S., O.G. Fragoso D., J. Pérez O., L. Zambrano S.

Explicit Surrogate Dual Formulation and Algorithm for Quadratic Programming 714
Z.C. Xuan, K.H. Lee

Relationships Between Selected Software Measures and Latent Bug-Density: Guidelines for Improving Quality 724
S.C. Misra, V.C. Bhavsar

An Analysis of Idle CPU Cycles at University Computer Labs 733
S. Hwang, K. Jeong, E. Im, C. Woo, K.-S. Hahn, M. Kim, S. Lee

Self-configuring Algorithm for Software Fault Tolerance in (n,k)-way Cluster Systems 742
C. Choi, S. Kim

Parallel High-Level Image Processing on a Standard PC 752
M.F. Ercan, Y.F. Fung

Connection Mechanism Using Agent for Streaming in Multimedia Messaging Service Environment 761
J. Hong, J. Song, S. Han, K. Kim

Complex Dynamics and Financial Fragility in an Agent Based Model 770
M. Gallegati, G. Giulioni, N. Kichiji

Functional Analysis

Uniform Approximations for Transcendental Functions 780
S. Winitzki

Computing the Incomplete Gamma Function to Arbitrary Precision 790
S. Winitzki

On Monotonic Solutions of an Integral Equation Related with the Chandrasekhar Equation 799
J. Caballero, J. Rocha, K.B. Sadarangani

A Compact Scheme for the Streamfunction Formulation of Navier-Stokes Equations 809
D. Fishelov, M. Ben-Artzi, J.-P. Croisille

Large Scale Computing

A Performance Comparison of Matrix Solvers on Compaq Alpha, Intel Itanium, and Intel Itanium II Processors 818
C.J.K. Tan, D. Hagan, M. Dixon

The Role of Multi-method Linear Solvers in PDE-based Simulations 828
S. Bhowmick, L. McInnes, B. Norris, P. Raghavan

Time-Memory Trade-Offs Using Sparse Matrix Methods for Large-Scale Eigenvalue Problems 840
K. Teranishi, P. Raghavan, C. Yang

Computer Technology for Solving Large Scale Matrix Problems 848
O.D. Joukov, N.D. Rishe

A Compact and Fast Division Architecture for a Finite Field $GF(2^m)$ 855
C.H. Kim, S. Kwon, J.J. Kim, C.P. Hong

Montgomery Multiplication and Squaring Algorithms in GF(2^k) 865
N.-Y. Kim, K.-Y. Yoo

Time-Space Efficient Exponentiation over GF(2^m) 875
K.-M. Ku, K.-J. Ha, K.-Y. Yoo

Efficient Architecture for Exponentiation and Division in GF(2^m) Using Irreducible AOP 883
W.-H. Lee, Y.-J. Heo, K.-Y. Yoo

A High-Performance UL Factorization for the Frontal Method 893
M. Bianco

The p-th Order Necessary Optimality Conditions for Inequality–Constrained Optimization Problems 903
O. Brezhneva, A. Tret'yakov

A Direct Method for Block-Toeplitz Systems with Applications to Re-manufacturing Systems 912
W.-K. Ching, M.M. Ng, W.-O. Yuen

Reduced Fractal Analysis of the Multidimensional Attractor Reconstructed from Chaotic Time Series 921
V.F. Dailyudenko

Resource Management for Finite Element Codes on Shared Memory Systems 927
J. Hungershöfer, J.-M. Wierum, H.-P. Gänser

n-Tuples of 0s and 1s: Necessary and Sufficient Conditions for Intrinsic Order 937
L. González

Computational Algorithm and Architecture for AB^2 Multiplication in Finite Fields 947
J.-C. Jeon, K.-Y. Yoo

A Novel De-interlacing Technique Using Bi-directional Motion Estimation . 957
Y. Kim, K.-S. Choi, J.-Y. Pyun, B.-T. Choi, S.-J. Ko

On the Design and Implementation of a Portable DSM System for Low-Cost Multicomputers 967
F. Meza, A.E. Campos, C. Ruz

On a Coupled Thermo-Hydro-Mechanical Problem Based on the Thermo-Visco-Plastic Rheology 977
J. Nedoma

Parallel Genetic Algorithm for a Flow-Shop Problem with Multiprocessor Tasks 987
C. Oğuz, Y.-F. Fung, M.F. Ercan, X.T. Qi

A New Availability Concept for (n,k)-way Cluster Systems Regarding Waiting Time 998
K. Park, S. Kim, J.-C. Liu

A Study of Tuning Hyperparameters for Support Vector Machines 1006
Y. Quan, J. Yang, C. Ye

A Method for Discretization in Time Based on Cayley Transform for Parabolic Transmission Problem 1016
N. Rossokhata

Design and Implementation of a Parallel Prime Edge-Length Symmetric FFT 1025
J. Seguel

A Posteriori Output Bound for Partial Differential Equations Based on Elemental Error Bound Computing 1035
Z.C. Xuan, K.H. Lee, J. Peraire

A Bird's Eye View of Matrix Distributed Processing 1045
M. Di Pierro

Author Index 1053

Table of Contents, Part II

Modeling of Location Management in Mobile Information Systems

Dynamic Recovery Algorithm for the Multiple Failures in the WDM Based UMTS Networks 1
H.C. Kim, S. Ahn, J.W. Chung

Efficient Queuing Management Scheme with Image/Voice/Data Calls for Reducing Blocking Rate in Multimedia Mobile Networks 11
W.S. Na, D.C. Lee, I.T. Ryoo

Anomaly Detection Scheme Using Data Mining in Mobile Environment . . . 21
K.-j. Park, H.-b. Ryou

Network-Based Real-Time Connection Traceback System (NRCTS) with Packet Marking Technology 31
Y.-S. Choi, D.-i. Seo, S.-W. Sohn, S.-H. Lee

Mobile Broadcasting Copyrights Model Based on XrML 41
C.-W. Kang, H. Yoo, H.K. Jung

Mobility Management Scheme for Reducing Location Traffic Cost in IMT-2000 Networks 50
D.C. Lee, J.G. Kim, K.J. Kim

Web-Based Object-Oriented Modeling on Internet 60
S.H. Cho, H.-J. Kim, D.C. Lee

Automatic Differentiation and Applications

Uncertainty Analysis Based on Sensitivities Generated Using Automatic Differentiation 70
J. Barhen, D.B. Reister

Evaluation of a Computer Model for Wavy Falling Films Using EFCOSS . . 78
C.H. Bischof, H.M. Bücker, A. Rasch, E. Slusanschi

Automatic Differentiation for Optimum Design, Applied to Sonic Boom Reduction 85
L. Hascoët, M. Vázquez, A. Dervieux

An Example of an Automatic Differentiation-Based Modelling System 95
T. Kaminski, R. Giering, M. Scholze, P. Rayner, W. Knorr

Coupling Tangent-Linear and Adjoint Models 105
U. Naumann, P. Heimbach

Hierarchical Automatic Differentiation by Vertex Elimination and Source Transformation 115
M. Tadjouddine, S.A. Forth, J.D. Pryce

Mobile Computing: Routing and Handoff

An Enhanced Hand-Off Policy to Reduce Blocking Probability in Two-Layer Wireless Systems 125
T.-J. Lee

On Bandwidth Adjusted Multicast in Pipelined Routing Architecture for Mobile Environment 135
Y.-C. Bang, W.-S. Jang, B.J. Ahn, J.-H. Hahm

Cost Effective Location Management Scheme Based on Hierarchical Mobile IPv6 144
D.K. Choi, H. Choo, J.-K. Park

On Employing Hierarchical Structure in PCS Networks 155
I.-H. Shin, G.-L. Park

Design of Mobile IP Extension for Nested Mobile Network 163
H. Kang, K. Kim

Dual Transmission Architecture for Mobile Multicasting 171
K.-Y. Park, S.-Y. Han, B.-G. Joo

Mobile Computing: Authentication Technology

Secure Source Specific Multicast for Networks Including Mobile Hosts 181
Y.-C. Shim, J.-C. Park

Security Analysis and Applications of Standard Key Agreement Protocols . 191
S. Oh, J. Kwak, S. Lee, D. Won

An Improved Password Authentication Key Exchange Protocol for 802.11 Environment 201
S.J. Yu, J.S. Song

A Ticket-Based AAA Security Mechanism in Mobile IP Network 210
J.-M. Park, E.-H. Bae, H.-J. Pyeon, K. Chae

Secure and Scalable Mobile IP Registration Scheme Using PKI 220
J.-P. Yoo, K. Kim, H. Choo, J.-i. Lee, J.S. Song

Secure Session Key Exchange for Mobile IP Low Latency Handoffs 230
H.G. Kim, D.H. Choi, D.Y. Kim

Coarse Grained Parallel Algorithms for Scientific Applications

A Coarse-Grained Multicomputer Algorithm for the Longest Repeated Suffix Ending at Each Point in a Word 239
T. Garcia, D. Semé

A Parallel Wavefront Algorithm for Efficient Biological Sequence Comparison 249
C.E.R. Alves, E.N. Cáceres, F. Dehne, S.W. Song

Mining Multivariate Time Series Models with Soft-Computing Techniques: A Coarse-Grained Parallel Computing Approach 259
J.J. Valdés, A.J. Barton

Towards Realistic Implementations of External Memory Algorithms Using a Coarse Grained Paradigm 269
J. Gustedt

Designing an Efficient Partitioning Algorithm for Grid Environments with Application to N-body Problems 279
D.J. Harvey, S.K. Das, R. Biswas

Coarse-Grained Parallel Matrix-Free Solution of a Three-Dimensional Elliptic Prototype Problem 290
K.P. Allen, M.K. Gobbert

Parallel *CLUSTAL W* for PC Clusters 300
J. Cheetham, F. Dehne, S. Pitre, A. Rau-Chaplin, P.J. Taillon

Theme: Information Systems and Technologies

Security and Watermarking

A Robust Watermarking Technique in Geometric Distortion of Digital Image 310
W. Kim, I.-J. Ko, N.-Y. Lee, G.-Y. Kim, H.-I. Choi

Design of Secure Monitor in the Secure Zone Cooperation 320
G. Ahn, B. Chang, T.M. Chung

A Proposal for DoS-Defensive Internet Key Exchange 328
M.S. Choi, D.J. Kwak, S.J. Moon

A Random M-ary Method Based Countermeasure against Side Channel Attacks 338
M.K. Ahn, J.C. Ha, H.J. Lee, S.J. Moon

Wavelet-Based Blind Watermarking Technique for Real-Time Watermark Interpretation 348
K.-S. Yoo, W.-H. Lee

Design of Active HoneyPot System 356
M. Kim, M. Kim, H.K. Lee, Y. Mun

Localized Authentication Scheme Using AAA in Mobile IPv6 365
M. Kim, J. Park, M. Kim, Y. Mun

Autonomous Agents-Based Security Infrastructure 374
S. Corsava, V. Getov

Graphics and Image

The Modified Block Matching Algorithm for a Hand Tracking of an HCI System 383
J.O. Kim, H.J. Chang, C.H. Chung

Face Retrieval Using 1st- and 2nd-order PCA Mixture Model 391
S. Lee, H.-C. Kim, D. Kim, Y.S. Choi

Concept Based Image Retrieval Using the Domain Ontology 401
W. Kim, H. Kong, K. Oh, Y. Moon, P. Kim

On Natural Motion Editing by a Geometric Mean Filter 411
J.O. Kim, C.H. Oh, C.H. Chung, J. Hwang, W. Lee

High Quality VoD Service Architecture on Broadband Internet 419
S. Lee

Geometric Querying for Dynamic Exploration of Multidimensional Data . . 427
O. Sourina

Mobile and Wireless Networking

Design of Dynamic Slot Assignment Protocol for Wireless Communication 437
H. Yoe, Y. Lee, J. Koh

Adaptive Modulation and Coding of MIMO in Next Generation Mobile Systems 445
S. Ro, I. Hwang, D. Hong, C. Kang, M. Kang

Performance Improvement of DS/CDMA System under Impulsive Noise Environments 455
H.G. Kang, S.E. Cho, H. Yoe, J.G. Koh, Y.S. Choi

The Modeling and Traffic Feedback Control for QoS Management on Local Network 463
J. Park, E.-N. Huh, Y. Mun, B.-G. Lee

Design and Verification of Interworking Protocol for CC and SIP in Generation Mobile Network 472
S.-H. Jung, S.-C. Park

VOD Service Using Web-Caching Technique on the Head-End-Network . . . 482
I. Kim, B. Kim, Y. Woo, T. Hwang, S. Kang

Operating Systems Techniques and Databases

Hybrid Scheme for Ubiquitous Computing 490
H.-K. Kim

An Enhanced Main Memory Index Structure Employing the Level Prefetching Technique 498
H.T. Hong, J.I. Pee, S.I. Song, J.S. Yoo

Reusable Software Component Retrieval System Based on Web 507
K.S. Young, C.D. Oun

A Hybrid Concurrency Control with Deadlock-free Approach 517
S.H. Cho

Weak Consistency in Hybrid Group Update for Replication 525
B.-w. Lee

Experimental Evaluation of Query Fingerprinting with Embedded Search Term Markers 532
B. Kim

Software Development

An Efficient Optimization Algorithm of Java Bytecode to Reduce Network Traffic 542
D.-W. Kim, M.-S. Jung

An Efficient Small Sized On-Card Verifier for Java Card 552
J.-B. Cho, M.-S. Jung, S.-I. Jun

Distributed Object-Oriented Parallel Programming Environment on Grid . 562
Y.-J. Woo, C.-S. Jeong

An Architecture for C-commerce System Design 571
H. Park, W. Suh

A Probabilistic Model for Predicting Software Development Effort 581
P.C. Pendharkar, G.H. Subramanian, J.A. Rodger

Knowledge Acquisition in the Nutri-Fuzzy-ORIXAS Project 589
V.M. Benjamim Werneck, A. Brito F. Oliveira, R. Serrão Lanzillotti, H. Serrão Lanzilotti, E. de Abreu Soares, E. Souza Portella, S. da Silva Ávila

Speech Recognition and Agent Technologies

The Merging Algorithm for an Extraction of Valid Speech-Sounds 599
J.O. Kim, H.W. Paek, C.H. Chung, W.Y. Yim, S.H. Lee

Agent Migration Information System for the Efficient Migration of the Mobile Agent 607
H.-J. Park

Schematic Aspect for Autonomous Agent 614
K.S. Tae

Robust Speaker Recognition Against Utterance Variations 624
J.J. Lee, J.Y. Rheem, K.Y. Lee

Efficient Speaker Identification Based on Robust VQ-PCA 631
Y. Lee, J. Lee, K.Y. Lee

An Amplitude Warping Approach to Intra-speaker Normalization for Speech Recognition 639
K.-S. Hong

Computational Theory and Test and Simulation

Scenario Based Testing & Test Plan Metrics Based on a Use Case Approach for Real Time UPS (Uninterruptible Power System) 646
R.Y.-C. Kim, B.-G. Joo, K.-C. Kim, B.-k. Joen

A Study on Insuring the Full Reliability of Finite State Machine 656
S. Yang, M.J. Kim, J.H. Park, H. Chang

Distributed Multiple-Path Searching Algorithm for Fault Detection 664
S. Kim, S. Ahn, J.W. Chung

An Object-Oriented Simulation System for Air Defense 674
C.-S. Jeong, S.-Y. Choi

Evaluation and Analysis of Computational Complexity for Secure Multicast Models 684
E. Blessing R., R. Uthariaraj

An Algorithm for Diagnosing System with Structured Description 695
S. Luan, G. Dai

Wave Phenomena in Physics and Engineering: New Models, Algorithms, and Applications

FETD Simulation of Wave Propagation Modeling the Cari Breast Sonography 705
A. Bounaïm, S. Holm, W. Chen, Å. Ødegård, A. Tveito, K. Thomenius

Finite Element Simulation of BAW Propagation in Inhomogeneous Plate Due to Piezoelectric Actuation 715
A. Chakraborty, D. Roy Mahapatra, S. Gopalakrishnan

Exact Solutions of the Generalized Equal Width Wave Equation 725
S. Hamdi, W.H. Enright, W.E. Schiesser, J.J. Gottlieb

Quantum Dot and Acoustic Enclosure Problems in Lens-Shaped Structures 735
M. Willatzen and L.C. Lew Yan Voon

A Nonlinear Spectral Finite Element Model for Analysis of Wave Propagation in Solid with Internal Friction and Dissipation 745
D. Roy Mahapatra, S. Gopalakrishnan

Finite Element Analysis of Nanowire Superlattice Structures 755
M. Willatzen, R.V.N. Melnik, C. Galeriu, L.C. Lew Yan Voon

On a Class of Boussinesq Equations for Shallow Water Waves 764
P. Daripa, R.K. Dash

Elastic Waves Propagation in Damped Media 774
E.L. Albuquerque, P.W. Mauriz

A Fast Algorithm for Moving Interface Problems 782
S. Dutta, J. Glimm, J.W. Grove, D.H. Sharp, Y. Zhang

Computational Aspects of Conservative Difference Schemes for Shape Memory Alloys Applications 791
R.V.N. Melnik, L. Wang, P. Matus, I. Rybak

Optimization in the Context of Active Control of Sound 801
J. Lončarić, S. Tsynkov

On Disintegration of Cellular Flames 811
L. Kagan, S. Minaev, G. Sivashinsky

Investigation of a Three-Dimensional Spectral Element Method for Helmholtz's Equation 819
O.Z. Mehdizadeh, M. Paraschivoiu

Effect of Vortex Profile on Sound Generation in a Non-uniform Flow 826
A. Povitsky, T. Zheng, G. Vatistas

Numerical Simulation of Laminar Mixing Surfaces in Converging Microchannel Flows 837
M. McGarry and D.L. Hitt

Preconditionning Techniques for the Solution of the Helmholtz Equation by the Finite Element Method 847
R. Kechroud, A. Soulaimani, Y. Saad

Direct Numerical Simulations of Instability-Wave Generation and Propagation in Supersonic Boundary Layers 859
L. Jiang, M. Choudhari, C.-L. Chang, C. Liu

Modeling of Plume Dynamics and Shock Wave in Laser Ablation with Application to Nano-Technology 871
D.C. Lobão, A. Povitsky

Monte Carlo Simulation of Spin-Polarized Transport 881
M. Shen, S. Saikin, M.-C. Cheng, V. Privman

Web-Based Learning

A Web-Based Environmental Education System for Elementary School Students 892
Y. Chae, S. Hong, W. Jun

Evaluating the Web-Based Instruction by Item Analysis 901
E. Chi, C. Park, H. Rim

Supporting Transactional Service for Virtual School Environments 909
S. Hong, W. Jun

Design and Implementation of a Hybrid Instruction Model for Web-Based Learning 916
C. Park, M. Kim

Design and Implementation of a Web-Based Information Literacy Cultivation System for Emotionally Disturbed Students 924
G. Lee, S.-K. Hong, W. Jun

Author Index 935

A Variable Resolution Approach to Cluster Discovery in Spatial Data Mining

Allan J. Brimicombe

Centre for Geo-Information Studies, University of East London, UK
a.j.brimicombe@uel.ac.uk

Abstract. Spatial data mining seeks to discover meaningful patterns from data where a prime dimension of interest is geographical location. Consideration of a spatial dimension becomes important when data either refer to specific locations and/or have significant spatial dependence which needs to be considered if meaningful patterns are to emerge. For point data there are two main groups of approaches. One stems from traditional statistical techniques such as *k*-means clustering in which every point is assigned to a spatial grouping and results in a spatial segmentation. The other broad approach searches for 'hotspots' which can be loosely defined as a localised excess of some incidence rate. Not all points are necessarily assigned to clusters. This paper presents a novel variable resolution approach to cluster discovery which acts in the first instance to define spatial concentrations within the data thus allowing the nature of clustering to be defined. The cluster centroids are then used to establish initial cluster centres in a *k*-means clustering and arrive at a segmentation on the basis of point attributes. The variable resolution technique can thus be viewed as a bridge between the two broad approaches towards knowledge discovery in mining point data sets. Applications of the technique to date include the mining of business, crime, health and environmental data.

1. Introduction

For the spatial sciences, the 1990's were a period of transition from data-poverty to data-richness. Digital spatial data sets have grown rapidly in scope, coverage and volume [1]. This state change has been facilitated by:

- improved technology and wider use of GPS, remote sensing and digital photogrammetry for collecting physical data;
- the introduction of new approaches to obtaining lifestyle and preference data such as through loyalty cards;
- the operation of 'Moore's Law' resulting in increased computing power to process raw data coupled with the falling cost of data storage;
- the advent of data warehousing technologies;
- increasingly efficient ways of accessing and delivering data on-line.

The technical advances in hardware, software and data have been so profound that their effect on the range of problems studied and the methodologies

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 1-11, 2003.
© Springer-Verlag Berlin Heidelberg 2003

used has been fundamental [2]. Thus the exponential rise in the size of databases, their increasingly complex structures and the rate at which they can accumulate on even a daily basis are leading to an urgent need for techniques that can mine very large databases for the knowledge they contain. Spatial data mining can hence be defined as *techniques for the discovery of meaningful patterns from large data sets where a prime dimension of interest is geographical location*.

2. Cluster Detection in Point Data

Most transactions databases with a geographical identifier (whether for business, crime or health) can be regarded as point data sets with geocoding at the resolution of an address. From a location perspective the point is a binary occurrence – either it happened there or it did not. From a data perspective, the binary occurrence has added dimensions of attributes that describe the nature or content of the transaction and may relate to the location, the individual or the event that has been recorded. The traditional approach of non-spatial analyses of attributes may reveal apparently meaningful knowledge but may well be lacking in precision or even misleading if underlying spatial patterns are ignored. The exploratory analysis of point data seeks to establish patterning from which causal processes can be hypothesised or inferred [3] [4]. The analysis of point patterns in geography, ecology and epidemiology have a long tradition (e.g. [5] [6] [7] [8] [9] [10]). Over the last five years, however, two broad thrusts have lead to a renewed interest in analysing point patterns. These are the rise of geocomputation and a re-focusing away from global to local analyses.

The adoption of geocomputational approaches to spatial data analysis represents a paradigm shift in which computers play a pivotal role in the form of analysis as an essential defining ingredient of the science alongside observation, experimentation and theory [11] [12] [13] [14] [15]. The rationale for a geocomputational approach is driven by a rapid increase in the volume of spatial data that are available and necessary for non-trivial analyses and by the growth of computationally intensive simulation models. The tools for geocomputation naturally include geographical information systems (GIS) but they are increasingly viewed as just one class of tool to be used alongside neural networks, artificial intelligence, heuristics, spatial statistics, fuzzy computation, fractals, genetic algorithms, cellular automata, simulated annealing and parallel computing [16]. The other broad thrust has been a re-focusing within quantitative geography on spatial variation at the local level rather than in the search for global patterns [17] [18] [19]. The new emphasis is on exploring and understanding the spatial differences between localities rather than on quantifying their more general, global similarities. Such approaches are often data-rich and geocomputational.

The patterns detected in point data are usually broadly classified as random, uniform or clustered. Spatially random data are usually assumed to have no underlying process of interest that can be modelled, but as [20] has pointed out, such apparent randomness may be attributable to chaotic, complex deterministic patterns. For spatial uniformity a space-filling, mutual exclusion process can be hypothesised. Nevertheless, it is clustered patterns that raise the strongest hypotheses for and interest in identifying underlying processes. Thus spatial cluster detection lies at the

heart of spatial data mining [1] [21] [22] [23] [24] [25] [26] when data either refer to specific locations and/or have significant spatial dependence which needs to be considered if meaningful patterns are to emerge. Within this new context of geocomputation and local analyses, there are two broad approaches to cluster detection wherein which lies a significant dichotomy in the meaning of 'cluster'.

One set of approaches is allied to mainstream statistics of cluster analysis emanating from the work of [27]. Clustering is a means of classification or grouping and clusters are "groups of highly similar entities" [28]. Spatially, cluster analysis will seek to form a segmentation into regions which minimise within-cluster variation but maximise between-cluster variation. There is a general expectation that the clustering mutually exclusively includes all points and is therefore space-filling within the geographical extent of the data. Examples of this approach are to be found in [21] [23] [25] [29]. A widely-used algorithm is the *k*-means clustering [30] due to its relative efficiency in processing large data sets. Its weakness, however, is sensitivity to outliers [29] and the need to specify *ab initio* the number (*k*) of desired clusters or the location of *k* centroids. Such prior specification is counter to the spirit of spatial data mining in which the data themselves should indicate the number and location of clusters rather than speculated by the analyst. This has lead Halls *et al.* [24] and Estivill-Castro & Lee [26] to use Dirichlet and Delaunay diagrams respectively to define spatial clusters. These algorithms, however, will fail where points occupy the same location and to de-duplicate the data set will lead to important data loss.

The other broad set of approaches treat a cluster as a 'hotspot' which can be loosely defined as a localised excess of some incidence rate. This approach is typified by Openshaw's Geographical Analysis Machine (GAM) and its descendents [22] [31] [32]. Similar approaches are based around kernel density functions in which the highest densities form hotspots (e.g. [33] [34] [35]). Whereas the clustering techniques discussed in the previous paragraph derive similarities from a set of attributes, these techniques are used primarily on binary event data. The hotspot approach is the mainstay of spatial epidemiology [36] which seeks to identify any significantly elevated risk above what might be expected from an at-risk background population. Defining a population at risk is clearly critical to this approach and in some, if not many, data mining applications this may not be possible at the outset. Mis-specification of an at-risk background population is likely to lead to erroneous results. Furthermore, intrinsic to this approach is that some of the points form hotspots and the remaining points are no longer the focus of analysis. It is this fundamental difference from the previous set of techniques where every point is assigned to a group. The approach outlined in the rest of this paper aims to bring the two together.

3. A Variable Resolution Approach to the Analysis of Point Patterns

The theory of adaptive recursive tessellations is given in [37] with applications of their use for spatial analysis in [38] and with more specific application to point pattern analysis in [39]. At the heart of adaptive recursive tessellations is a variable resolution approach to space. No longer are scale and resolution treated as being uniform across an area but are allowed to vary spatially in response to the point pattern. This is

achieved through a recursive decomposition of space, similar to quadtrees, but allowing variable decomposition ratios and rectangular cells. The algorithm makes no prior assumptions about the statistical or spatial distribution of points. Each point is treated as a binary occurrence of some phenomenon without further descriptive attributes. The decision to further decompose any one cell larger than the atomic cell size is based on the variance at the next level of decomposition and a heuristic on the number of empty cells that result. The atomic cell size is mediated between the average nearest neighbour distance between points and average area per point expressed as a cell size. Tests have shown the algorithm to be consistently effective in comparison with other approaches of point cluster detection [39]. The resulting clusters, or hotspots, are termed Geo-ProZones (GPZ) as they represent zones of geographical proximity in the point pattern (see example in the next section).

An at-risk population is not used in the initial identification of GPZ clusters. As pointed out above, in many data mining applications this may not be possible at the outset without serious risk of mis-specification. GAM-type and epidemiological approaches are dismissive of identifying clusters without reference to an at-risk population or some control population. Count data on their own do, however, reflect workload, revenue stream or commitment of resources in meeting a spatially distributed demand as represented by a pattern of point occurrences. GPZ clusters provide just such a picture. Once that picture and its attribute dimensions are clearly understood then a second stage analysis of risk can be carried out, if necessary, for the application at hand.

The method being proposed in this paper is to use GPZ cluster centroids as a guide to setting up and running the *k*-means clustering. GPZ clusters are only concerned with the spatial distribution of binary events typical of a hotspot approach. The *k*-means clustering can be used to introduce the other descriptive attributes of the points to derive a spatial segmentation which incorporates all points. A risk-based cluster analysis can then be carried out within each segment in relation to the at-risk population appropriate to that segment in order to identify statistically significant clusters. This then brings together two broad approaches to cluster detection in spatial data mining discussed above.

4. Geo-ProZones and *k*-means Clustering in Tandem: A Case Study

The example shown here is a business transactions database for one year. Customers are geocoded by address and the attribute SPEND (total spent by each customer in the year) is singled out for analysis. Through the postcode, each customer was assigned to one of ten GB-Profiles lifestyle categories [40]. These lifestyle categories are given in Table 1. The business operates from a single hub in a provincial town in England and aims to serve a regional market rather than a national one.

1	**Struggling**; Multi-ethnic Areas - Pensioners and single parents - high unemployment - Local Authority rented flats
2	**Struggling**; Council Tenants - Blue collar families and single parents – Local Authority rented terraces
3	**Struggling**; Less Prosperous Pensioner Areas - Retired blue collar residents – Local Authority rented semi-detached
4	**Struggling**; Multi-ethnic Areas; Less Prosperous Private Renters - Young blue collar families with children - privately renting terraces and bedsits
5	**Aspiring**; Academic Centres & Student Areas - Young educated white collar singles and couples - privately rented bedsits and flats
6	**Aspiring**; Young Married Suburbia - Young well-off blue collar couples and families - mixed tenure terraces
7	**Climbing**; Well-Off Suburban Areas - Young white collar couples and families - buying semi-detached and detached
8	**Established**; Rural Farming Communities - Mature well-off self-employed couples and pensioners - owning or privately renting large detached houses
9	**Prospering**; Affluent Achievers - Mature educated professional families - owning and buying large detached
10	**Established**; Comfortable Middle Agers - Mature white collar couples and families - owning and buying semi-detached

Table 1: GB-Profiles classification in 10 classes

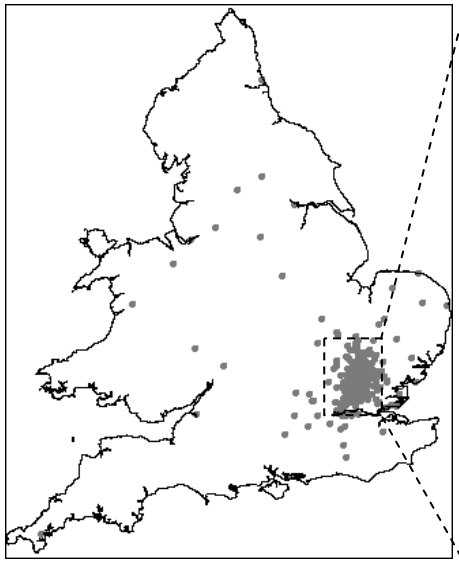


Figure 1: Distribution of customers



Figure2: Analysed customers with hub

On first plotting the distribution of points (Figure 1) it became evident that a small proportion of customers were from outside the region and had used the business opportunistically whilst temporarily staying in the area or passing through. Since the 'region' is not spatially defined, these were pruned on the basis of being outliers when the distance to the hub of all customers is calculated and normalised. The distribution of customers thus used in the analysis is given in Figure 2. For databases with undefined spatial boundaries (i.e. not specifically linked to any boundary system), a convex hull is first fitted to the data. GPZ clustering is then carried out. The results are given in Figures 3 and 4.

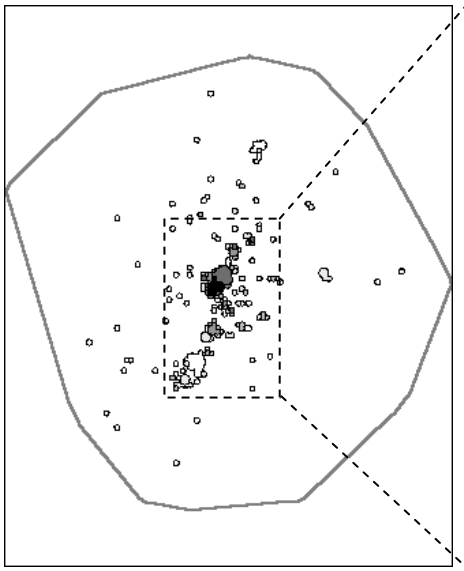


Figure 3: Convex hull and GPZ clusters

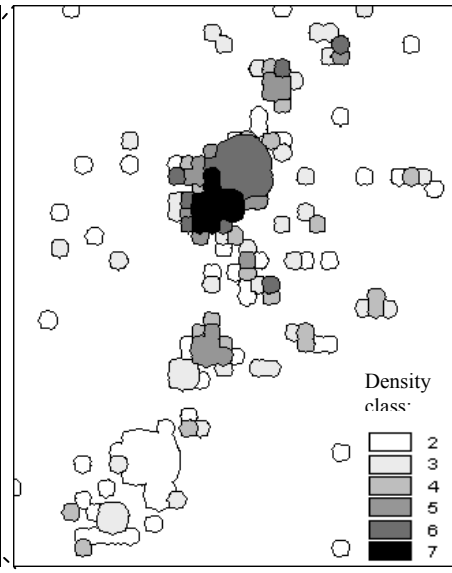


Figure 4: Detail of GPZ clusters

The GPZ cluster pattern shows a trend from south east to north west through the hub. This pattern conforms with the direction of the main road (motorway) and railway links in the area. On further inspection of this pattern, eight cluster centres (mostly highest density) were selected as the seed for the *k*-means clustering (Figure 5). Distances from each of these centroids to all the customers were calculated. These distance fields and SPEND were then normalised using the technique of *robust normalisation* [41] [42]. Robust normalisation produces a distribution of median 0, lower quartile of –1 and upper quartile of +1 and is not sensitive to long tails as would a z-score normalisation. The k-means clustering was first run on just the distance measures without attributes (Figure 6) and then using the SPEND and GB-Profile attributes (Figure 7). The characteristics of each of the resulting clusters are summarised in Table 2.

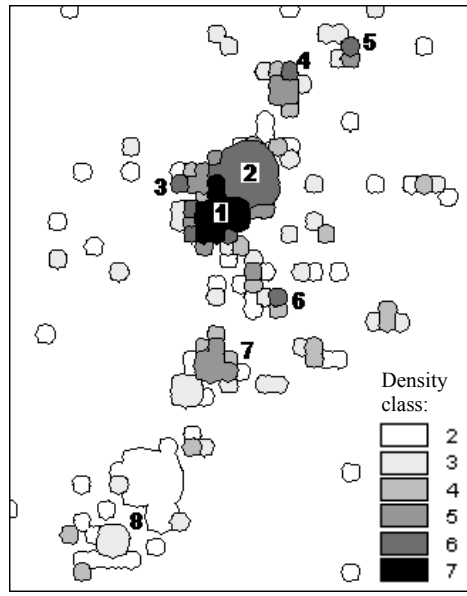


Figure 5: Location of k=8 centroids.

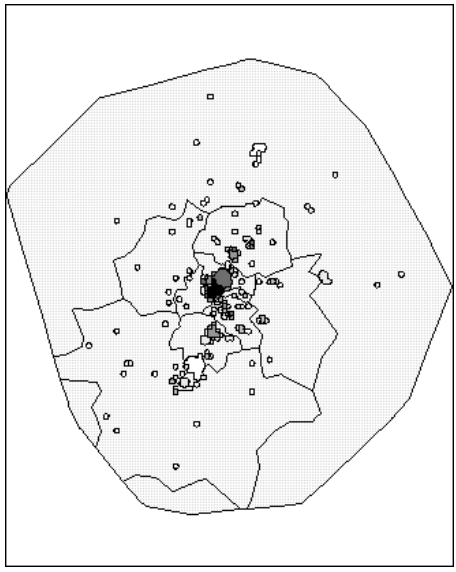


Figure 6: k-means, distance only.

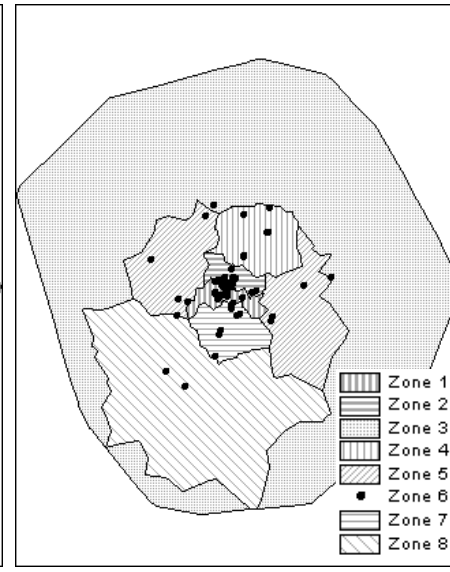


Figure 7: k-means, all attributes.

The k-means clusters in Figure 6 are shown in relation to the GPZ clusters. Since no other attributes were used (i.e. treated as binary point events), this clustering

is purely spatial and has resulted in a mutually exclusive spatial segmentation. Centroids 4 and 5 (Figure 5) have been combined in favour of a zone around the periphery representing sparsely dispersed customers that are relatively far from the hub. The *k*-means clustering which includes all attributes (Figure 7) produced an unexpected result – one cluster (Zone 6) that is not spatially mutually exclusive with the others. In order for this additional cluster to emerge, two adjacent clusters were 'merged' (Zone 7).

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8
n	602	469	178	185	107	70	218	195
Med. Dist	1164	1082	17100	5230	7228	1764	5929	13640
Tri. Spend	153	213	249	208	209	2884	226	247
GB-Prof1								2%
GB-Prof2			2%			1%		30%
GB-Prof3	8%	5%	4%	5%	2%	4%	14%	7%
GB-Prof5	2%	2%	3%				3%	3%
GB-Prof6	6%	4%	3%	6%		3%	5%	18%
GB-Prof7	47%	17%	24%	52%	7%	36%	9%	12%
GB-Prof8	7%	32%	25%	12%	36%	21%	28%	6%
GB-Prof9	15%	27%	28%	24%	52%	29%	35%	10%
GB-Prof10	15%	13%	10%		4%	6%	6%	12%

Table 2: Characteristics of the 8 clusters. Med. = median, Tri. = trimean. *(note: class GB-Prof4 is not present in the study area)*

From inspection of Table 2, the eight *k* clusters (Z1-Z8) have quite different characteristics, either spatially and/or in their attributes – median distance (in metres), trimean of SPEND and the GB-Profile classes. The trimean is defined as [43]:

$$\text{(lower quartile + 2.median + upper quartile) / 4} \qquad (1)$$

Although initially from Figure 5, centroids 1 and 2 may be viewed as forming the same cluster, their characteristics have emerged as being quite different. Z1 (Zone 1 in Figure 7) has the lowest trimean spend and is dominated by geodemographic class 7 (*climbing*) in contrast to spatially neighbouring Z2 which has higher trimean spend and a modal geodemographic class 8 (*prospering*). Cluster Z5 (Zone 5 in Figure 7) is spatially split on either side of the central clusters immediately around the hub. It nevertheless has a distinctive geo-demographic make-up being dominated by *prospering* and *established* households though not the greatest spenders. On the other hand Z8, which is comparatively far from the Hub and has 39% in the three *struggling* classes are the third highest spending. But probably of most interest from a business perspective is Z6, the cluster that spatially overlaps with

the others. The customers from this cluster are the highest spenders, an order of magnitude above the others. They come from the *aspiring*, *established* and *prospering* geodemographic classes. Inspection of Figure 7 suggests a spatial hotspot within this cluster that can be further analysed into GPZ clusters and tested for significance against the specific geodemographic groups. What is substantive is that such a small cluster (less than 5% of the customers) can be separated out from the larger database through the mining process. If a mutually exclusive spatial segmentation were to have been carried out, these customers would have been outliers within their respective spatial segments and it is therefore likely that they would have introduced some bias into the analysis.

5. Conclusions

This paper has demonstrated a dual approach to the spatial data mining of point data. A combination of hotspot-style clustering of point events to identify *k* number candidate centroids followed by *k*-means clustering incorporating all attributes assigns all points to clusters most of which are spatially mutually exclusive without precluding the emergence of distinct attribute-based clusters that overlap with the strongly spatial clusters. A follow-on stage of the analysis is to do again the hotspot-style clustering within each of the *k*-means clusters to establish those that are significant against the appropriate at-risk population. Although the example used here has been a relatively small data set, it has allowed the workings of the technique to be explained and visualised. The variable resolution approach to producing GPZ clusters has shown itself to be effective. The overall approach can be used on very large databases as no prior assumptions are required.

References

1. Miller, H. J. and Han, J. (2001) *Geographic Data Mining and Knowledge Discovery*. Taylor & Francis, London.
2. Macmillan, W. (1998) Epilogue. In Longley *et al*. (eds) *Geocomputation: A Primer*. Chichester: Wiley: 257-264
3. Fotheringham, A. S. (1992) Exploratory spatial data analysis and GIS. *Environment and Planning A* 24: 1675-1678
4. Unwin, D. (1996) GIS, spatial analysis and spatial statistics. *Progress in Human Geography* 20: 540-551
5. Snow, J. (1855) *On the Mode of Communication of Cholera*. Churchill Livingstone, London.
6. Clark, P. J. and Evans, F. C. (1954) Distance to nearest neighbour as a measure of spatial relations in populations. *Ecology* 35: 445-453
7. Knox, E. G. (1964) The detection of space-time interactions. *Applied Statistics* 13: 25-29
8. Harvey, D. W. (1966) Geographical processes and point patterns: testing models of diffusion by quadrat sampling. *Transactions of the Institute of British Geographers* 40: 81-95
9. Mantel, M. (1967) The detection of disease clustering and a generalised regression approach. *Cancer Research* 27: 209-220

10. Cliff, A. D. and Ord, J. K. (1981) *Spatial Processes: Models and Applications*. Pion, London.
11. Couclelis, H. (1998) Computation and space. *Environment & Planning B*, 25th Anniversary Issue: 41-47
12. Fotheringham, A. S. (1998) Trends in quantitative methods II: Stressing the computational. *Progress in Human Geography* 22: 283-292
13. Longley, P. A.; Brooks, S. M.; McDonnell, R. & MacMillan, B. (1998). *Geocomputation: A Primer*. Chichester: Wiley.
14. Armstrong, M. P. (2000) Geography and computational science. *Annals of the Association of American Geographers* 90: 146-156
15. Openshaw, S. and Abrahart, R. J. (2000) *GeoComputation.* Taylor & Francis, London.
16. Brimicombe, A. J. (2002) GIS: where are the frontiers now? *Proceedings GIS 2002*, Bahrain: 33-45
17. Fotheringham, A. S. (1997) Trends in quantitative methods I: Stressing the local. *Progress in Human Geography* 21: 88-96
18. Fotheringham, A. S. and Brunsdon, C. (1999) Local forms of spatial analysis. *Geographical Analysis* 31: 340-358
19. Fotheringham, A. S.; Brunsdon, C. and Charlton, M. (2000) *Quantitative Geography*. Sage, London.
20. Phillips, J. D. (1999) Spatial analysis in physical geography and challenge of deterministic uncertainty. *Geographical Analysis* 31: 359-372
21. Murray, A. T. and Estivill-Castro, V. (1998) Cluster discovery techniques for exploratory spatial data analysis. *International Journal of Geographical Information Science* 12: 431-443
22. Openshaw, S. (1998) Building automated geographical analysis and explanation machines. In Longley *et al*. (eds) *Geocomputation: A Primer*. Chichester: Wiley: 95-115
23. Murray, A. T. (2000) Spatial characteristics and comparisons of interaction and median clustering models. *Geographical Analysis* 32: 1-18
24. Halls, P.J.; Bulling, M.; White, P. C. L.; Garland, L. and Harris S. (2001) Dirichlet neighbours: revisiting Dirichlet tessellation for neighbourhood analysis. *Computers, Environment and Urban Systems* 25: 105-117
25. Kiang, M. Y. (2001) Extending the Kohonen self-organizing map networks for clustering analysis. *Computational Statistics & Data Analysis* 38: 161-180
26. Estivill-Castro, V. and Lee, I. (2002) Argument free clustering for large spatial point-data sets via boundary extraction from Delaunay Diagram. *Computers, Environment and Urban Systems* 26: 315-334
27. Sokal, R. and Sneath, P. (1963) *Principles of Numerical Taxonomy*. Freeman, San Francisco.
28. Aldenderfer, M. S. and Blashfield, R. K. (1984) *Cluster Analysis.* Sage, California.
29. Han J.; Kamber, M. and Tung, A. (2001) Spatial clustering methods in data mining. In Miller & Han (eds.) *Geographic Data Mining and Knowledge Discovery*. Taylor & Francis, London: 188-217
30. MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Maths and Statistics Problems* Vol1: 281-297
31. Openshaw, S.; Charlton, M. E.; Wymer, C. and Craft, A. W. (1987) A mark I geographical analysis machine for the automated analysis of point data sets. *International Journal of Geographical Information Systems* 1: 359-377
32. Openshaw, S. (1994) Two exploratory space-time attribute pattern analysers relevant to GIS. In Fotheringham & Rogerson (eds.) *Spatial Analysis and GIS*. Taylor & Francis, London: 83-104
33. Rowlingson, B. S. and Diggle, P. J. (1993) Splancs: spatial point pattern analysis code in S-Plus. *Computers and Geosciences* 19: 627-655

34. Gatrell, A. C. and Rowlingson, B. S. (1994) Spatial point process modelling in a geographical information system environment. In Fotheringham & Rogerson (eds.) *Spatial Analysis and GIS*. Taylor & Francis, London: 147-163
35. Gatrell, A. C.; Bailey, T. C.; Diggle, P. J. and Rowlingson, B. S. (1996) Spatial point pattern analysis and its application in geographical epidemiology. *Transactions of the Institute of British Geographers* NS 21: 256-274
36. Lawson, A. B. (2001) *Statistical Methods in Spatial Epidemiology*. John Wiley & Sons, Chichester.
37. Tsui, H. Y. and Brimicombe, A. J. (1997a) Adaptive recursive tessellations (ART) for Geographical Information Systems. *International Journal of Geographical Information Science* 11: 247-263
38. Tsui, H. Y. and Brimicombe, A. J. (1997b) Hierarchical tessellations model and its use in spatial analysis. *Transactions in GIS* 2: 267-279
39. Brimicombe, A. J. and Tsui H. Y. (2000) A variable resolution, geocomputational approach to the analysis of point patterns. *Hydrological Processes* 14: 2143-2155
40. Openshaw, S. and Blake, M. (1996) *GB Profiler 91*. Department of Geography, University of Leeds.
41. Brimicombe, A. J. (1999) Small may be beautiful - but is simple sufficient?" *Geographical and Environmental Modelling* 3: 9-33
42. Brimicombe, A. J. (2000) Constructing and evaluating contextual indices using GIS: a case of primary school performance" *Environment & Planning A* 32: 1909-1933
43. Tukey, J.W. (1977) *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

PGIS, a Software for Paleogeographic Reconstruction in ArcGIS

Shilong Mei

Alberta geological Survey, Edmonton, Alberta, Canada T6B 2X3
shilong.mei@gov.ab.ca

Abstract. Geologists need to restore the past locations of interested features. Reconstruction of geological features to their past locations involves moving a feature around the Earth surface. Current GIS software packages do not have the function to restore the past locations of geological features, but do have the potential to realize it. Paleogeographic Information System (PGIS) is an approach to incorporate the Paleocontinent Reconstruction function into GIS. This approach has been realized in ArcGIS by developing a Paleogeographic Reconstruction functionality using ArcObjects and Visual Basic for Application (VBA) embedded in ArcMap.

1 Introduction

Current GIS tools and GIS applications deal with localities and areas that have fixed present day locations in a selected coordinate system (ESRI, 1997). They are static in terms of spatial location. Geologists, however, deal with an Earth history of 4.6 billion years, during which continents are believed to have moved around the Earth's surface. This means that any present day locality changed its spatial location during geological history. To reveal the true spatial pattern of geological features such as mineralization and fossil distribution of the past, these features need to be dynamically restored to their past positions (for example, see Scotese, 1997; Ziegler et al., 1997; Mei and Henderson, 2001). However, this functionality has not been incorporated to the present day GIS packages.

Since the advent of the plate tectonic paradigm in the 1960s, palaeogeographic reconstructions have become an integral part of global tectonic research. Bullard et al. (1965) first used the computer to carry out the fitting of South America and Africa. Smith and Hallam (1970) used the computer to quantify the fitting of Gondwanaland that was made manually by Du Toit (1937). Since then, Paleocontinent Reconstruction together with Plate Tectonics became a hot topic, and numerous Paleocontinent Reconstruction maps were generated (Morel and Irving, 1978; Scotese et al., 1979; Scotese and Mckerrow, 1990; Smith et al., 1981; Piper, 1987; Dalzie, 1991; Moores, 1991; Unrug, 1993).

Reconstruction of past locations mainly involves moving or rotating paleocontinents or paleo-plates on the Earth's surface. This movement can be defined by the Euler's theorem that was proved by L. Euler in 1776 (Cox and Hart, 1986). It is also called the fixed-point theorem and states that a layer on a rigid sphere can be moved

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 12-22, 2003.
© Springer-Verlag Berlin Heidelberg 2003

to any other orientation or position by a single rotation about a suitably chosen axis through the centre of the sphere. This rotation is referred to as the Euler rotation. In order to undertake global tectonic modeling, a few Paleocontinent Reconstruction software packages have been developed to carry out the Euler rotation. The most popular examples are Gmap (Torsvik and Smethurst, 1999), PGIS/Mac (Mac OS) version 4.0 and PLATE TRACKER (Windows 9x/NT) version 2.0 (see http://www.scotese.com/), ODSN Plate Tectonic Reconstruction Service (see http://www.odsn.de/odsn/services/ paleomap/paleomap.html), GLG-Plates Version 2 (http://www.glg.ed.ac.uk/people/ academic/cef/glgsoft/plates/plates.html), GUDALU (Li, 1993) and Paleo-Continental Map Editor (PCME) (Schettino, 1998, 1999; http://www.itis-molinari.mi.it/Intro-Reconstr .html). These software packages were developed in isolation from GIS. As a result, they are used mainly for generating Paleocontinent Reconstruction sketch maps and lack most of the functionalities of GIS in data management, visualization and spatial analysis. However, initial attempts have been made to combine the functionality of GIS with Paleocontinent Reconstruction software packages. Earth System History GIS (ESH-GIS v2.0, see http://www.scotese.com/software.htm) is a software package developed for converting paleocontinent reconstruction maps into the shapefile format so that they can be viewed with ArcView GIS or ArcView compatible software. GeoArctic Tectonic Modeller © is a separate custom module developed by GeoArctic to enable the import and reconstruction of both raster and vector data in ArcView (Whittaker et al., 2000). However, the application is confined to the North Atlantic Ocean and the function is still limited and not fully incorporated into GIS. Mei and Henderson (2002) proposed a Paleogeographic Information System (PGIS) approach and discussed the procedure, methodology and calculation for incorporating the Euler rotation function into a GIS package.

This paper introduces PGIS version 1, which is a software developed using Visual Basic for Applications (VBA) and ArcObjects. It has realized the Paleogeographic Reconstruction (including Paleocontinent Reconstruction) functionality in ArcGIS ArcMap. It enables geologists to use ArcGIS to model tectonic evolution, reconstruct and map past geological features and perform spatial analysis based on the past locations.

2 Algorithms for Paleogeographic Reconstruction Used by PGIS

The procedures for Paleogeographic Reconstruction are: 1. to obtain the latitudes and longitudes of both the present and the past locations for a sample point, or reference point, associated with a continent block; 2. to calculate the Euler rotation matrix based on the present and the past locations of the sample point; 3. to convert the coordinates of the present locations of continent blocks to the coordinates of their past locations using the rotation matrix. Once geological features have been restored to the past locations, they can be analyzed as a normal feature in a GIS package. The steps involved in a typical Paleogeographic Reconstruction are described as below.

2.1 Obtaining the Present and Past Locations of the Reference Point

The present and past latitudes, longitudes and azimuths of the reference point are usually provided by geologists as the present sample location, the past location and azimuth calculated through paleomagnetic analysis (McElhinny and McFadden, 2000). When the paleomagnetic data are not available, the reconstruction can be made on the screen by trial and test based on a geologist's experience. In this case, the centroid or label point of a continent block polygon can be used as the reference point, and an arbitrary azimuth can be used for test.

2.2 Calculating the Euler Rotation Matrix

To calculate the rotated coordinates for other points on the block, the rotation matrix needs to be calculated first. If **A** is a vector with global Cartesian coordinates (A_x, A_y, A_z) prior to rotation, then the components (A_x', A_y', A_z') of **A** after rotation to **A'** may be found from the matrix multiplication (after Cox and Hart, 1986):

$$\boldsymbol{A' = RA}$$

where **R** represents a 3 X 3 matrix and is referred to as the rotation matrix.

Writing all of the terms of the vector and matrix we have

$$\begin{bmatrix} A_x' \\ A_y' \\ A_z' \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}$$

Suppose that the Euler pole E (λ, ϕ) and the rotation angle Ω are already known, to define the elements of the rotation matrix **R**, we need to first calculate the Cartesian coordinates of the Euler pole on a globe. The conversion of the paleolatitude and paleolongitude of the Euler pole E (λ, ϕ) into the Cartesian coordinates E (E_x, E_y, E_z) can be calculated using the following equations (after Cox and Hart, 1986):

$$\begin{cases} E_x = r\cos\lambda\cos\phi \\ E_y = r\cos\lambda\sin\phi \\ E_z = r\sin\lambda \end{cases}$$

where r is the radius of the sphere.

The elements of the rotation matrix are then given by (after Cox and Hart, 1986):

$$R_{11} = E_xE_x(1 - \cos\Omega) + \cos\Omega$$
$$R_{12} = E_xE_y(1 - \cos\Omega) - E_z\sin\Omega$$
$$R_{13} = E_xE_z(1 - \cos\Omega) + E_y\sin\Omega$$
$$R_{21} = E_yE_x(1 - \cos\Omega) + E_z\sin\Omega$$
$$R_{22} = E_yE_y(1 - \cos\Omega) + \cos\Omega$$
$$R_{23} = E_yE_z(1 - \cos\Omega) - E_x\sin\Omega$$
$$R_{31} = E_zE_x(1 - \cos\Omega) - E_y\sin\Omega$$
$$R_{32} = E_zE_y(1 - \cos\Omega) + E_x\sin\Omega$$
$$R_{33} = E_zE_z(1 - \cos\Omega) + \cos\Omega$$

In Paleogeographic Reconstruction, the Euler pole and the past azimuth need to be obtained first before calculating the Euler rotation matrix. Geologists find the Euler pole and the past azimuth in several ways (Mei and Henderson, 2002). The following

shows how the Euler pole and the Euler rotation matrix are calculated when using the latitudes, longitudes and azimuths of both the present and past locations of a reference point as input, which represents a typical example of computational reconstruction process.

Suppose the past and present locations of a point on a block are known as P (λ, ϕ) and P_0 (λ_0, ϕ_0) respectively, and the rotation in the azimuthal orientation is known as α, then the reconstruction can be achieved by the following three steps (after Li, 1993; Mei and Henderson, 2002; see Figure 1):

First, rotate the block along the meridian ϕ_0 from P_0 (λ_0, ϕ_0) to P' (λ, ϕ_0); the Euler pole (E_1) and rotation angle (Ω_1) can be calculated as:

$$E_1 = (0, \phi_0 - 90^0)\,,\ \Omega_1 = \lambda - \lambda_0$$

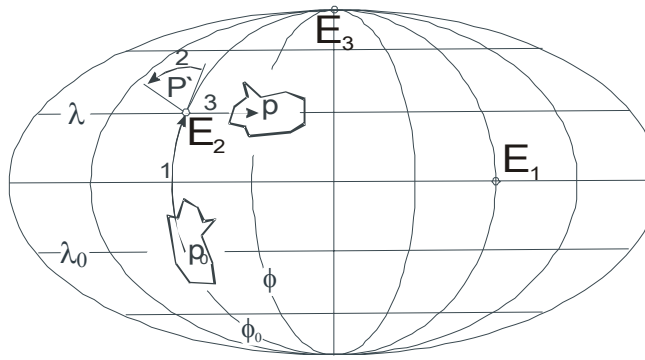


Figure 1. Showing the three steps for rotating the block from P_0 to P (after Li, 1993; Mei and Henderson, 2002)

Secondly, rotate the block at point P' (λ, ϕ_0) for an angle of α; the Euler pole (E_2) and rotation angle (Ω_2) can be calculated as:

$$E_2 = (\lambda, \phi_0)\,,\ \Omega_2 = \alpha$$

Thirdly, rotate the block along the latitude λ from P' (λ, ϕ_0) to P (λ, ϕ); the Euler pole (E_3), which is the north pole in this case, and the rotation angle (Ω_3) can be calculated as:

$$E_3 = (90^0, 0^0)\ ,\ \Omega_3 = \phi - \phi_0$$

In the example, there are three rotations involved. We name these three corresponding rotation matrixes as **R'**, **R''** and **R'''**. The rotation matrix for each of these three steps can be calculated as described previously. According to the Euler's theorem, any point of Cartesian coordinates (A_x, A_y, A_z) can always be rotated to coincide with another point A''' of Cartesian coordinates (A_x''', A_y''', A_z''') by just one direct rotation about a specially chosen axis. This single or direct rotation has the same effect as that obtained by successively applying the three rotations. The direct rotation matrix **R** can be acquired by using the associative property of matrix multiplication. This says that

R = R''' R''R',

$$\begin{bmatrix} A_x''' \\ A_y''' \\ A_z''' \end{bmatrix} = \begin{bmatrix} R'''_{11} & R'''_{12} & R'''_{13} \\ R'''_{21} & R'''_{22} & R'''_{23} \\ R'''_{31} & R'''_{32} & R'''_{33} \end{bmatrix} \left(\begin{bmatrix} R''_{11} & R''_{12} & R''_{13} \\ R''_{21} & R''_{22} & R''_{23} \\ R''_{31} & R''_{32} & R''_{33} \end{bmatrix} \left(\begin{bmatrix} R'_{11} & R'_{12} & R'_{13} \\ R'_{21} & R'_{22} & R'_{23} \\ R'_{31} & R'_{32} & R'_{33} \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \right) \right),$$

or $$\begin{bmatrix} A_x''' \\ A_y''' \\ A_z''' \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}.$$

Note that matrix multiplication does not commute, i.e., **R''' R''R' ≠ R' R'' R'''.** The elements of the direct rotation matrix can be found as follows (slightly modified after Cox and Hart, 1986).

Let

$$T = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = \begin{bmatrix} R'''_{11} & R'''_{12} & R'''_{13} \\ R'''_{21} & R'''_{22} & R'''_{23} \\ R'''_{31} & R'''_{32} & R'''_{33} \end{bmatrix} * \begin{bmatrix} R''_{11} & R''_{12} & R''_{13} \\ R''_{21} & R''_{22} & R''_{23} \\ R''_{31} & R''_{32} & R''_{33} \end{bmatrix}$$

then,

$$\begin{aligned}
T_{11} &= R'''_{11}R''_{11} + R'''_{12}R''_{21} + R'''_{13}R''_{31} \\
T_{12} &= R'''_{11}R''_{12} + R'''_{12}R''_{22} + R'''_{13}R''_{32} \\
T_{13} &= R'''_{11}R''_{13} + R'''_{12}R''_{23} + R'''_{13}R''_{33} \\
T_{21} &= R'''_{21}R''_{11} + R'''_{22}R''_{21} + R'''_{23}R''_{31} \\
T_{22} &= R'''_{21}R''_{12} + R'''_{22}R''_{22} + R'''_{23}R''_{32} \\
T_{23} &= R'''_{21}R''_{13} + R'''_{22}R''_{23} + R'''_{23}R''_{33} \\
T_{31} &= R'''_{31}R''_{11} + R'''_{32}R''_{21} + R'''_{33}R''_{31} \\
T_{32} &= R'''_{31}R''_{12} + R'''_{32}R''_{22} + R'''_{33}R''_{32} \\
T_{33} &= R'''_{31}R''_{13} + R'''_{32}R''_{23} + R'''_{33}R''_{33}
\end{aligned}$$

Since

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = T * R' = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} * \begin{bmatrix} R'_{11} & R'_{12} & R'_{13} \\ R'_{21} & R'_{22} & R'_{23} \\ R'_{31} & R'_{32} & R'_{33} \end{bmatrix}$$

then, we get

$$\begin{aligned}
R_{11} &= T_{11}R'_{11} + T_{12}R'_{21} + T_{13}R'_{31} \\
R_{12} &= T_{11}R'_{12} + T_{12}R'_{22} + T_{13}R'_{32} \\
R_{13} &= T_{11}R'_{13} + T_{12}R'_{23} + T_{13}R'_{33} \\
R_{21} &= T_{21}R'_{11} + T_{22}R'_{21} + T_{23}R'_{31} \\
R_{22} &= T_{21}R'_{12} + T_{22}R'_{22} + T_{23}R'_{32} \\
R_{23} &= T_{21}R'_{13} + T_{22}R'_{23} + T_{23}R'_{33} \\
R_{31} &= T_{31}R'_{11} + T_{32}R'_{21} + T_{33}R'_{31} \\
R_{32} &= T_{31}R'_{12} + T_{32}R'_{22} + T_{33}R'_{32} \\
R_{33} &= T_{31}R'_{13} + T_{32}R'_{23} + T_{33}R'_{33}
\end{aligned}$$

Now that we have found the direct rotation matrix, the direct Euler pole E (λ_E, ϕ_E) and the direct rotation angle Ω can also be found as follows (after Cox and Hart, 1986):

$$\phi_E = \tan^{-1}\left(\frac{R_{13} - R_{31}}{R_{32} - R_{23}}\right)$$

$$\lambda_E = \sin^{-1}\left(\frac{R_{21} - R_{12}}{\sqrt{(R_{32} - R_{23})^2 + (R_{13} - R_{31})^2 + (R_{21} - R_{12})^2}}\right)$$

$$\Omega = \tan^{-1}\left(\frac{\sqrt{(R_{32} - R_{23})^2 + (R_{13} - R_{31})^2 + (R_{21} - R_{12})^2}}{R_{11} + R_{22} + R_{33} - 1}\right)$$

where $0^0 \le \Omega \le 180^0$.

The Arctangent function returns a value from –90 to 90, which must be converted to the desired range.

2.3 Calculating the Coordinates of the Past Locations

First, get the global Cartesian coordinates of the present locations from the present day latitudes (λ) and longitudes (ϕ) using the following equations (after Cox and Hart, 1986):

$$\begin{cases} A_x = r\cos\lambda\cos\phi \\ A_y = r\cos\lambda\sin\phi \\ A_z = r\sin\lambda \end{cases}$$

Then, applying the usual rules for matrix multiplication gives the equations for calculation of the rotated coordinates using the direct rotation matrix (after Cox and Hart, 1986):

$$A_x' = R_{11}A_x + R_{12}A_y + R_{13}A_z$$
$$A_y' = R_{21}A_x + R_{22}A_y + R_{23}A_z$$
$$A_z' = R_{31}A_x + R_{32}A_y + R_{33}A_z$$

These rotated coordinates (A_x', A_y', A_z') are then converted back to the latitude and longitude on the auxiliary sphere (λ', ϕ'). This step is necessary for the map projection (Snyder, 1987; Yang *et al.*, 2000).

$$\begin{cases} \lambda' = \sin^{-1}(A_z'/R) \\ \phi' = \tan^{-1}(A_y'/A_x') \\ r = \sqrt{A_x'^2 + A_y'^2 + A_z'^2} \end{cases}$$

The planar x, y coordinates of the past locations can then be obtained through map projection and then used to reconstruct the rotated features in ArcGIS.

3 Paleogeographic Reconstruction Functionality of PGIS and Its GUI

The PGIS approach, which incorporates the Paleogeographic Reconstruction function into the present day GIS, has been accomplished in ArcGIS ArcMap using Ar-

cObjects and Visual Basic for Applications (VBA). The PGIS's GUI developed is represented by a toolbar and related forms (Figure 2).

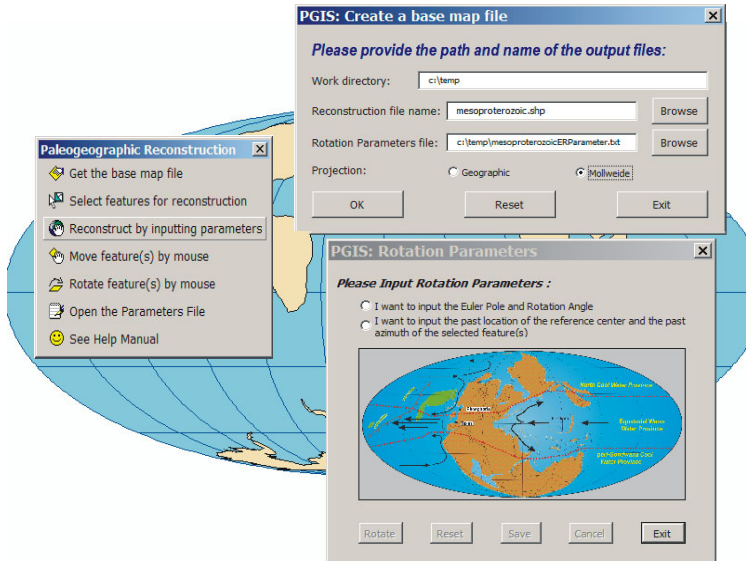


Figure 2. Showing the main graphic user interface of the Paleogeographic Reconstruction functionality

The tools on the tool bar are arranged according to the typical workflow of Paleogeographic Reconstruction. The <Get the base map layer> tool allows the user to open an existing base map file or compile a new base map file using the paleocontinents that have already been digitized (Figure 3). It also allows the user to select or create a text file for recording the Euler rotation parameters used for reconstructing each feature. The base map file contains all the paleocontinents that need to be reconstructed, and it is also used as the output file for storing the reconstructed map. The <Select features> tool is one of the built-in selection tools of ArcMap. It is used for selecting features for reconstruction. The <Reconstruct using parameters> tool allows the user to carry out reconstruction by inputting parameters for calculating the Euler rotation. It opens the Rotation Parameters Form that allows the user to carry out reconstruction by inputting parameters such as the coordinates of the Euler pole and the rotation angle, or the past coordinates of the reference center and the past azimuth. The present location of the reference center is set as the middle point of selected features or the selection anchor point. The Euler rotation matrix needed for reconstruction is calculated from the Euler pole and the rotation angle, or the past and the present coordinates and the past azimuth of the reference center. The user can also choose to input the coordinates in decimal, or in radian. The <Move feature(s) by mouse> and <Rotate feature(s) by mouse> tools allow the user to micro-adjust the reconstruction by parameters through moving or rotating selected feature(s) by pressing down the left mouse button and drag the features. They are similar to those of ArcMap Editor except that features moved or rotated change shape following projection. The Open the parameters file tool opens the text file with records of the Euler

pole and rotation angle used to rotate each feature. More detailed explanation of the functionality and its GUI are provided in the Help manual.

Presently, the customization for the Paleogeographic Reconstruction functionality is
delivered as an ArcMap document and an ArcMap template. This software can be obtained from the present author (Shilong Mei). It can be further developed as an additional module or extension to ArcGIS in the future. The addition of Paleogeographic Reconstruction functionality in ArcGIS makes the location of features changeable and the ArcGIS dynamic.

4 An Example

An example of reconstruction of the world during late Mesoproterozoic (1 billion years ago) is presented here to further demonstrate the Paleogeographic reconstruction functionality developed. Figure 3 shows the paleocontinent blocks used for reconstruction in the developed Paleogeographic Reconstruction functionality. The teconoic subdivisions and coding system used for these blocks are after Wang *et al.* (1997).

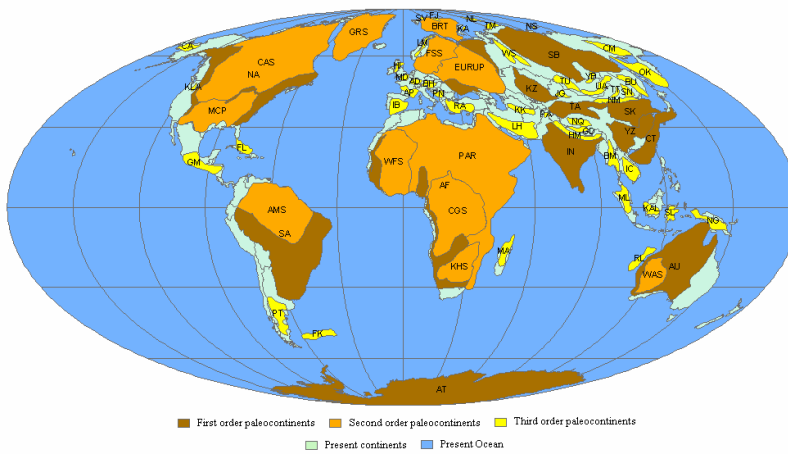


Figure 3. Showing the Paleocontinent blocks against the present day world map (for coding see Appedix B). The Paleocontinent outlines are simplified from Wang *et al.* (1997)

The example of reconstruction consists of two steps. The first step is to reassemble the paleocontinents that were originally merged together forming a single block (see Figure 4). The Euler rotation parameters used are from Smith and Hallam (1970) and Li (1993).

The second step is to restore the past locations of the reassembled blocks as well as other paleocontinent blocks (Figure 5). The Euler rotation parameters used for the reconstruction are from Li (1993), which is also recorded in a text file with records of the Euler rotation parameters created by the developed software.

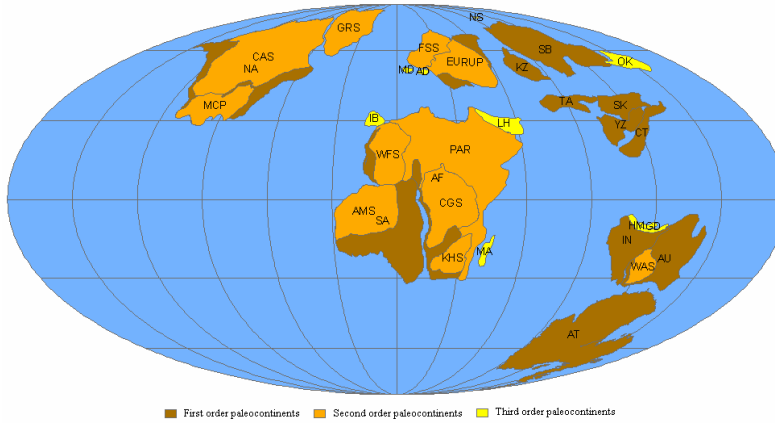


Figure 4. Showing the reassembled Paleocontinent blocks

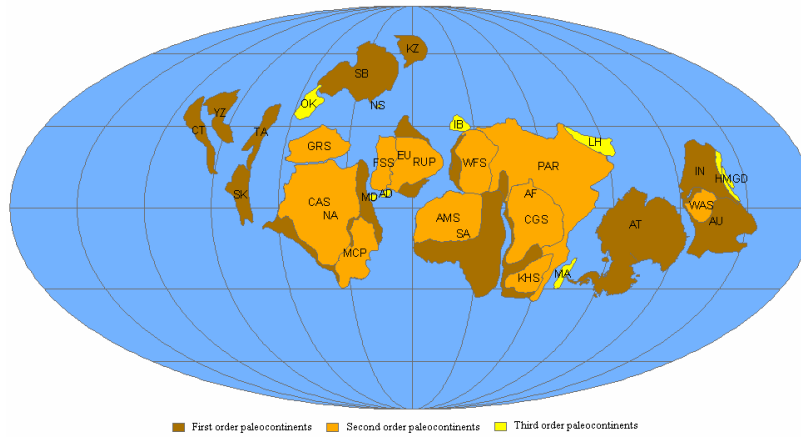


Figure 5. Paleocontinent reconstruction map of Meosoproterozoic

5 Discussion and Conclusion

Plate tectonics has become the fundamental theory in geology, and Paleogeographic Reconstruction is a computational modeling of plate tectonics and geological evolution. Incorporating the Paleogeographic Reconstruction function into the present day GIS makes it possible to restore the past locations of features in ArcGIS. It enables geologists to use ArcGIS to model tectonic evolution, reconstruct and map past geological features and perform spatial analysis based on the past locations. As a result, it provides geologists with the full access to the powerful GIS functionality in geological data management, analysis and presentation.

Although the functionality of Paleogeographic Reconstruction has been added into ArcGIS ArcMap, it needs to be pointed out that some issues remain to be solved. One of them concerning topology. Topology is one of the fundamental foundations for spatial analysis in the main flow GIS software such as the old ArcInfo (ESRI, 1997). Once feature locations have been digitized and the topology has been established, they remain unchanged in a present day GIS application. However, all the features may change shape and location in Paleogeographic Reconstruction. They may also be overlapping each other in some cases due to low accuracy and ambiguity of geological data. Topology may need to be created after reconstruction for the purpose of spatial analysis.

Other issues concern the error associated with the reconstruction of the past locations. One of the major error sources may be the estimation of the past location by paleomagnetism (McElhinny and McFadden, 2000). One another major source is the calculation of the Euler rotation. The Euler rotation can be calculated only on a datum of sphere. As a result, the original map projection has to be converted into a projection with a sphere datum (in this case, GCS_Sphere). After the calculation, the projection needs to be converted back to the original map projection. These transformations introduce errors, and the amount of error has not been evaluated. In addition, map projection itself also introduces errors. For example, Yang et al. (2000) showed that the accuracy level for an inverse transformation can be reached at 0.0001'. However, these errors are minor compared to the magnitude of the errors introduced in estimating the past locations by geologists. As a result, the reconstruction map generated by this software can be used safely as long as the resolution for estimating the past locations is acceptable to geologists.

References

1. Bullard, E.C., Everett, J.E., Smith, A.G.: The fit of the continents around the Atlantic: a symposium on continental drift. Phil. Trans. R. Soc. London, Ser. A (1965) 258, 41-51
3. Cox, A., Hart, R.B.: Plate Tectonics: How it Works. Blackwell Scientific Publications, Palo Alto, California (1986)
4. Dalzie, I.W.D.: Pacific margins of Laurentia and East_Antactica-Australia as a conjugate rift-pair: Evidence and implications for an Eocambrian supercontinent. Geology **19** (1991) 598-601
5. Du Toit, A.L.: Our Wandering Continents. Oliver & Boyd, Edimburgo (1937)
6. Environmental Systems Research Institute (ESRI), Inc.: Understanding GIS: The ARC/ INFO® Method. Environmental Systems Rerearch Institute, Inc., New York (1997)
7 .http://www.glg.ed.ac.uk/people/academic/cef/glgsoft/plates/plates.html
8. http://www.itis-molinari.mi.it/Intro-Reconstr.html
9. http://www.odsn.de/odsn/services/paleomap/paleomap.html
10. http://www.scotese.com/
11. Li, X.: Paleocontinental Reconstruction: methodology, computer software and an example of application. Journal of China University of Geosciences **4**(1) (1993) 7-13
12. McElhinny, M.W., McFadden, P.L.: Paleomagnetism, continents and oceans. Florida: Academic Press (2000)

13. Mei, S., Henderson, C.M.: Evolution of Permian conodont provincialism and its significance in global correlation and paleoclimate implication. Palaeogeography, Palaeoclimatology, Palaeoecology **170**(3-4) (2001) 237-260
14. Mei, S., Henderson, C.M.: Paleogeographic information system (PGIS): A potential dynamic GIS approach in geology. In: Hills, L.V., Henderson, C.M., and Bamber, E.W. (eds.), Carboniferous and Permian of the World. Canadian Society of Petroleum Geologists, Memoir **19** (2002) 924-933
15. Moores, E.M.: Southwest U.S. – East Antactic (SWEAT) connection, a Hypothesis. Geology, **19** (1991) 598-601
16. Morel, P., Irving, E.C.: Tentative paleocontinental maps for the Early Phanerozoic and Protozoic. Journal of Geology **86**(5) (1978) 535-561
17. Piper, J.D.A.: Paleomagnetism and continental crust. Open University Press, Milton Keynes (1987)
18. Schettino, A.: Computer aided paleogeographic reconstructions. Computers & Geosciences **24**(3) (1998) 259-267
19. Schettino, A.: Computational methods for calculating geometric parameters of tectonic plates. Computers & Geosciences **25**(8) (1999) 897-907
20. Scotese, C.R.: Paleogeographic Atlas, PALEOMAP Project Progress Report No. 90-0497. Department of Geology, University of Texas at Arlington, Arlington (1997)
21. Scotese, C. R., Bambach, R. K., Barton, C., van der Loo y R., Ziegler, A. M.: Paleozoic Base Maps. Journal of Geology **87** (3) (1979) 217-277
22. Scotese, C.R., Mckerrow, W.S.: Revised world maps and introduction. In: McKerrow, W.S., Scotese, C.R. (eds.), Paleozoic paleogeography and biogeography. Geological society of London Memoir **12** (1990) 1-12
23. Smith, A.G., Hallam, A.: The fit of the southern continents. Nature **225** (1970) 1328-1333.
24. Smith, A.G., Hurley, A.M., Briden, J.C.: Phanerozoic Paleocontinental World Maps. Cambridge: Cambridge University Press, New York (1981)
25. Snyder, J.P.: Map projections: a working manual. US Geological Survey Professional paper 1395 (1987)
26. Torsvik, T.H., Smethurst, M.A.: Plate tectonic modelling: virtual reality with GMAP. Computers & Geosciences **25** (1999) 395-402
27. Unrug, R.: The Gondwana supercontinent: Middle Proterozoic crustal fragments, Late Proterozoic assembly, and unsolved problems. In: Findley, R.H., Unrug, R., Banks, M.R., and Veevers, J.J. (eds.), Gondwana Eight. Netherlands: A.A. Balkema (1993) 3-8.
28. Wang Hongzhen, Li Xiang, Mei, Shilong and Zhang Shihong: Pangaea Cycles, Earth's Rhythms and Possible Earth Expansion. in: Wang Hongzhen, Borming Jahn and Mei, Shilong (eds.), Proceedings of the 30th International Geological Congress, Volume 1-Origin and History of the Earth, p. 111-128. VSP, Utrecht, the Netherlands, Tokyo, Japan (1997)
29. Whittaker, R.C., Karpuz R., Wheeler W., Ady B.E.: 4D Regional Tectonic Modeling: Plate Reconstructions Using a GIS. Published at http://www.geoarctic.com/papers&pres.php (2000)
30. Yang Q., Snyder J.P., Tobler W.R.: Map projection transformation principles and applications. London: Taylor & Francis (2000)
31. Ziegler, A.M., Hulver, M.L., Rowley, D.B.: Permian World Topography and Climate. In: Martini, I.P. (Ed), Late Glacial and Postglacial Environmental Changes—Quaternary, Carboniferous-Permian and roterozoic. Oxford University Press, New York (1997) 111-146

Spatial and Temporal Autocorrelation in Innovation Diffusion Analysis

Stefania Bertazzon

Department of Geography, University of Calgary,
2500 University Dr. NW, T2N 1N4Calgary, AB, Canada,
bertazzs@ucalgary.ca

Abstract. The paper discusses the effects of spatial, temporal, and joint spatio-temporal autocorrelation. The purpose of the study of spatio-temporal autocorrelation is to optimize the efficiency of dynamic regression estimates, in a spatio-temporal univariate regression model. After calculating and examining the effects of global indicators of spatial and temporal autocorrelation, a new spatio-temporal global autocorrelation index is proposed. The index can be view as a preliminary proposal for a dynamic version of Moran's I.

1 Introduction

Scope of the paper is a quantitative assessment of the spatial and temporal autocorrelation among spatial units and across time periods for an empirical innovation diffusion process. Spatial and temporal autocorrelation are analyzed separately for each time period and spatial unit. The focus of the analysis is on evaluating the joint spatio-temporal autocorrelation, aimed at comprehending the specific contribution of spatial and temporal autocorrelation, at separating the two effects, and understanding their joint effect, i.e. additive, multiplicative, or other. The assessment of spatial, temporal, and spatio-temporal autocorrelation is viewed as a preliminary analysis to the specification of a dynamic spatial diffusion model which is autoregressive in space and time.

The paper discusses the theoretical implications and the computational form of a "hypermatrix" for the study of spatio-temporal autocorrelation. Computational applications in Splus scripts are used. To this end, time series are stored as attributes of the spatial units of interest in the ArcView database. The final outcome of the paper is the proposition and discussion of alternative forms of a spatio-temporal version of traditional spatial autocorrelation indices, i.e. Moran's I.

1.1 Background

Innovation diffusion is, by its nature, a dynamic process, occurring in space and time. Hagerstrand's (1973) pioneering work established the foundations for quantitative assessment and modeling of innovation diffusion processes, but the subsequent developments have been comparatively less fruitful, undoubtedly due, in large part, to the computational problems induced by the joint treatment of space and time (Rogers, 1995; Clark, 1984). Indeed the development of dynamic analysis and modeling in geography, to date, is a lacking area, compared to other fields within GIScience (Raper, 2000). It should finally be noted that the development of true spatio-temporal

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 23-32, 2003.
© Springer-Verlag Berlin Heidelberg 2003

analytical functions in GIS is, to date, a problematic area (Fotheringham et al., 2000; Egenhofer and Golledge, 1998).

Some spatio-temporal models, exemplified by STARIMA (Space-Time AutoRegressive Integrated Moving Average) (Pfeifer and Deutsch, 1980), have been developed by traditionally non-spatial disciplines, such as econometrics, and constitute extensions of temporal models to the spatial dimension, with the consequence that the role of space is subordinate, and the conceptual framework is less than appropriate to dialing with spatial issues. Of particular concern in this work is the lack of an appropriate treatment of spatial autocorrelation, an issue that in recent years is increasingly imposing itself to the attention of quantitative geographers (Longley et al., 1998; Griffith and Layne, 1999),

A spatial perspective on dynamic models is expressed in Anselin (1988), who discusses a spatial version of the more traditional SUR (Seemingly Unrelated Regression) model. In its spatial version, the SUR model contains a set of spatial regression equations, one for each time period: the resulting system analyses the spatial and temporal variation of the process, and the autocorrelation structure is examined in the error matrix, in order to provide an efficient GLS (Generalized Least Squares) estimator.

The following discussion focuses on assessing spatial and temporal autocorrelation among spatial units in each time period and across the temporal sample. Such assessment constitutes a preliminary step before the specification and estimation of an autoregressive model.

2 Methods

2.1 Case Study and Study Area

The analysis is conducted in ArcView GIS and Splus Spatial Statistics. The case study is the diffusion of ski-lifts in the Italian Eastern Alps in the second half of the 20th century (Fig. 1).

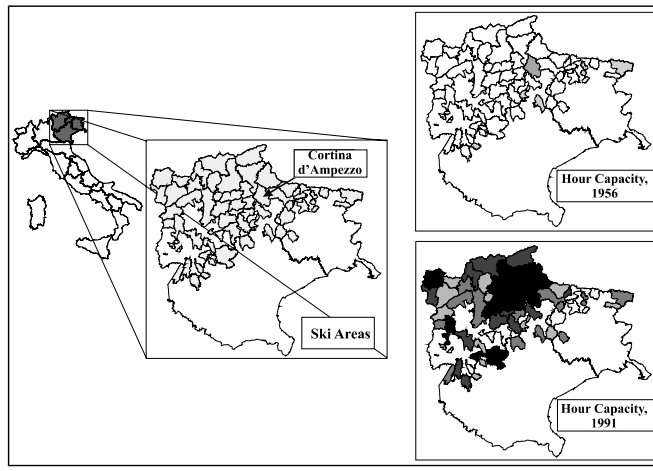


Fig. 1: Ski Areas and hour capacity (hc) in 1956

Ski-lifts can well be considered an innovation in the Italian Alps in the 1950s: even though the oldest lifts had been installed as early as the 1920s, the starting date of the process is considered 1956, when the Winter Olympics were held in Cortina d'Ampezzo, the oldest and most renowned tourist resort in the Dolomite region. The last period considered is 1991[1].

The data sources are the regional and provincial "cable-vehicle" departments, and the national Ministry for transportation. The observed variable is the hour capacity (hc) of each SkiArea, i.e. the number of persons that can be carried uphill by the entire set of lifts of a SkiArea in an hour. The analyses are conducted at varying spatial resolution, mainly at the municipality, ski area, and province levels.

2.2 Spatial Diffusion as a Spatial Interaction Model

Traditional methods in spatial diffusion analysis have focused on spatial interaction models (Clark, 1984). In analogy with the gravity model (1), the spatial interaction model requires the definition of an indicator of mass and the subsequent calibration of three parameters: a constant value k, the exponent α, indicating the attraction effect, and the inverse exponent β, representing the friction of distance (d_{ij}). Traditional applications include urban areas, and the "mass" (Pi, Pj) is typically constituted by the population of the interacting spatial units.

In spatial diffusion analysis, the model presents the advantage of describing the transmission mode of the diffusion, i.e. hierarchy vs. contagion. The significance of the α coefficient expresses the prevalence of the attraction effect, i.e. a hierarchical transmission that "jumps" from large to smaller centers, and where the distance effect is secondary. Vice versa, the significance of the β coefficient indicates a contagious transmission, where the prevalent interaction takes place among nearby units, and the mass effect tends to be negligible.

$$I_{ij} = k\,(P_i\,P_j)^{\alpha} / d_{ij}^{\beta} \quad (1)$$

A true spatial interaction model for the skilift case study should consider the interaction (I_{ij}) between the mountain resorts, where skilifts are located, and the urban areas, where the tourist flow originates. The estimation of such a model should begin with the definition of a common "mass" indicator (Pi, Pj), i.e. a variable that can represent the attraction exerted by the mountain resorts as well as that exerted by the population of the lowland urban centers.

Within the scope of this paper, a simpler version of the model is discussed, which analyzes only the diffusion of ski-lifts, in a univariate model. The latter model will consider the hour capacity (hc) as the interest variable. Equation (1) thus becomes:

$$Hc_{ij} = k\,(hc_i\,hc_j)^{\alpha} / d_{ij}^{\beta} \quad (2)$$

providing a model that describes only the spatial interaction among ski resorts.

The estimation of the spatial interaction model requires the preliminary evaluation of the spatial autocorrelation among the spatial units of interest. One serious limitation of the model is its lack of dynamics, in fact the model is spatial. A feasible

[1] A few seasons of unusually scarce snow precipitation in the early 1990s have significantly altered the spatial pattern and the investiment strategy of the ski industry in the Dolomite region. This consideration has suggested excluding recent data from the present analysis.

solution to this limitation can be the specification of a spatial interaction SUR model (Anselin, 1988), where a time series of spatial equations is estimated jointly. The spatial autocorrelation would then be considered in the error matrix for the GLS estimation, following Anselin (1988). A serious limitation of this model remains its lack of direction. The interaction value and the model parameters, in fact, gives no indication as to the direction of the diffusion, nor in the spatial or in the spatio-temporal SUR specification.

An alternative to the spatial interaction model is a spatial and temporal autoregressive (AR) model. The AR specification is univariate, as the dependent variable (hc) is explained by its lagged spatial and temporal values. In its general form an autoregressive stochastic process in space and time is expressed by (2):

$$X_{i\,t} = f(x_{j\,t-k}) + \varepsilon \qquad (2)$$

The autoregressive specification is strongly directional in time and in space, providing an alternative perspective on the dynamic diffusion process. The complexity of the spatio-temporal autoregressive structure requires an in-depth analysis of the spatial and temporal autocorrelation prior to the model estimation. An innovation diffusion process is assumed to present spatial and temporal non-stationarities, and likely spatial anisotropy. These characteristics require integration (ARIMA model), which in turn should be calibrated on the autocorrelation structure of the process.

3 Application and Results

3.1 Spatial Autocorrelation and Its Effects

The effects of spatial autocorrelation are well known and can be broadly summarized in a loss of efficiency, or increase in model variance, resulting in less reliable parameter estimates. The traditional solution to this problem is a GLS approach, which includes a spatial component, represented by a contiguity matrix, where the definition of contiguity remains the most crucial element, yet no universally accepted solution has been found, and the definition of contiguity is typically based on shared borders or on a threshold distance. Common indices for the measurement of spatial autocorrelation are the traditional Moran and Geary indices. Not only spatial non-stationarity, but also anisotropy, are very reasonable hypotheses for innovation diffusion models.

One of the main problems in the calculation of spatial autocorrelation is the definition of a contiguity matrix (Bertazzon, 2002); in light of the observed characteristics of the diffusion process, a contiguity matrix based on shared borders up to the second order neighbour was selected, and the spatial weights are defined by the inverse of the squared distances. Due to the complexity of the alpine morphology and the socio-economic differences within the area, the following calculations are limited to two representative provinces: Belluno, the historic center of the process, and Bolzano, the latest province to join the process, but the farthest reaching in terms of spatial diffusion and hour capacity.

The selected autocorrelation index is Moran's I. The results for the two selected provinces are reported in Table 1. Moran's I displays non-0 values for all the time

periods[2]. The negative value of 1966, consistent across the provinces, marks the entrance of Bolzano in the diffusion process. The result may be interpreted as the transition between the first stage of the diffusion process (initial agglomerations), and the second stage (the emergence of secondary agglomerations) (Hagerstrand, 1973). Following Hagerstrand's model, the 1991 interval, characterized by a large discrepancy between new cumulated hour capacity, may be interpreted as the beginning of the third stage (saturation). Accordingly, for Bolzano, 1976 marks the beginning of the second stage, which continues at least until 1991.

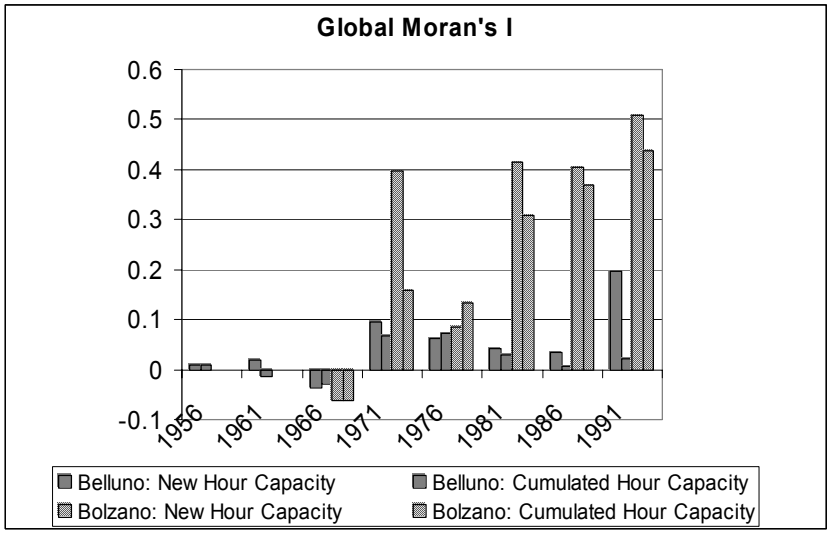


Table 1: Spatial Autocorrelation among municipalities

The temporal pattern of the spatial autocorrelation in the province of Bolzno exhibits an extraordinary regularity, suggesting a systematic diffusion process, that gradually filled the entire territory, achieving an increasing similarity among spatial units.

3.2 Temporal Autocorrelation and Its Effects

Temporal autocorrelation also results in a loss of efficiency, usually dealt with by means of GLS methods. Yet temporal non-stationarity, as well as spatial non-stationarity and anisotropy are the assumptions of any diffusion model. Temporal autocorrelation has been tested on single municipalities and ski-areas, on the entire study area, and on the three regions in study area. At any spatial resolution, and considering time lags of 1 year, the autocorrelation displays remarkably high values for low order lags: values consistently above .09 for the first lag, and above .08 for the second lag. While tapering off for subsequent periods, the autocorrelation remains significant also for high order lags. At 5-year intervals, consistent with the temporal break-down of the spatial analysis, temporal autocorrelation still displays values

[2] The diffusion process started only in 1966 in the province of Bolzano.

generally around .6 (for the cumulated hc) at a coarse spatial resolution, i.e. the regions, and the total. At a finer spatial resolution, i.e. municipalities and skiaerea, the values are consistently lower.

Table 2 displays the temporal autocorrelation at 5-year lags for the provinces of Belluno and Bolzano. For a first order lag (5 years interval) the temporal autocorrelation is above .05 for Bolzano and for the cumulated variable in Belluno.

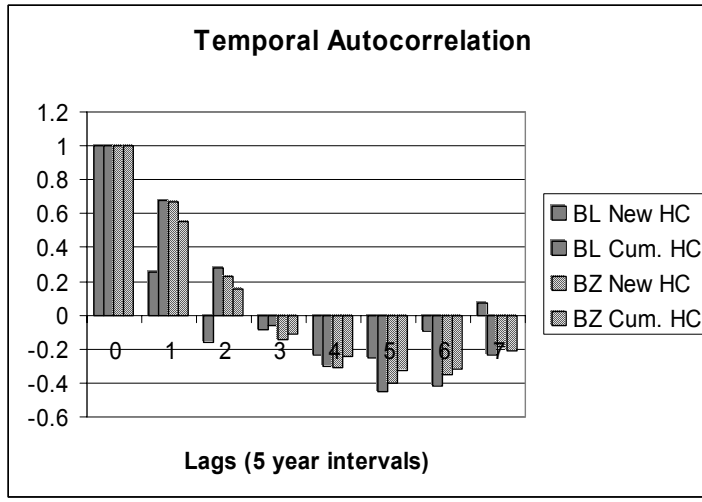


Table 2: Temporal autocorrelation for the provinces of Belluno and Bolzano

At lag 2 (10 year interval) the temporal autocorrelation decreases drastically, reaching consistently negative values at the third degree. Negative temporal autocorrelation at high order lags is frequently observed in time series analysis.

3.3 Joint Effects of Spatial and Temporal Autocorrelation

Considering the spatial and dynamic nature of spatial diffusion processes, quantitative models for these processes should include a spatio-temporal specification. As discussed, this is not a common solution in GIS and spatio-temporal analysis. The traditional analysis has focused on univariate specifications such as the traditional spatial interaction model known as gravity model (1), or in the autoregressive model (2). Such models requires the solution of a regression model where the effect of distance is calibrated by the exponent β, but the spatial autocorrelation between p_i and p_j is not considered. A spatial regression specification of this model, which would take into account the effect of spatial (and temporal) autocorrelation, represent a major step towards an efficient model.

The separate analyses of the spatial and temporal autocorrelation have produced interesting results. An aspect of particular interest, which may appear paradoxical, is the apparent temporal autocorrelation in the Moran's I. Table 3 represents such temporal autocorrelation for the province of Bolzano.

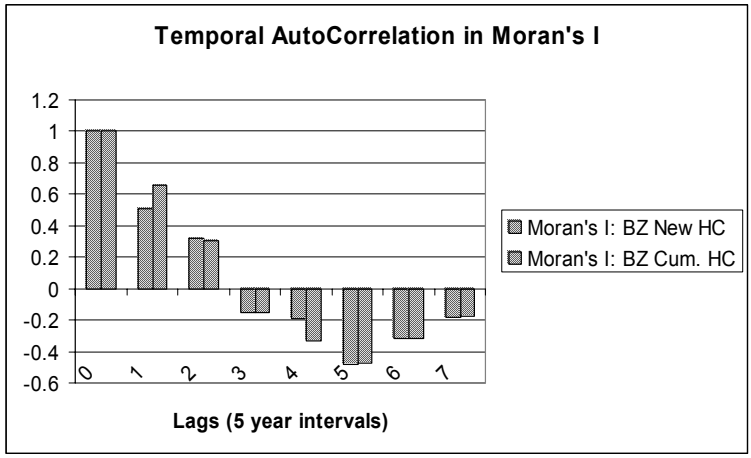


Table 3: Temporal autocorrelation in Moran's I

The pattern shown in Table 3 is remarkably close to the pattern of the temporal autocorrelation temporal autocorrelation in the data (Table 2).

3.4 Computational Solutions for a Spatio-temporal Moran's I

A spatio-temporal specification of the spatial interaction model (spatial SUR) is represented by a set of 8 equations, each representing one time period, while each of the equations represents a spatial interaction model[3].

$$
\begin{aligned}
&| I_{ji(1956)} = k_1 \, (hc_{i(1966)} \, hc_{j(1956)})^{\alpha_1} / d_{ij}^{\beta_1} | \\
&| I_{ji(1971)} = k_2 \, (hc_{i(1971)} \, hc_{j(1971)})^{\alpha_2} / d_{ij}^{\beta_2} | \\
&| \ldots. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad | \qquad (3) \\
&| \ldots. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad | \\
&| I_{ji(1991)} = k_1 \, (hc_{i(1991)} \, hc_{j(1991)})^{\alpha_6} / d_{ij}^{\beta_6} |
\end{aligned}
$$

The GLS solution to the set of equations in model (3) requires the estimation of a "hyper" autocorrelation matrix. In fact the autocorrelation matrix for each single spatial equation will be a full matrix due to the presence of spatial autocorrelation among the spatial observations. The autocorrelation matrix for each spatial equation can be represented as follows:

$$
Cor\,(hc_i \, hc_j) = \quad Cor \quad \begin{matrix} hc_{1(t1)} \, hc_{1(t1)} & hc_{1(t1)} \, hc_{2(t1)} & hc_{1(t1)} \, hc_{3(t1)} \\ hc_{2(t1)} \, hc_{1(t1)} & hc_{2(t1)} \, hc_{2(t1)} & hc_{2(t1)} \, hc_{3(t1)} \\ hc_{3(t1)} \, hc_{1(t1)} & hc_{3(t1)} \, hc_{2(t1)} & hc_{3(t1)} \, hc_{3(t1)} \end{matrix} \qquad (3)
$$

Using the symbol γ to focus on the spatial correlation, (3) becomes (4):

$$
SpCor\,(hc_i \, hc_j) = \Gamma\,(hc_i \, hc_j) = \begin{matrix} \gamma_{11(t1)} & \gamma_{12(t1)} & \gamma_{13(t1)} \\ \gamma_{21(t1)} & \gamma_{22(t1)} & \gamma_{23(t1)} \\ \gamma_{31(t1)} & \gamma_{32(t1)} & \gamma_{33(t1)} \end{matrix} \qquad (4)
$$

[3] The autoregressive model in (2) is not discussed in detail, since the considerations on the spatial and temporal autoroccelation are equally applicable.

To investigate the spatial and temporal autocorrelation structure, each element γ_{ij} in the spatial autocorrelation matrix Γ should also include a calculation of the temporal autocorrelation. The temporal autocorrelation for each spatial element hc_i can be expressed as:

$$\text{Cor}\,(hc_{i(t1)\,(t+k)}) = \text{Cor} \begin{matrix} hc_{i(t1)\,(t1)} & hc_{i(t1)\,(t2)} & hc_{i(t1)\,(t3)} \\ hc_{i(t2)\,(t1)} & hc_{i(t2)\,(t2)} & hc_{i(t2)\,(t3)} \\ hc_{i(t3)\,(t1)} & hc_{i(t3)\,(t2)} & hc_{i(t3)\,(t3)} \end{matrix} \quad (5)$$

Using the symbol ρ to express autocorrelation in time:

$$\text{TCor}\,(hc_{i(t1)\,(t+k)}) = P(hc_{i(t1)\,(t+k)}) \quad \begin{matrix} \rho_{i(t1)\,(t1)} & \rho_{i(t1)\,(t2)} & \rho_{i(t1)\,(t3)} \\ \rho_{i(t2)\,(t1)} & \rho_{i(t2)\,(t2)} & \rho_{i(t2)\,(t3)} \\ \rho_{i(t3)\,(t1)} & \rho_{i(t3)\,(t2)} & \rho_{i(t3)\,(t3)} \end{matrix} \quad (6)$$

The joint evaluation of spatial and temporal autocorrelation will produce what could be called a Spatio-Temporal AutoCorrelation (STAC) index. Its calculation requires that each γ_{ij} in (4) be multiplied by the entire matrix P, producing a spatio-temporal Γ–P matrix, that can be expressed, using the Kronecker product, as $\Gamma \otimes P$:

$$\text{STAC} = \Gamma \otimes P = \begin{matrix} \gamma_{11(t1)}\,P & \gamma_{12(t1)}\,P & \gamma_{13(t1)}\,P \\ \gamma_{11(t1)}\,P & \gamma_{12(t1)}\,P & \gamma_{13(t1)}\,P \\ \gamma_{11(t1)}\,P & \gamma_{12(t1)}\,P & \gamma_{13(t1)}\,P \end{matrix} \quad (7)$$

The effect of the Kronecker product is that each element in the first matrix is multiplied by the entire second matrix, as expressed in (7), where each γ_{ij} is multiplied by the entire P, producing, in fact, a hyper-matrix, or a matrix of matrices, or a multi-dimensional matrix. The calculation of the spatio-temporal autocorrelation described in (7) can be achieved in alternative ways. Alternative decisions on the computational strategy impact on the resulting index.

In this paper the Moran's I has been discussed, as a global indicator of spatial autocorrelation. One alternative to the Moran's I is the calculation of local indicators of spatial association (LISA) (Anselin, 1999), which would indeed provide a Γ matrix as described in (4). Likewise, "local" indicators of temporal association can be calculated. Once both of these local measures have been calculated, the Kronecker product can be specified and the local spatio-temporal autocorrelation can be evaluated as in (7).

Using global indicators, such as the Moran's I for spatial autocorrelation and the correlation coefficient for temporal autocorrelation, a spatio-temporal index can be derived using the same approach. The following implementation will focus on the province of Bolzano, which displays the greatest spatial autocorrelation and the highest regularity in time. Indeed, considering the break-up of the data into 5-year temporal intervals, in the hypothesis of a multiplicative effect of spatial and temporal autocorrelation, a global spatio-temporal autocorrelation index can be derived by multiplying the raw-vector of Moran's I values in each period, by the column-vector of the ρ temporal autocorrelation coefficients, in reverse order, to reflect the chronological sequence of the diffusion process. The Spatio-Temporal Global Index (ST-GI) is expressed in (8).

$$\text{ST-GI} = [I_{56} \quad I_{61} \quad \ldots \quad I_{91}] * [\rho_8 \quad \rho_7 \quad \ldots \quad \rho_1]^T \quad (8)$$

In this application, for the province of Bolzano, the value of the indicator is 0.76; for the province of Belluno, it results in a 0.177 value. Such global indicators of spatio-temporal autocorrelation are of rather difficult interpretation.

Alternative, the vector multiplication in (8) could be replaced by

$$\text{ST-GI} = [I_{56} \quad I_{61} \quad \dots \quad I_{91}]^T * [\rho_8 \quad \rho_7 \quad \dots \quad \rho_1] \qquad (9)$$

Transposing the first vector instead of the second produces instead a matrix of spatio-temporal indicators, that combines each period with a temporal lag, as shown in (10).

$$\begin{array}{cccccccc}
I_{56}*\rho_8 & I_{56}*\rho_7 & I_{56}*\rho_6 & I_{56}*\rho_5 & I_{56}*\rho_4 & I_{56}*\rho_3 & I_{56}*\rho_2 & I_{56}*\rho_1 \\
I_{61}*\rho_8 & I_{61}*\rho_7 & I_{61}*\rho_6 & I_{61}*\rho_5 & I_{61}*\rho_4 & I_{61}*\rho_3 & I_{61}*\rho_2 & I_{61}*\rho_1 \\
I_{66}*\rho_8 & I_{66}*\rho_7 & I_{66}*\rho_6 & I_{66}*\rho_5 & I_{66}*\rho_4 & I_{66}*\rho_3 & I_{66}*\rho_2 & I_{66}*\rho_1 \\
I_{71}*\rho_8 & I_{71}*\rho_7 & I_{71}*\rho_6 & I_{71}*\rho_5 & I_{71}*\rho_4 & I_{71}*\rho_3 & I_{71}*\rho_2 & I_{71}*\rho_1 \\
I_{76}*\rho_8 & I_{76}*\rho_7 & I_{76}*\rho_6 & I_{76}*\rho_5 & I_{76}*\rho_4 & I_{76}*\rho_3 & I_{76}*\rho_2 & I_{76}*\rho_1 \\
I_{81}*\rho_8 & I_{81}*\rho_7 & I_{81}*\rho_6 & I_{81}*\rho_5 & I_{81}*\rho_4 & I_{81}*\rho_3 & I_{81}*\rho_2 & I_{81}*\rho_1 \\
I_{86}*\rho_8 & I_{86}*\rho_7 & I_{86}*\rho_6 & I_{86}*\rho_5 & I_{86}*\rho_4 & I_{86}*\rho_3 & I_{86}*\rho_2 & I_{86}*\rho_1 \\
I_{91}*\rho_8 & I_{91}*\rho_7 & I_{91}*\rho_6 & I_{91}*\rho_5 & I_{91}*\rho_4 & I_{91}*\rho_3 & I_{91}*\rho_2 & I_{91}*\rho_1
\end{array} \qquad (10)$$

The calculations for the provinces of Belluno and Bolzano produce (11) and (12), respectively.

0.001	0.000	-0.002	-0.002	-0.001	-0.002	0.002	0.010
0.001	0.000	-0.005	-0.005	-0.002	-0.003	0.005	0.019
-0.003	0.001	0.009	0.009	0.003	0.006	-0.009	-0.036
0.007	0.009	-0.024	-0.023	-0.008	-0.015	0.024	0.095
0.004	0.004	-0.016	-0.015	-0.005	-0.010	0.016	0.062
0.003	0.002	-0.010	-0.010	-0.003	-0.007	0.011	0.042
0.002	0.001	-0.008	-0.008	-0.003	-0.005	0.009	0.033
0.014	0.038	-0.049	-0.047	-0.016	-0.032	0.050	0.196

(11)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0.012	0.022	0.026	0.02	0.009	-0.01	-0.04	-0.06
-0.08	-0.14	-0.16	-0.12	-0.06	0.092	0.264	0.396
-0.02	-0.03	-0.03	-0.03	-0.01	0.02	0.057	0.086
-0.08	-0.14	-0.17	-0.13	-0.06	0.096	0.276	0.414
-0.08	-0.14	-0.16	-0.13	-0.06	0.094	0.27	0.405
-0.1	-0.18	-0.21	-0.16	-0.07	0.118	0.339	0.509

(12)

Even though the structure of matrix (10) is simple, the interpretation of all values in the applied tables (11) and (12) results complex. The multiplication of the spatial autocorrelation by the temporal and autocorrelation for each lag results in a general reduction of the magnitude of the coefficients. The sign of the resulting correlation is a product of the two effects.

4 Conclusion and Future Research

Global spatial and temporal autocorrelation indices have been calculated for the innovation diffusion process, the most interesting results were yielded at the provincial level. After discussing the spatial and temporal autocorrelation separately, a single spatio-temporal global index has been proposed. More detailed solutions can be proposed using local indicators, both spatially and temporally.

The main issue, both at the local and the global scale, remains whether the joint effect of spatial and temporal autocorrelation is in fact multiplicative or of a different nature. If the effect is not multiplicative, but, for example, additive, a summation operation should replace the product, whether among matrices or scalars. Other effects, beyond additive and multiplicative, should be experimented with.

After testing different effects, the following step should be a quantitative evaluation of each hypothesized effect. Such evaluation requires the definition of one or more criteria. The objective of the calculation of spatio-temporal autocorrelation is the minimization of the statistical inefficiency, or the minimization of the variance of the estimates in a dynamic spatial model (SUR spatial interaction model, or in a spaio-temporal autoregressive integrated model). Consequently, the criterion to evaluate the best specification of the effect of spatio-temporal autocorrelation should be the comparison of the variances of GLS models, where the inverse variance-covariance matrix obeys the respective specifications.

References

1. Anselin L. 1999. Interactive Techniques and Exploratory Spatial Data Analysis. In: Longley PA, Goodchild MF, Maguire DJ, Rhind DW, editors. Geographical Information Systems. Second Edition. New York: Wiley. p 253-266.
2. Anselin L (1988) *Spatial Econometrics: Methods and Models*, New York: Kluwer Academic Publisher.
3. Bertazzon S (2002) Metaspace: From a Model of Spatial Contiguity to the Conceptualization of Space in Geo-Analyses. *Joint International Symposium on Geospatial Theory, Processes and Applications*. Ottawa, July 8-12, 2002.
4. Clark, G., 1984, Innovation Diffusion: Contemporary Geographical Approaches, CATMOG 40, Norwich: Geobooks.
5. Egenhofer anf Golledge, 1998, Spatial and Temporal Reasoning in Geographical Information Systems. Oxford: Oxford University Press.
6. Fotheringham A S, Brundson C, Charlton M, 2000, *Quantitative geography. Perspectives on Spatial Data Analysis*. London: Sage.
7. Griffith D A, Layne L J (1999) *A Casebook for Spatial Statistical Data Analysis*, Oxford: University Press.
8. Hagerstrand, T., 1973, Innovation Diffusion as a Spatial Process. Chicago, The University of Chicago Press.
9. Longley P A, Brooks S M, McDonnell R, MacMillan B (Eds.), *Geocomputation*, Chichester: John Wiley & Sons, 77-94.
10. Pfeifer and Deutsch, 1980, Identification and Interpretation of of First order Space-Time ARMA Models; Technometrics, 22, pp 397-408.
11. Raper J (2000) *Multidimansional Geographic Information Science*. London: Taylor & Francis
12. Rogers, E, 1995, Diffusion of Innovations. New York: Free Press, 1995.

Testing Differences between Case and Control Point Patterns Using Nearest Neighbour Distances and Bootstrapping

Kevin A Henry[1], Leif M Burge[1], and Dao Nguyen[2]

[1] McGill University, Department of Geography, Montreal, QC, Canada H3A 2K6,
Khenry1@po-box.mcgill.ca, lburge2@po-box.mcgill.ca
[2] McGill University Health Centre, Department of Medicine,
1650 Cedar ave, QC, H3G 1A4, Canada
dao.nguyen@mail.mcgill.ca

Abstract. This paper proposes a method for comparing point patterns in case and control data by using nearest neighbour distances, bootstrapping and the Wilcoxon Signed-Rank Test. Case-control comparisons are frequently used in medical geography and epidemiology to examine the patterns of disease and infer transmission of infectious pathogens. Our strategy addresses the problem of handling spatial analysis when the numbers of cases differ from controls. Differences in sample sizes can affect density of points and therefore bias nearest neighbour distances. To demonstrate this method we created a control set of 250 points and two sets of cases 125 points each. Bootstrapping the control data and comparing each run statistically to the cases can provide confidence intervals and estimate the risk of erroneously rejecting the null hypothesis. We follow with a case study of tuberculosis. The spatial distributions of different bacterial strains were compared and the nearest neighbour distances were analyzed as a surrogate for possible transmission of tuberculosis. The method may be useful to epidemiologists, geologists, biologists, geographers and ecologists for evaluating differences between the spatial structures of points.

1. Introduction

This paper presents a method for the comparison of two point pattern populations differing in size. In a case-control design, where cases are the data points representing the event of interest and controls are the data points representing the comparison group, the control sample is frequently larger in size. It is often common in a case-control study to have a greater number of controls to compensate for variability, but this difference can introduce errors when trying to interpret spatial point patterns or nearest neighbour distances because as the number of points (controls) increases so does the density of points, thereby decreasing the nearest neighbour distances.

We propose bootstrapping as a method to adjust for bias in nearest neighbour distances caused by differences in the numbers of cases and controls, a bias not specifically addressed by tools such as Geographical Information Systems or other available analytic strategies such as spatial autocorrelation, nearest neighbour

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 33-42, 2003.
© Springer-Verlag Berlin Heidelberg 2003

analysis, quadrat analysis, the G, F, and K distance functions, and other geographic clustering techniques [1],[2],[3],[4],[5]. Tests that do use case-controls to assess point patterns tend to use the controls as a variable to appraise whether or not the cases are clustered or whether they meet an expected distribution or pattern that matches the controls. For example, Cuzick and Edward's and Diggle's clustering methods takes into account point density, but like so many of the specialized methods both location and distance are used to determine point analysis [15],[16][1].

Bootstrapping is a technique where a sample of a specified size is repeatedly and randomly sampled with replacement from the original population. When little is known about a population, bootstrapping provides a guide to the population distribution and confidence limits and in this situation provides a straightforward way of dealing with a greater number of controls [6],[7],[8],[9]. Boostraping has been extensively applied to estimate standard errors, construct confidence intervals and test statistical significance in the fields of biology, genetics and ecology.

Through a simulated example, we demonstrate how unequal sample size can result in erroneous statistical analyses of the nearest neighbours when unadjusted for the sample size, or when multiple resamplings are not performed. This method is then illustrated by a real world case study of drug resistant tuberculosis. The nearest neighbour distance served as a surrogate for the probability of infectious transmission. The point pattern analysis of tuberculosis was used to understand the spatial structure of these events and infer the transmission dynamics of tuberculosis in the different groups (cases and controls).

2. Simulation and Exposition of Method

Our method for comparing the spatial structure of cases and controls involves: bootstrapping the controls (using x and y coordinates), calculating the nearest neighbour population for each new set of bootstrapped controls, statistically comparing the new control set with the cases and estimating the probability of rejecting the null hypothesis (Fig. 1). Our null hypothesis (Ho) is that the cases and controls (points) have a similar spatial structure (nearest neighbour distances). Our alternative hypothesis (Ha) is that the cases and controls have a different spatial structure (nearest neighbour distances). We incorporated and automated these steps in a SAS macro program

Two simulated scenarios are presented: a set of dispersed cases and a set of clustered cases (both with 125 points per set) are compared with a set of controls (with a larger population of 250 points) (Fig. 2).

[1] Most point statistic algorithms can be found in programs like S Plus, SAS, Clusterseer or in specialized packages for the free R statistical software (http://www.stats.bris.ac.uk/R). Comprehensive reviews of point pattern statistics can be found in literature by Diggle and Boots and Getis.

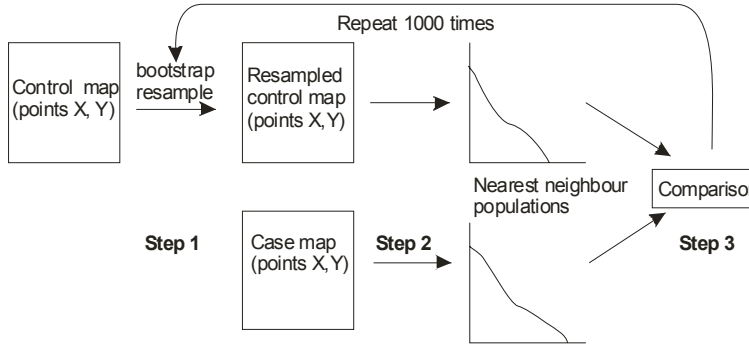


Fig. 1. Steps in method for comparing cases and controls using bootstrapping.

At step 1, 125 controls were bootstrapped (resampled) from the 250 controls, 1000 times to produce case-control data sets of identical size (Fig. 1). Each resampled data set was then used to produce confidence intervals and compare each new control sample to the cases to determine whether the two data sets were similar or different.

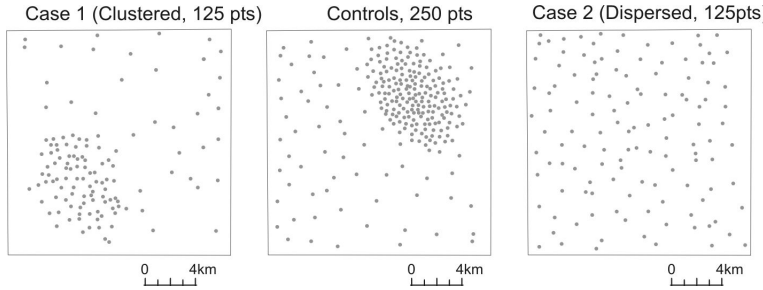


Fig. 2. Maps displaying clustered and dispersed cases and controls.

In our method, bootstrapping produces a data set that has the same number of controls as cases and is adjusted to represent more accurately the density of cases. Also, by resampling a large number of times you are taking into account almost all possible combinations of nearest neighbour distances for randomly selected data. Commonly scientists assume that a single sample of data represents a population. There are however, differences in median values, and confidence limits vary when the data is resampled hundreds of times.

At step 2, we determined the nearest neighbour distances for each of the 1000 bootstrapped controls and the two sets of 125 cases (Fig. 1). Each of the nearest neighbour distances was used to compare cases to controls (Fig. 2). Nearest neighbour distance was defined as the Euclidean distance between a point (case or control) and the nearest other point in the same set.

Without the bootstrapping method, the nearest neighbour distances for the clustered cases ranged from 0.01 km to 5.6 km, from 0.7 km to 3.1 km for the dispersed case and from 0.3 km to 3.8 km for the original data set of 250 controls (Table 1). Median nearest neighbour distance was greatest for the dispersed cases (1.6

km), moderate for the clustered cases (0.8 km), and lowest for the 250 controls (0.7 km). For all three examples, we derived the G distance function that calculates the cumulative frequency distribution of nearest neighbour distances.[2]

Table 1. Nearest neighbour values for the two cases and controls.

	Case 1 (Clustered)	Controls	Case 2 (Dispersed)
N of cases	125	250	125
Minimum	0.01	0.3	0.7
Maximum	5.6	3.8	3.1
Range	5.6	3.5	2.4
Median	0.8	0.7	1.6
Mean	1.2	1.0	1.6
Std. Error	0.1	0.04	0.05

The G statistic clearly show the difference between dispersed and clustered cases as well as the differences in the 250 controls and 125 clustered cases (Fig. 3). From those descriptive statistics, one may conclude that each of the three point distributions (cases-controls) has different nearest neighbour distances.

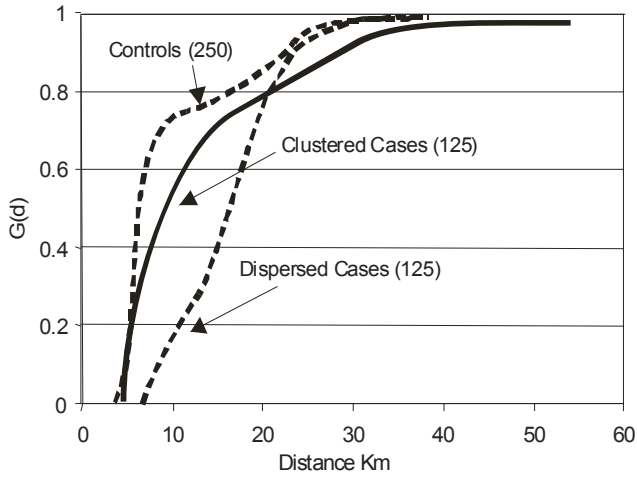


Fig. 3. Nearest neighbour G function for the two cases and controls.

Since the larger control data set inherently has a greater point density, and thereby shorter nearest neighbour distances, we adjust for point density by taking a sub-sample of the 250 controls. With 1000 resamplings, the control population had medians ranging from 0.69 to 0.96, close to that for the clustered cases (Fig. 4). If only a single random sample of 125 was taken from the control population, the null hypothesis might be falsely rejected (type 1 error). Scientists commonly assume that

[2] The G distance function is calculated as $G(d) = \frac{no.[d_{\min(si)} < d]}{n}$

the resampled data represents the original population. A large number of resampling allows an approximation of confidence levels. Researchers recommend that a minimum of 100 samples be taken for bootstrapping, but suggest using a thousand or more to accurately represent the data [10].

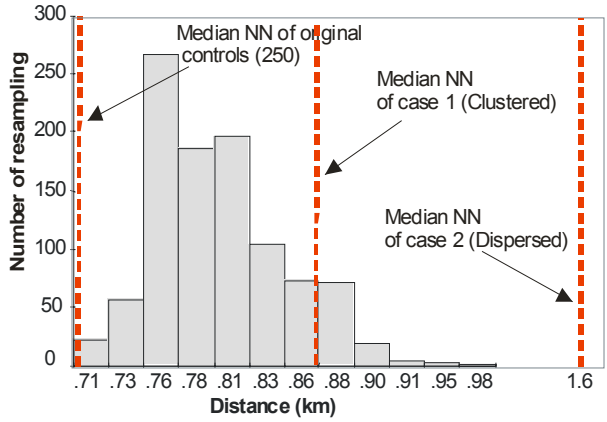


Fig. 4. Median values for the two cases and the 1000 bootstrapped controls (NN means nearest neighbour).

The third step in the method is comparison of the 1000 bootstrapped control data sets to the case data sets. Nearest neighbour data is non-parametric and is largely skewed to lower values because of a large number of close nearest neighbours and a small number of distant nearest neighbours. Attempts to transform and normalize the data failed, therefore the non-parametric Wilcoxon Signed-Rank Test was chosen. This test estimates the differences between the resampled control data sets and each case data set. The Wilcoxon test compares the rank values of two data sets, pair by pair and then computes the sum of ranks associated with positive differences and the sum of ranks associated with negative differences. The test statistic is the lesser of the two sums of ranks.

We obtained p values for each comparison of controls to cases. The 1000 probability values provide a final p value and standard error of the p as way of deciding whether to accept or reject the null hypothesis (Ho). For each of the 1000 permutations the test statistic was compared to the chosen p-value (*tprob*). We set *tprob* at 0.05. Each time T is greater than *tprob,* we accept the null hypothesis of no difference. The equation below can be used to compute probability values for all permutations [7].

$$p = [T \geq t_{prob} | H_0] = \frac{\#\{T^{(i)} \geq t_{prob}\}}{Nb} \quad \textbf{(1)}$$

For the dispersed cases, T was smaller than the *tprob* (0.05) in 98% of the 1000 comparisons with the bootstrapped control data sets. We therefore accept and infer with a high level of significance the alternate hypothesis (Ha) that the dispersed cases have a different spatial structure from the controls.

$$p = [T \geq t_{prob} | H_0] = \frac{\#\{987 \geq .05\}}{1000} = .987 \quad (2)$$

For the clustered case data set, T was greater than *tprob* (0.05) in 99% of the 1000 comparisons with the bootstrapped control data sets. Comparison of these cases to controls indicated that only 10 times in 1000 would we reject the null hypothesis. The p-values varied from 0.99 to 0.01 during the 1000 runs (Fig. 5), indicating variability in the resampled controls making certain controls not as strongly related to the cases as others. Overall, we must accept the null hypothesis (Ho) that the clustered data set does not have spatial structure different than the control data set, at least in terms of its nearest neighbour distance properties.

$$p = [T \geq t_{prob} | H_0] = \frac{\#\{10 \geq .05\}}{1000} = .01 \quad (3)$$

In non-spatial epidemiological studies, unequal controls and cases are often the norm, but when the analysis involves geographic point distances with a greater number of controls it is important to consider point density and its influence on decreasing nearest neighbour distances. Our application to the simulation with its exemplary design shows why bootstrapping is necessary: we reach a different conclusion when we adjust for comparable sample sizes.

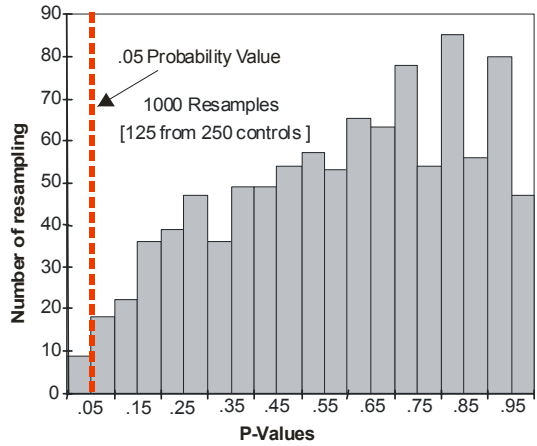


Fig. 5. P values for the difference tests between the clustered and 1000 bootstrapped controls.

3. Case Study: Tuberculosis Transmission

During the last decade, a number of new spatial statistical methods and the use of Geographic Information Systems have improved epidemiological and geographic investigations of diseases [11]. They have provided additional insights into the

transmission of tuberculosis (TB), a contagious infection caused by a mycobacterium transmitted from person-to-person through aerosolized droplets, and hence physical proximity. For TB, many of these techniques have been mostly used in areas of high incidence in U.S. urban areas, but those studies have for the most part handled aggregate data with a spatial ecology approach [11],[12],[13]. The study of TB transmission has further been refined by the ability to differentiate TB strains by their molecular (or DNA) fingerprints. Two subjects infected with the same TB strain are identified by mycobacteria carrying the identical DNA fingerprint [13].

This example is a case-control application for comparing the spatial structure of different TB strains and examines the nearest neighbour distances of the controls and cases at a fine grained scale. Distinguishing genetic strains of TB also provides a variable for differentiating the spatial distributions of TB. Due to its airborne transmission, people living nearby are more likely to contract a specific local strain. TB cases scattered throughout a region can be differentiated by genetic markers, and transmission patterns of specific TB strains can be inferred from their spatial structure.

The case study originated from the study of a particular drug resistant strain in Quebec (77 case points). A control population, chosen from a random sample of TB strains without drug resistance in Quebec, served as the comparison group (253 control points). These case-controls were further subdivided into subsets for further genetic differentiation: 34 drug resistant cases with identical IS6110 restriction fragment length polymorphism (RFLP) (genetically similar) and 191 controls without identical RFLP matches. The cases and controls selected for this study all come from the same pool of reported TB cases and shared comparable epidemiologic links and demographic characteristics (older age, non-urban dwellers). The goal of the geographical analysis was to determine whether the cases and controls had similar spatial structures, and by extension whether the drug resistant TB strain had a different transmission pattern compared to the controls. The null hypothesis (Ho) is that there is no difference in the spatial structure of cases and controls.

Six-digit postal codes were used to map each of the cases and controls. Canadian postal codes provide a very fine grained geocoding, as compared to the U.S. Zip code. There are a total of 225,000 for the province; whereby the postal code coverage varies from a village of a few thousand people or areas as small as a city block or even an apartment building [14]. The postal code coverage allows us to geocode each of the case-control addresses to a very precise location.

As in our simulated example, unadjusted descriptive statistics suggest that the spatial structures differed between case and control. The median nearest neighbour distance for the 77 drug resistant cases was 6.6, while only 2.2 for the 238 controls (Fig. 6). The median nearest neighbour distance for the 34 identical RFLP cases was 11.6 and only 2.3 for the 191 controls (Table 2).

In each instance, we have an over-representation of controls. When adjusting for the different sample sizes with our bootstrapping method, we found no significant differences between cases and controls. The median nearest neighbour distance for the 77 drug resistant cases was 6.6 km compared to 5.3 km for the median of the median of the 1000 bootstrapped controls (Fig. 6). When the nearest neighbour distances for each of the 1000 permutations were compared with the nearest neighbour distances from 77 drug resistant cases using the Wilcoxon Signed-Rank Test, 97 % (28 of the 1000 runs) of the comparisons produced p-values ≥ 0.05 (Fig.

Table 2. Nearest neighbour values for the cases and controls

	Controls Drug Sensitive	Cases Drug Resistance	Controls Non Identical IS6110	Cases Identical IS6110
N of cases	238	77	194	34
Minimum	0.001	0.639	0.001	0.736
Maximum	69.3	131.9	69.3	207.1
Range	69.3	131.2	69.3	206.3
Median	2.2	6.6	2.3	11.6
Mean	7.0	17.0	7.6	28.6
Std. Error	0.7	3.2	0.9	8.0

7). In our second set of cases and controls, we found that our cases differed from the controls only 17 of the 1000 permutations (Fig. 7).

$$p = [T \geq t_{prob} | H_0] = \frac{\#\{972 \geq .05\}}{1000} = .972 \quad \textbf{(4)}$$

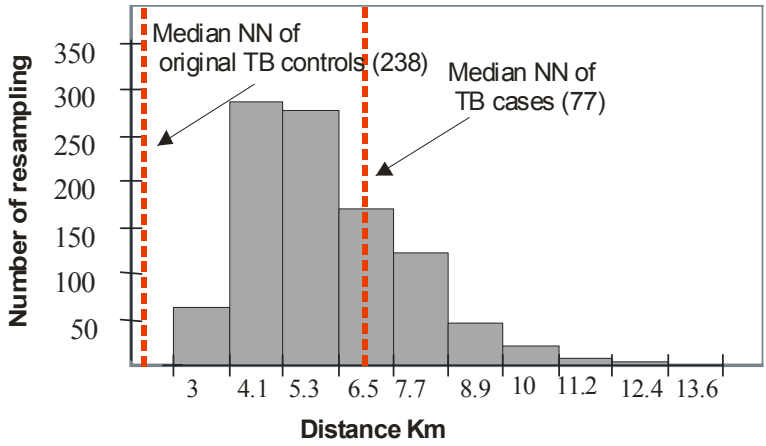


Fig. 6. Median values for the drug resistant cases and the 1000 non-drug resistant bootstrappedcontrols. NN means nearest neighbour.

We therefore accepted the null hypothesis and concluded that in both examples cases had a spatial structure similar to the controls, and hence transmission of the resistant TB strain was not more likely than the non-resistant control strains.

4. Discussion and Conclusion

This paper proposed a new technique for comparing point patterns of cases and controls when there are a greater number of controls than cases. Bootstrapping was

employed because nearest neighbour distances decrease when point density increases and point density is controlled, for a given area, by the number of points. Therefore, comparing point data sets with uneven sample sizes produces misleading results. Uneven sample size for non-spatial data is usually acceptable when data is randomly sampled because you can estimate confidence intervals, but when measuring distances between points uneven samples will cause a distance bias. This is particularly

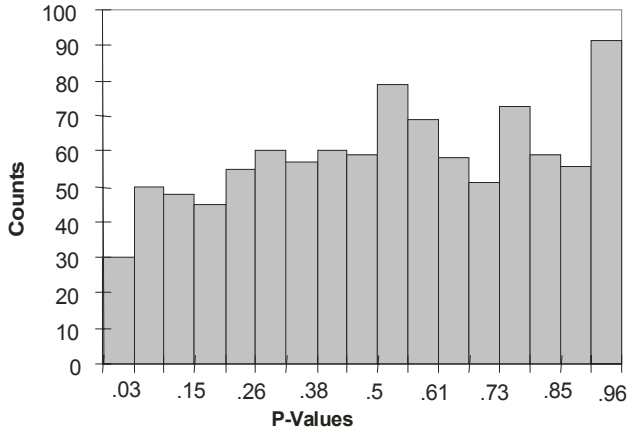


Fig. 7. P values for the difference tests between the 77 drug resistant cases and 1000 bootstrapped non-drug resistant controls.

important when dealing with small sample sizes. In these situations nearest neighbour distances may decrease when the sample size is increased because of the high probability that the points being added would be in higher population locations. Thus, to compare a larger number of controls to cases, the controls must be bootstrapped to equal the number of cases.

We examined the method using a simulated example where the spatial structure of controls had similar or different spatial structure to the cases. In our case study, we examined the spatial structure of two unique tuberculosis strains and observed that each had similar spatial structure to their paired controls. In both examples we initially examined the possibility of wrongfully rejecting the null hypothesis that differences in nearest neighbour distances are different in the cases and controls. We then illustrated that bootstrapping can be used to create equal sample sizes that allows, with a high level of confidence, whether to accept or reject the null hypothesis.

Our case study of tuberculosis illustrates the potential application of this method in epidemiological research of infectious diseases where the spatial structure of events can be used to infer the transmission of the infectious pathogens, and the nearest neighbour distance bias could lead to erroneous inferences. This method may also be useful to any researcher analyzing the spatial structure and variability of geographic data and needs to test its spatial structure, variability and accuracy. We believe integrating similar bootstrapping methods into GIS software would be beneficial for treating sampled data and deriving confidence intervals.

5. References

1. Getis, Arthur. *Second-Order Analysis of Point Patterns: Chicago as a Multi-Center Urban Region*. Professional Geographer 35 (1983): 73-80.
2. Boots, Barry N, and Arthur Getis. *Point Pattern Analysis*. Edited by Grant Ian Thrall, Scientific Geography. London: Sage Publications, 1988.
3. Diggle, P.J. *A Point Process Modelling Approach to Raised Incidence of a Rare Phenomenon in the Vicinity of a Prespecified Point*. Journal of the Royal Statistical Society 157 (1990): 433-40.
4.Kulldorff, G. M, and N. Nagarwalla. *Spatial Disease Clusters: Detection and Inference*. Statistics in Medicine 14 (1995): 799-810.
5.Diggle, P.J. *Statistical Analysis of Spatial Point Patterns*. New York: Academic Press, 1983.
6.Efron, B. *Nonparametric Standard Errors and Confidence Intervals*. *Canadian Journal of Statistics* 9 (1981): 139-72.
7. Davison, A.C, and D.V. Hinkley. *Bootstrap Methods and Their Applications*, *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge: Cambridge University Press, 1997.
8.Solow, A. R. *Bootstrapping Sparsely Sampled Spatial Point Patterns*. Ecology 70, no. 2 (1989): 379-82.
9.Chernick, Michael R. *Bootstrap Methods: A Practitioner's Guide*, *Probability and Statistics Series*. New York: Wiley, John & Sons, Incorporated, 1999.
10.Manly, B. F. J. *Randomization and Monte Carlo Methods in Biology*. London: Chapman and Hall, 1991.
11.Beyers, N. et al. 1996. *The use of a geographical information system (GIS) to evaluate the distribution of tuberculosis in a high-incidence community*. S Afr Med J. 86: 40-44
12. Acevedo-Garcia D. *Zip-code level risk factors for tuberculsosis: neighborhood environment and residential segregation in New Jersey, 1985-1992*. Am J Public Health 2001; 91: 734-741.
13.Kulaga S, Behr MA, Schwartzman K. *Genetic fingerprinting in the study of tuberculosis transmission*. CMAJ 1999; 161: 1165-9.
14.Statistics Canada. Postal code counts - *Population and dwelling counts*. 2001,Catalogue no. 92-F0086-XCB
15.Cuzick, J, and R. Edwards. *Spatial Clustering for Inhomogeneous Populations*. Journal of the Royal Statistical Society 52 (1990): 73-104.
16.Diggle, P.J, and Julian Besag. *Statisitcal Analysis of Spatial Point Patterns by Means of Distance Measures*. Biometrics 32, no. 3 (1976): 659-67.

Assessment of Different Link Functions for Modeling Binary Data to Derive Sound Inferences and Predictions

Falk Huettmann, Julia Linke

Geography Department, University of Calgary, Calgary AB T2N 1N4, Canada,
falk@ucalgary.ca

Abstract. Binary data are widely used for spatial modeling and when inferences and predictions are to be derived. If a Generalized Linear Model (GLM) is applied, logit functions are often used. Here we show alternatives to the traditional logit approach using probit and the complementary log log link functions. We present a software-based approach and two methods of assessing which link function performs best for inferences and for predictions. The first decision criterion is centered around the model deviance, e.g. relevant for inferences. The second criterion is based on predicting the findings back to the training data and then using the differences between expected and predicted values for known presences and absences as an indication of the fit. As an example we use Marbled Murrelet (*Brachyramphus marmoratus*) nesting habitat data derived from aerial telemetry and overlaid with GIS habitat layers (DEM and Forest Cover). This data set is large and carries inherent noise due to field data and a complex landscape; therefore it well covers the extremes of the fitted link functions. It is a representative example for a situation where the selection of a link function could affect the results. Findings indicate that for our data all three link functions behave similar, but logit link functions perform better than the cloclog and probit link functions when inferences as well as predictions are the study goals.

1 Introduction

Multiple regression techniques are widely used to assess wildlife-habitat relationships. Frequently, binary data are modeled with a variety of predictors, e.g. in a context of a Generalized Linear Model [1]. Such approaches mostly use the logit link function. Alternative link functions exist but are rarely considered and assessed. This situation is likely due to the general belief that the logit link function addresses the needs for most applications and thus is widely recommended [2]. The alternatives are seldom applied and tested although they could affect inferences and predictions in applications with larger data sets and which are distributed along the full gradient of the fitted link functions, e.g. at the extremes. The link functions are known to perform similar at the center, but differ at the lower and upper extremes. Besides the traditional logit link function, here we investigate the use of two alternatives, probit and cloglog, which has never been done for applications dealing with a large dat set of radio-telemetry wildlife data. Using field research data we present approaches for selecting the most appropriate link function when the emphasize is on inference

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 43-48, 2003.
© Springer-Verlag Berlin Heidelberg 2003

and/or prediction. We use nesting and habitat data from the Marbled Murrelet (*Brachyramphus marmoratus*), a tree-nesting seabird species of international conservation concern, to investigate the effects that the correct link function has for large-scale wildlife conservation and landscape management decisions.

1.1. Background

A GLM with a logit link function assumes that data are truly binomial [3]. The fit of such functions can be assessed with a Hosmer-Lemeshow test [4,5]. However, such tests can be potentially biased and their validity is debated [5,6]; the true distribution remains unknown. A wide choice of alternative link functions exist, and besides logit here we use two other functions that are commonly suggested [1, 2, 5] and which can relatively easily be applied. The probit link function is based on the inverse normal function [7]. In contrast to the two previous link functions, the complementary log-log (cloglog) link function is not symmetric around p=0.5 [3]; this function is the inverse of the cumulative extreme-value function (also called Gompertz distribution). A forth link function is the log log function, but it is seldom used because its behaviour is inappropriate for most applications [1]; thus, it was excluded from this analysis. For more details on link function characteristics and relations see [1,3,7,8]. Our approach is based on existing SPLUS code [9], [2, 10], which we modified accordingly to address the outlined research questions.

2 Methods

Data. We used the existing data set for nesting locations of Marbled Murrelets in Desolation Sound, British Columbia [9]. Nesting locations were derived from aerial telemetry, and then overlaid in a GIS with a Digital Elevation Model (DEM) and with Forest Cover polygons [10]. Besides nest locations, this allows to derive random locations within available and comparable habitat (old-growth forest polygons) and to obtain a Resource Selection Function (RSF) [11].

Computations. We developed an SPLUS script, which allows, in an automated fashion, to compare 51 nest locations in old-growth forest with a set of available locations, randomly drawn from an overall pool of 5000 random locations (compare also with [9]). To these data, we applied a binomial GLM with slope and elevation as predictors, and used each time a different set of link functions, logit, probit and cloglog. Results for each GLM were stored externally and ready for external analysis as an ASCII file.

Assessing the link functions. The mean coefficients derived from the 1000 comparisons were used to derive a regression formula for each individual link function. The standard deviations (SD) for the coefficients were also computed. We decided to use 1000 comparisons since it is suggested to use an ‘infinite’ number of comparisons, and the findings don’t show a change beyond 1000 comparisons [11]. If the focus is on inference, we used the mean and standard deviation of the model

deviance for the 1000 models as a criterion for the fit (first criterion). Using the derived 'mean' regression formula for each of the three link function results, data were predicted back to themselves (training data) using formulas (1), (2) and (3). For the second criterion with an emphasize on prediction, we present the descriptive statistics of the predictions for the known absences and presences. Alternatively, the predictions could also be presented spatially in order to assess visually the impacts that the use of the individual link functions made on the predicted spatial distribution of nests.

$$\exp(\alpha+\beta1*\text{slope}+\beta2*\text{elevation})/1+\exp(\alpha+\beta1*\text{slope}+\beta2*\text{elevation}) = P \quad \textbf{(1 logit)}$$

$$1/(\text{sqrt}(2*1\text{^}2*3.141569))*\exp(-(\alpha+\beta1*\text{slope}+\beta2*\text{elevation})\text{^}2/2*1\text{^}2) = P \quad \textbf{(2 probit)}$$

$$1-\exp(-\exp(\alpha+\beta1*\text{slope}+\beta2*\text{elevation})) = P \quad \textbf{(3 cloglog)}$$

3 Results

The three mean regression formulas in table 1 show that the link functions affect the coefficients and the deviances of the fitted models. Using the first suggested criterion (emphasize on inference), the highest standard deviation for the model deviance is found for the link function clogloc and thus performs the least reliable; the model deviances for logit and probit behave identical. The intercept and the slope coefficients vary for each of the link functions, but they remain within their positive and negative ranges. The elevation coefficient is identical for the logit and probit link function, but differs for the clolog link function. Generally, the slope coefficients have a slightly higher standard deviation than the coefficients for elevation.

Table 1: Mean regression parameters for three link functions.

	Mean Intercept (SD)	**Mean Slope Coefficient (SD)**	**Mean Elevation Coefficient (SD)**	**Mean Model Deviance (SD)**
Logit	-0.56346 (0.31362)	0.16687 (0.06877)	-0.08149 (0.03118)	189.9673 (2.25170)
Probit	-0.34014 (0.19472)	0.09836 (0.04143)	-0.08149 (0.03118)	189.9673 (2.25170)
Cloglog	-0.84116 (0.23765)	0.14861 (0.05758)	-0.06659 (0.02417)	189.5573 (5.05914)

For the second suggested criterion (emphasize on prediction), depending on the link function used the predictions deviate from the known and expected presence and absence of the training data (table 2). The findings show a consistent trend and behavior among link functions in that presence predictions are higher than absence predictions. Logit and cloglog prediction values are on the same magnitude, but the probit link function generally has lower predicted values and does not indicate a truly statistical difference between predicted absence and presence, as the other two link functions indicate. The probit function has also inconsistent SDs (lowest for predicted absences and highest for predicted presences); it appears to perform poorly and as an 'outlier'. It should be kept in mind that these comparisons are based on different sample sizes for predicted absences (n=5000) and presences (n=51). Generally, the differences between predicted absences and predicted presences are relatively small and do not even reach one; this is likely due to the inherent noise within field- and GIS data for a very heterogeneous landscape. The predictions are biased towards lower prediction values. Using the model deviance as a decision criterion, the findings indicate that for the Marbled Murrelet nesting habitat data example the logit and probit link functions would fit the data best. Using the predictive performance as a criterion, the logit and cloglog function behaves best. Overall, it can be concluded that for the used data set, the logit link function performs well overall, but less so the cloglog and probit functions.

Table 2: Predictions for each of the link functions on training data (known presence and known absence)

	Logit		**Probit**		**Cloglog**	
Predictions	Known Absence n=5000	Known Presence n=51	Known Absence n=5000	Known Presence n=51	Known Absence n=5000	Known Presence n=51
Min	0.14358	0.20127	0.07340	0.12815	0.14684	0.19873
Mean	0.32648	0.35437	0.25164	0.27289	0.32575	0.35385
Median	0.32683	0.34764	0.25190	0.27892	0.31839	0.34891
Max	0.56826	0.60865	0.39758	0.39807	0.60173	0.65635
SD	0.07239	0.08732	0.06316	0.69907	0.07459	0.09423
LCI	0.32447	0.32981	0.24988	0.25322	0.32368	0.32935
UCI	0.32848	0.37893	0.25339	0.29255	0.32781	0.38236

4 Discussion

We investigated the effect of applying different link functions to the fitting of binary data in a GLM. Using the correct link function is important for inference and predictions, e. g. for Resource Selection Functions [11]. We use empirical field-based telemetry-and GIS-data which present a real-life example and which are crucial to investigate towards sound conservation and management decisions, based on true data. We used two methods to assess the quality of the link function: standard

deviation of the model deviance, and predictive accuracy. We did not assess how stepwise approaches (adding or dropping) for the individual predictors to the model vary with the link function since we believe that this approach is not providing relevant evidence for the selection of the most appropriate link function. For the data used, it was found that the smallest standard deviation of the model deviance occurred with the logit and probit link functions, suggesting either of these two functions to be used to model these binary data. For the predictions, we found that the logit and clogclog link functions performed best. Overall, our findings are in agreement with the assumption that the traditionally used logit function performs well overall [12.]. However, depending on the specific data this can change. Although our findings are pointing towards consistent results for the two assessment methods, we emphasize the conflict between using 'tight' regression coefficients vs. minimizing SD of predictions to select the best link function for a specific study and data set. Which of these two criteria are to be used depends on the overall context and goals of the individual study. Therefore, it is crucial that the project goals are specified in advance, e.g. (i) inference, (ii) prediction or (iii) both, so that the best link function is selected for the specific research purpose. In the absence of truly accepted fitting diagnostics and tests, our findings and approaches demonstrate an example of selecting the appropriate link functions for solid and sound inferences and predictions when using binary GLMs. Towards sound inference and predictions, it is recommended that an assessment as presented here be done for any application which uses binary data in a multiple regression and for an RSF approach, e.g. as the case for many modern GIS applications based on large data sets with inherent 'noise' from field research in complex landscapes.

References

1. McCullagh, P. and J. A. Nelder. Generalized Linear Models. Monographs on Statistics and Applied Probability 37. Chapman and Hall. (1989)

2. Venables, B. and B.D. Ripley. Modern Applied Statistics with S. Fourth Edition. Springer Verlag, New York. (2002)

3. Collet, D. Modelling Binary Data. Chapman & Hall. New York. (1991)

4. Hosmer, D.W. and S. Lemeshow. Goodness-of-fit tests for the multiple logistic regression model. Communications in Statistics – Theory and Methods: (1980) 1043-1068.

5. Hosmer, D.W. and S. Lemeshow. Applied Logistic Regression. Wiley & Sons (1989).

6. Harrell, F. E. Jr. Regression Modeling Strategies. Springer Series in Statistics. Springer-Verlag. New York. (2002)

7. Daganzo, C. Multinomial Probit: The Theory and its Application to Demand Forecasting. Economic Theory, Econometrics, and Mathematical Economics. Academic Press. (1979)

8. Menard. S. Applied Logistic Regression Analysis. Sage Publications (2001)

9. Mathsoft Inc. SPLUS. Professional Release 2. Seattle (2000)

10. Huettman, F. and Linke J. An automated method to derive habitat preferences of wildlife in GIS and telemetry studies: A flexible software tool and examples of its application. European Journal for Wildlife Research. (in press)

11. Huettmann, F., E. Cam, R.W. Bradley, L. Lougheed, L.M. Tranquilla, C., Lougheed, P. Yen, Y. Zharikov and F. Cooke. Breeding habitat selectivity by Marbled Murrelets in a fragmented old-growth forest landscape. Wildlife Monograph. (in review)

12. Manly, B.F., L. L. McDonald, D. L. Thomas, T. L. McDonald and W. P. Erickson. Resource Selections by Animals. Kluwer Academic Publishers, Netherlands. (2002)

Improvements to the Helpful-Set Algorithm and a New Evaluation Scheme for Graph-Partitioners*

Stefan Schamberger

Fakultät für Elektrotechnik, Informatik und Mathematik
Universität Paderborn
Fürstenallee 11, D-33102 Paderborn
schaum@uni-paderborn.de

Abstract. Graph partitioning is an important subproblem in many applications. To solve it efficiently, the multilevel strategy in combination with a matching algorithm and a local refinement heuristic has proven to be a powerful method, and several libraries exist providing such an implementation. Due to the large involvement of heuristics, the evaluation of these libraries is usually based on experiments. In this paper we show that single experiments are usually not sufficient to judge the quality of an algorithm, since even results obtained for graphs of and identical structure show high variations. This is still true, even if the applied algorithms do not contain any nondeterminism. We propose a scheme that considers these variations and therefore makes evaluations and comparisons of different implementations more meaningful. We have applied this technique to evaluate the improvements of the Helpful-Set 2-partitioning implementation and present the obtained results.

1 Introduction

The graph partitioning problem is well known and occurs as a subproblem in many important applications. Given a graph, the task is to assign a graph's vertices to one of the equal sized partitions, such that the number of edges connecting vertices of different partitions is minimized. This paper deals with the 2-partitioning problem of unweighted graphs.

Since the graph partitioning problem often only represents a subproblem, it has to be solved fast and as space-efficient as possible. Due to the size of the graphs, state-of-the-art graph partitioning libraries like Metis [6], Jostle [8], Chaco [4] or Party [1] usually follow the multilevel scheme [4]. Vertices of the graph are contracted and a new level consisting of a smaller graph with a similar structure is generated. This is repeated, until in the lowest level only a small graph, sometimes only with 2 vertices, remains. The partitioning problem is then solved for this small graph and vertices in higher levels are assigned to partitions

* This work was partly supported by the German Science Foundation (DFG) project SFB-376.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 49–59, 2003.
© Springer-Verlag Berlin Heidelberg 2003

according to their representatives in lower levels, after a local refinement phase has been applied to further enhance the current solution. This process finally leads to a partitioning of the original graph. Hence, a multilevel algorithm consists of three important tasks: A matching algorithm, deciding which vertices are combined in the next level, a global partitioning algorithm applied in the lowest level (which actually can be omitted if the number of vertices in the lowest level meets the number of desired partitions), and a local refinement algorithm improving the quality of a given partitioning.

Due to speed requirements, all of the three described tasks usually involve some kind of heuristics. In most cases for example, the local refinement process is based on the Fiduccia-Mattheyses method [3], a run-time optimized version of the Kerninghan-Lin (KL) heuristic [7]. Even if approximation algorithms are applied to one of the tasks, the overall calculation can still be seen as a heuristic, since their interaction has not been theoretically investigated yet.

The high involvement of heuristics leads to the fact that partitioning libraries are usually evaluated by performing experiments on a set of selected graphs. These results are seen as an indication of how good the library would perform on similar graphs. However, in [2] it is shown that the calculations of the graph partitioning libraries are not always deterministic. More precisely, it has been shown that different initializations of the random number generator, used in some parts of the implementations, lead to deviations in the solutions quality. Thus, the question arises if a reliable prediction of the partitionings' quality can be made performing only one experiment on each graph, as it has been done in many comparisons so far.

This paper has two goals. First we demonstrate that results calculated by partitioning libraries are not totally predictable, even if the applied algorithms themselves do not include any nondeterminism. We do this by generating graphs of an identical structure and show, that the obtained partitionings vary a lot in their quality. This observation leads to a proposal of a new evaluation scheme. Second, we show that it is possible to further improve the local refinement Helpful-Set algorithm presented in [1]. During the development, our evaluation scheme has thereby shown to be quite useful, helping us to enlarge the reliability of the algorithm. Results of experiments performed with it are presented and finally we also test Party against state-of-the-art partitioning libraries using the new evaluation scheme.

The remaining part of this paper is organized as follows. In the next Section we describe why and how we randomize a graph, and present a new evaluation scheme for graph partitioning libraries. In Section 3, a very brief overview to the Helpful-Set heuristic and its improvements is given. Results obtained with it are presented in Section 4, showing the enhancements of Party and tests against two other state-of-the-art partitioning libraries, Jostle and Metis. In Section 5 we give a brief conclusion.

2 A New Evaluation Scheme

Using a fixed set of graphs to test a partitioning library includes some drawbacks. Even if not intended, it is easy to adopt an algorithm to the test set, meaning that the results become better for the selected graphs but deteriorate for others. During the development of the Helpful-Set heuristic, we have experienced this several times. To overcome this problem, we first enlarged the test set (we actually used many more graphs than presented in this paper). This makes the undesired adoption more unlikely, but also increases the testing time a lot. Another approach we found is based on permutation. It should be mentioned, that generating random graphs is no solution to the problem since their structure and their properties are completely different to those of 'real world' problems. However, a graph can be partly randomized keeping its structure. This is done by permuting its vertices.

Figure 1 shows an example of how this permutation works. First, a random permutation σ of numbers 0 to $|V|-1$ is determined. σ is then used to map each vertex v of the original graph G to vertex $\sigma(v)$ in the new graph G'. Next, the newly generated sequence of G's vertex numbers is sorted, changing the vertices' order. After that, the edges are adopted to the new graph by transforming an edge (v_i, v_j) into edge $(v_{\sigma(i)}, v_{\sigma(j)})$. Last, the outgoing edges of each vertex are sorted according to their destination's vertex number. It is obvious that the remaining graph G' has exactly the same structure and properties than the original G. However, in our experiments it is shown that the influence of the permutations on the 2-partitioning results is surprisingly high.

To make use out of this observation, we created a new evaluation scheme, based on the described technique. For each graph from our test set, we performed 100 runs. The first run consists in partitioning the original, unchanged graph, while for all following runs the graphs vertices are permuted. We are aware that this scheme does not solve the evaluation problem completely since an algorithm can still be tailored for special graph types. But at least it makes this more difficult for single test graphs that now represent a class of graphs. Last but not least, the method delivers some data about the variation of the solutions' quality.

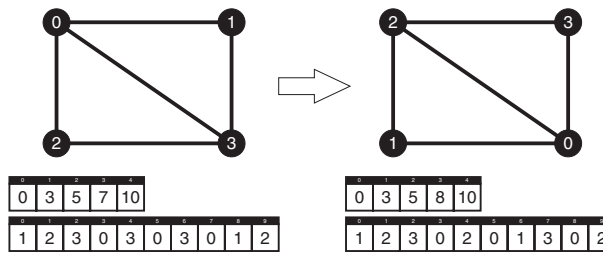


Fig. 1. Graph permutation. The original graph and its data structure (left) is transformed by the permutation $\sigma = (2, 3, 1, 0)$ into the new graph (right).

One can now determine if an algorithm does always find solutions of about the same quality or if they highly differ to each other, supplying a measurement for reliability. Thus we believe that the permutation makes comparisons more meaningful.

All results obtained in the experiments are displayed in a chart, generated by a script. Figure 2 gives an example obtained 2-partitioning the 100x100 grid, which we have chosen because optimal solutions (with edge cut 100) are known. The left part shows the bisection quality. The edge cut is shown on the x-axis while the y-axis displays the weight of the largest partition (pw_{max}), that is the balance. Every mark of each type represents the result of one of 100 runs applying one of the libraries, respectively. Furthermore, the first run with the genuine, unchanged graph is represented by a solid mark, while the average of all 100 runs is displayed by a large solid mark. The right part shows the required resources. Since these do not depend much on the chosen node permutation, only the average time and memory consumption from all runs for this graph is displayed.

In anticipation of Section 4, we now describe how the chart shown in Figure 2 can be interpreted. The yellow, orange and red squares display results obtained using Jostle with 0%, 1% and 3% allowed imabalce, respectively. One can see that the average edge cut computed gets smaller with a higher imbalance allowance. The average solution for 3% allowed imbalance is of a good quality (103.4), and the figure reveals that there is some variation due to the randomization. This variation is even higher for Metis, which results are displayed with the upper light and lower dark blue triangles for pmetis and kmetis, respectively. Especially

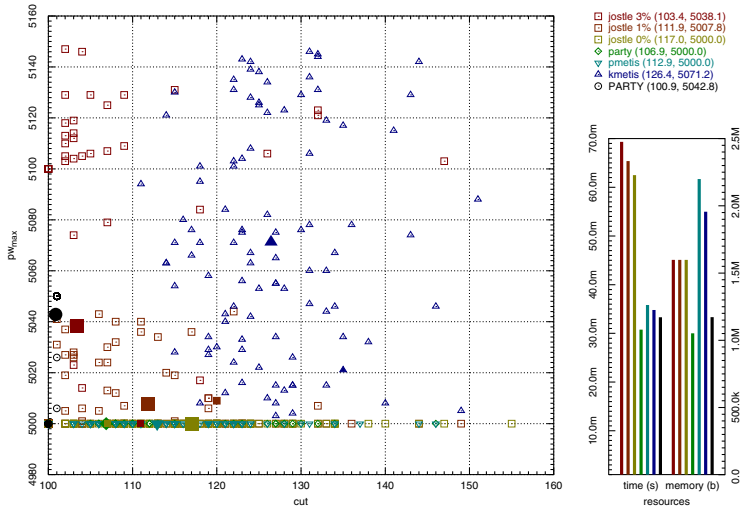


Fig. 2. Results of a 2-partitioning of a 100x100 2D grid

for kmetis achieving an average edge cut of 126.4 it is very high, both for the edge cut and for the balance. While the results obtained by pmetis, Jostle with no imbalance and the former implementation of Party (green diamonds) show a similar behavior in the edge cut, the new implementation of Party (black circles) does compute less divergent (often identical) solutions here, reaching an average edge cut of 100.9. On the right hand side the resources consumed by the different libraries are displayed. Since the variation here has shown to be negligible, only the average value is presented. In case of the 100x100 grid, Jostle needs longest to compute its results, followed by Metis and Party, while Metis needs the largest amount of memory, followed by Jostle and Party.

Another important point that can be observed is that if the libraries had been compared with only using the original graph, their rating would have been different. Metis (kmetis and pmetis) and Jostle with 1% and 3% imbalance allowance would have perfomed worse while Jostle with no imbalance and Party would have come off better, resulting in different conclusions.

3 Improvements to the Helpful-Set Heuristic

This Section is meant to give an idea about the functioning the Helpful-Set algorithm and the improvements we made to its implementation presented in [1]. The Helpful-Set (HS) Heuristic for local refinement is based on theoretic observations used to find upper bounds for the bisection width of regular graphs [5]. It has been implemented and presented in [1] and serves in the Party library as an alternative to Kerninghan-Lin style heuristics. Just as KL, the Helpful-Set heuristic is based on local search. Beginning with a given initial bisection π, it tries to reduce the edge cut with the help of local rearrangements. However, their choice is the main difference to KL since it does not only migrate single vertices but sets of vertices. The algorithm starts to search for l-helpful sets, that is a subset of nodes from either V_0 or V_1 decreasing the edge cut by l if moved to the other partition. If such a set is found, it is moved and the algorithm then tries to find a balancing set that eliminates the caused imbalance but also does not increase the edge cut by more than $l - 1$. If such a set can be found it is moved, resulting in an edge cut reduction of at least 1, and the whole process is repeated until no more improvements can be made. Figure 3 shows an example of a successful round reducing the edge cut from 9 to 7.

Due to runtime concerns, the search for Helpful-Sets can only be performed using a simple greedy algorithm always adding the current best vertex to the set. This approach uses the 'bucket' data structures known from the KL improvement [3]. It has turned out however, that the crucial point when implementing the Helpful-Set algorithm is the choice of the helpfulness value l of the sets to be searched for. If l is chosen too small, then promising sets are overlooked, while setting l too large quickly increases the runtime but usually does not discover better sets. The former implementation in Party therefore uses a technique called 'adaptive limitation' [1] to determine l. Setting it to $cut/2$ initially, l is halved

or set to the best occurring helpfulness if a l-helpful set can not be found in any of the two sets and doubled if the search has succeeded.

The improvements we made can be characterized into three aspects. First, instead of using only one limit value l for both sets, we introduce a separate one for each. This also leads to some changes in the 'adaptive limitation'. Second, we included the possibility of having slight imbalances. This feature has also been implemented in Jostle and Metis, and it has proven to be quite successful since slight imbalances often make better edge cuts possible. Especially in deeper levels of the multilevel approach it is not necessary (often not even possible) to completely balance the two sets. Third, the former implicit balancing mechanism (that is modifying the weight of the balancing sets) has been moved to the end of the algorithm and is performed explicitly, now always ensuring the requested bounds and increasing the reliability of the library.

4 Experimental Results

To evaluate the changes to the HS heuristic described in Section 3, we have implemented it (denoted as 'Party' and tested it against the former implementation (denoted as 'party 2.001') and two other well known graph partitioning libraries, Jostle (version 2.3) and Metis (version 4.0.1). Both of the latter are also based on the multilevel paradigm and apply a Kerninghan-Lin like refinement algorithm.

Throughout this paper we used a set of well known graphs for our comparisons. Most of the graphs have also been used in [8], comparing Jostle and Metis. Since we do not have access to the graphs 'mesh1m', 'oliker' and 'bmw1c', we replaced these two graphs by the 'airfoil1', 'biplane9', 'grid100', 'stufe10' and 'wave' graphs. Table 1 gives an overview of the test set and some of the graph's properties. Most of these graphs are FEM graphs, 2-dimensional as well as 3-

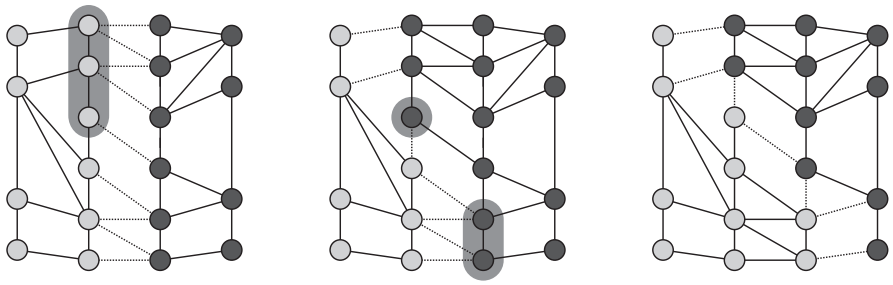


Fig. 3. Moving the marked set (left) reduces the current edge cut of 9 by 2 (thus the set is called 2-helpful), but it destroys the balance by 3. However, by moving the second marked set (middle), this can be fixed (thus this set is called a balancing set), and a balanced solution (right) is found leading to an overall edge cut reduction of 2.

dimensional, while some others represent the structure of sparse matrices from different sources.

As mentioned, a slight imbalance allowance does often lead to much smaller edge cuts. If an imbalance of $i\%$, $0 \leq i \leq 100$ is required, a partition's weight may not exceed $\lceil |V|/2 \cdot (100+i)/100 \rceil$. Jostle accepts an imbalance parameter and we used values of 3% (default), 1% and 0% of allowed imbalance. In the present version, Metis does not allow to adjust this parameter, instead it is set to constant 0% and 3% for the two versions pmetis and kmetis, respectively. While version 2.001 and earlier of the party library cannot handle imbalances, this feature has been added in the new implementation. All experiments have been performed on a Pentium III 933 MHz dual processor system with 1 GB of main memory using one processor.

We now describe the results we obtained by applying the new algorithm to the set of graphs listed in Table 1. First, we compare the improved implementation

Graph	$\|V\|$	$\|E\|$	min. deg.	av. deg.	max. deg.	diameter	origin
144	144649	1074393	4	14.86	26	35	FEM 3D
4elt	15606	45878	3	5.88	10	67	FEM 2D
airfoil1	4253	12289	3	5.78	9	43	FEM 2D
biplane.9	21701	42038	2	3.87	4	102	FEM 2D
crack	10240	30380	3	5.93	9	84	FEM 2D
dime20	224843	336024	2	2.99	3	1105	FEM 2D dual
grid100	10000	19800	2	3.96	4	198	FEM 2D
m14b	214765	1679018	4	15.64	40	33	FEM 3D
memplus	17758	54196	1	6.10	573	10	digital memory circuit
ocean	143437	409593	1	5.71	6	229	FEM 3D dual
stufe.10	24010	46414	2	3.87	4	54	FEM 2D
t60k	60005	89440	2	2.98	3	495	FEM 2D dual
vibrobox	12328	165250	8	26.81	120	3	vibroacoustic matrix
wave	156317	1059331	3	13.55	44	38	FEM 3D

Table 1. Graphs used in this paper and some of their properties.

Graph	party 2.001			Party									
	imbalance=0%			imbalance=0%		imbalance=1%			imbalance=3%			all
	edge cut	time	memory	edge cut	time	edge cut	time	pw_{max}	edge cut	time	pw_{max}	memory
144	6963.1	**1.949**	45741.8	6730.2	2.000	**6682.9**	2.200	72844.6	6814.1	2.318	74152.6	**38744.7**
4elt	167.6	**0.054**	2008.9	**148.6**	0.060	149.7	0.061	7854.9	149.4	0.061	7925.5	**1855.9**
airfoil1	84.3	**0.011**	537.1	81.5	0.014	78.8	0.014	2139.7	**77.5**	0.014	2174.6	**518.4**
biplane.9	90.5	**0.074**	**2186.9**	79.9	0.075	78.7	0.076	10917.8	**77.4**	0.075	11043.7	2494.1
crack	203.3	**0.035**	**1348.1**	192.5	0.046	192.2	0.046	5155.4	**191.3**	0.046	5221.1	1473.1
dime20	213.4	**1.227**	**18678.6**	**185.8**	1.285	189.5	1.282	113058.4	190.0	1.287	114377.9	23215.9
grid100	106.9	**0.031**	**1027.0**	**100.2**	0.034	100.9	0.034	5042.8	100.9	0.034	5141.0	1144.2
memplus	5858.7	**0.161**	**4620.6**	5791.4	0.372	5744.5	0.384	8965.9	**5642.3**	0.374	9139.8	5774.8
m14b	4127.0	**3.337**	71679.4	**3956.8**	3.602	4004.0	3.748	108236.1	4081.8	3.810	110064.4	**61310.7**
ocean	551.1	**1.204**	**22517.8**	496.1	1.262	505.9	1.287	72142.9	**391.6**	1.277	72932.5	26266.7
stufe.10	62.2	**0.079**	**2435.8**	52.0	0.083	51.3	0.083	12073.2	**51.0**	0.083	12204.9	2780.9
t60k	100.3	0.241	**5152.4**	91.9	**0.234**	88.7	0.235	30218.4	**79.4**	0.235	30836.2	6325.5
vibrobox	11974.8	**0.160**	7320.1	11711.7	0.287	**11691.8**	0.368	6191.4	11730.0	0.433	6269.4	**5576.5**
wave	9445.8	**2.051**	46399.3	9104.9	2.234	9070.8	2.388	78797.0	**9049.6**	2.471	80052.5	**40790.6**

Table 2. Comparison of party 2.001 and Party. The best average result is printed bold while results not within 10% (edge cut) and 25% (time and memory) of this value are shaded gray.

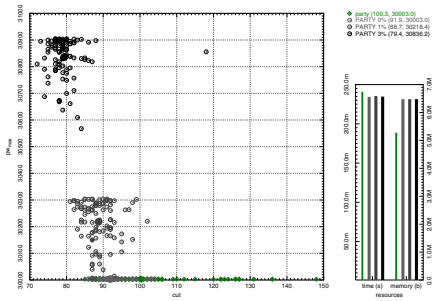


Fig. 4. 2-partitioning of 't60k'

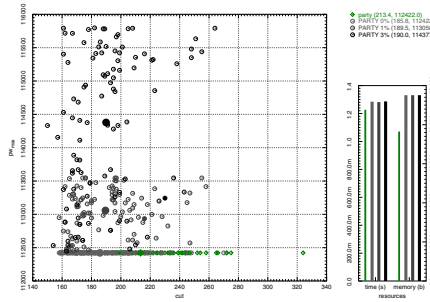


Fig. 5. 2-partitioning of 'dime20'

of the Helpful-Set method with the former one. Due to space limitations, we only present two of the charts and summarize results obtained for all graphs in Table 2. It shows that in all tested cases the new implementation achieves a smaller average edge cut even if no imbalance is allowed, while the time and memory requirements only slightly increase. If imbalance is added, the quality of the results can be further improved in some cases as shown for example for the 't60k' graph in Figure 4. For other graphs however, the opposite can be observed. One example is the 'dime20' graph, which is shown in more detail in Figure 5. There, the average edge cut increases from 185.0 in case of no allowed imbalance to 190.0 with 3% imbalance. We found this quite surprising since one does expect equal or better solutions by increasing the feasible set size. However, with this increase the search for the best solution becomes more difficult, since more, probably worse, local optima exist. On the other hand, Figure 5 also reveals that the cut size of the best solution found during all 100 runs is smaller for 3% imbalance than in the 1% or 0% case. Note, that the decrease would have been much higher if no permutation had been applied. With it, the results are acceptable. Additionally, the solutions found vary more in case of the 't60k' graph than for the 'dime20', indicating that Party has more difficulties in the latter case due to the combination of the graph's structure and the applied heuristics.

Next, we show the results we obtained comparing the new implementation of the Helpful-Set heuristic to the state-of-the-art partitioning libraries Jostle and Metis. Tables 3 and 4 give an overview to the results obtained for all graphs from Table 1. Due to space limitations and better comparability we only include results for Party obtained with 3% imbalance allowance. From Table 3 one can see that Party often computes the best average edge cut, followed by Jostle (allowing 3% imbalance) and pmetis (allowing no imbalance). Furthermore, Party seems to make more use of the freedom granted by the imbalance setting than Jostle or Metis. However, also allowing only 1% imbalance results in a similar picture (omitted here, see Table 2) with Party computing the best average results.

The resource requirements are displayed in Table 4. While Party always consumes least memory, Metis (kmetis and pmetis) are fastest. In some cases, Party

can compete with Metis, while in some others it is slightly slower than Jostle. This high difference between the graphs is probably due to problems inside the matching strategy, resulting in constructing many unnecessary levels and causing lots of unsuccessful searches. Hence, this matter requires further work.

For two graphs the results included in Tables 3 and 4 are presented in more detail in Figures 6 and 7. As mentioned earlier, in case of the 'dime20' the edge cut computed by Party does not increase with allowed imbalance, but is still good compared to the other libraries. Again, this is also a good example for a different ranking mentioned in Section 2 that would have been obtained without the permutation approach. Figure 7 is an example where Party performs slowly. In contrast to the 'm14b' where this time is more or less wasted, it turns into account for the 'memplus' graph finding much better edge cuts.

Concerning the influence of the permutations, the most interesting information is presented in Table 5. It shows the standard deviation of the computed solutions' quality (edge cut and balance) for each library, respectively. First, it is evident that permuting the graphs' vertices has a large impact on the results computed by all libraries. Furthermore, one can see that the variation of the

Graph	Jostle 3%		Jostle 1%		Jostle 0%		pmetis		kmetis		PARTY
	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}	edge cut
144	7030.4	73263.4	7040.0	72765.6	7047.8	72325.0	6855.9	72325.0	6989.6	72642.1	**6814.1**
4elt	162.8	7911.7	164.7	7836.7	169.9	7803.0	160.5	7803.0	169.0	7888.2	**149.4**
airfoil1	84.0	2162.2	86.6	2134.4	88.5	2127.0	84.7	2127.0	88.5	2168.9	**77.5**
biplane9	99.4	10993.3	101.1	10903.9	105.0	10851.0	84.8	10851.0	89.2	10949.4	**77.4**
crack	201.4	5185.3	203.7	5137.0	205.8	5120.0	199.8	5120.0	210.8	5188.9	**191.3**
dime20	224.3	113713.1	228.5	112942.7	243.3	112422.0	192.0	112424.9	194.1	112598.3	**190.0**
grid100	103.4	5038.1	111.9	5007.8	117.0	5000.0	112.9	5000.0	126.4	5071.2	**100.9**
memplus	6092.3	9085.6	6140.5	8966.5	6183.7	8879.0	6555.6	8879.2	6583.8	9126.8	**5642.3**
m14b	4203.5	108592.4	4206.3	107916.2	4221.7	107383.0	**4056.9**	107383.0	4145.0	107573.8	4081.8
ocean	438.6	73078.1	500.1	72319.8	564.4	71719.0	504.5	71722.7	512.9	72108.5	**391.6**
stufe10	62.5	12118.0	62.8	12052.3	64.8	12005.0	59.0	12005.6	61.7	12092.0	**51.0**
t60k	92.8	30469.2	97.1	30129.8	101.4	30003.0	96.5	30009.9	105.5	30192.2	**79.4**
vibrobox	11821.1	6320.0	11858.3	6218.7	11802.8	6164.0	11792.4	6164.1	11857.4	6297.5	**11730.0**
wave	9410.1	79262.9	9416.5	78628.5	9404.2	78159.0	9381.0	78159.0	9441.4	78539.7	**9049.6**

Table 3. Quality of 2-partitionings computed by Jostle, Metis and Party (with 3% imbalance allowance). The smallest average edge cut is printed bold while results not within 10% of this value are shaded gray.

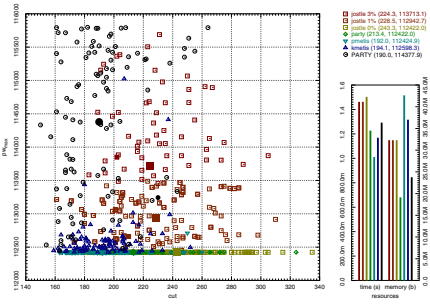


Fig. 6. 2-partitioning of 'dime20'

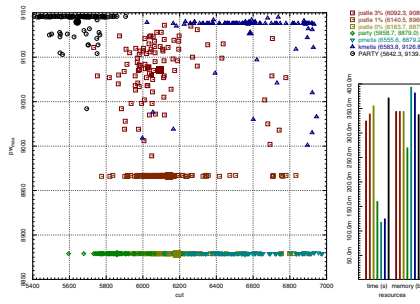


Fig. 7. 2-partitioning of 'memplus'

Graph	Jostle 3%		Jostle 1%		Jostle 0%		pmetis		kmetis		PARTY	
	time	memory	time	memory	time	memory	time	memory	time	memory	time	memory
144	2.180	47072.7	2.186	47072.7	2.257	47072.7	**1.321**	73596.3	1.515	70080.5	2.332	**38744.7**
4elt	0.115	2721.4	0.113	2721.4	0.113	2721.4	0.060	3969.8	0.062	3596.9	**0.058**	**1855.9**
airfoil1	0.024	739.0	0.024	739.0	0.028	739.0	0.016	1078.6	**0.014**	982.5	0.014	**518.4**
biplane9	0.145	3338.8	0.145	3338.8	0.148	3338.8	0.081	4581.5	0.083	4060.0	**0.072**	**2494.1**
crack	0.071	1794.5	0.069	1794.5	0.070	1794.5	0.043	2820.0	**0.042**	2580.5	0.045	**1473.1**
dime20	1.462	31531.6	1.463	31531.6	1.504	31531.6	**1.011**	41622.7	1.169	36157.1	1.292	**23215.9**
grid100	0.069	1560.4	0.065	1560.4	0.062	1560.4	0.036	2147.2	0.035	1910.7	**0.033**	**1144.2**
memplus	0.325	5887.4	0.340	5887.4	0.356	5887.4	**0.118**	6739.4	0.125	6535.7	0.372	**5774.8**
m14b	3.298	72498.3	3.298	72498.3	3.355	72498.3	**2.163**	114897.6	2.452	109672.5	3.817	**61310.7**
ocean	1.330	29339.7	1.334	29339.7	1.361	29339.7	**0.837**	40283.4	0.958	36795.2	1.274	**26266.7**
stufe10	0.148	3712.4	0.148	3712.4	0.151	3712.4	**0.084**	5084.7	0.090	4507.3	0.086	**2780.9**
t60k	0.369	8577.2	0.370	8577.2	0.382	8577.2	**0.225**	11276.4	0.253	9820.6	0.235	**6325.5**
vibrobox	0.402	6800.3	0.392	6800.3	0.417	6800.3	0.158	10640.1	**0.148**	10419.6	0.437	**5576.5**
wave	2.263	48204.8	2.267	48204.8	2.392	48204.8	**1.372**	74861.3	1.558	71069.1	2.473	**40790.6**

Table 4. Time (s) and memory (Kb) consumption of the computations. The best average result is printed bold while results not within 25% of this value are shaded gray.

Graph	Jostle 3%		Jostle 1%		Jostle 0%		pmetis		kmetis		PARTY	
	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}	edge cut	pw_{max}
144	541.6	575.1	540.7	184.6	536.2	0.0	211.2	0.0	236.1	261.5	151.6	331.1
4elt	26.4	72.7	27.3	25.5	27.2	0.0	23.4	0.0	19.6	60.3	15.4	65.5
airfoil1	8.3	19.6	8.1	5.6	8.7	0.0	7.0	0.0	8.3	19.3	5.0	13.4
biplane9	26.2	97.7	27.1	32.9	26.6	0.0	10.8	0.0	11.7	76.1	5.0	93.7
crack	14.4	44.6	15.1	15.2	15.5	0.0	14.2	0.0	10.9	45.6	6.7	50.2
dime20	29.5	784.1	30.8	306.6	33.3	0.0	20.3	28.8	18.2	341.3	24.6	1098.8
grid100	7.7	52.7	10.6	13.4	11.0	0.0	9.9	0.0	7.9	43.3	0.3	17.3
memplus	206.7	31.9	208.3	0.5	208.5	0.0	223.2	0.4	236.4	32.0	114.9	14.4
m14b	567.0	812.7	567.4	257.7	569.9	0.0	131.4	0.0	349.5	148.3	124.0	702.9
ocean	296.2	518.0	279.4	173.3	262.7	0.0	21.5	26.3	49.8	365.4	64.6	686.2
stufe10	22.0	97.4	21.7	33.5	22.0	0.0	16.9	6.1	18.5	82.5	5.6	116.1
t60k	14.1	222.1	12.6	86.3	12.4	0.0	9.8	40.2	49.9	224.7	4.9	74.3
vibrobox	426.2	47.2	406.9	12.3	380.7	0.0	319.1	0.4	367.1	55.0	207.9	40.4
wave	393.7	636.9	394.8	197.5	398.1	0.0	344.6	0.0	352.7	280.8	182.9	581.6

Table 5. Standard deviation of edge cut and balance occurring during 100 runs.

edge cut produced by Jostle is usually highest, followed by Metis, whereas Party more often calculates partitionings a bit more close together. There are still a few exceptions to this rule, like in the case of the 'm14b' and 'ocean' graphs. This might be an indication that the current implementation has some problems with these graph and that further investigation is required to solve this matter. Comparing the balance, it is more difficult to make any conclusions, since all libraries compute comparable results, the same imbalance setting implied. Furthermore, a high variation here on the one hand could be interpreted as an indicator how extensively the search space is explored. On the other hand, this could also point to problems in leaving local minima.

5 Conclusion

In this paper we have shown that the solutions computed by the graph partitioning libraries Jostle, Metis and Party are highly affected by a random permutation of the graphs' vertices. Since experiments and comparisons usually are thought as a kind of prediction of how good a heuristic performs on certain types of

graphs, we think that it is essential to consider this fact in this kind of experiments. Therefore, we have proposed a simple evaluation scheme, that is able to affect all, not only nondeterministic implementations.

Although our scheme does not overcome the evaluation problems caused by 'small' test sets, it has shown to be helpful in developing the improvements made to Party, saving time and pointing us to some difficulties producing high variations in the solutions' quality. Additionally, it provides information about how reliable a graph partitioning heuristic is and therefore indicates in what quality range solutions can be expected.

Concerning the improvement of Party's Helpful-Set algorithm, we were able to reduce the computed average edge cut by about 2% – 19% (no imbalance allowance) and 2% – 22% (3% imbalance allowance) for the graphs in our test set. Compared to state-of-the-art partitioning libraries, superior 2-partitionings can often be found in a comparable amount of time.

References

1. R. Diekmann, B. Monien, and R. Preis. Using helpful sets to improve graph bisections. In D.F. Hsu, A.L. Rosenberg, and D. Sotteau, editors, *Interconnection Networks and Mapping and Scheduling Parallel Computations*, volume 21 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 57–73. AMS, 1995.
2. U. Elsner. *Static and Dynamic Graph Partitioning.* PhD thesis, Technische Universität Chemniz, 2001.
3. C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *ACM IEEE Nineteenth Design Automation Conference Proceedings*, pages 175–181, Los Alamitos, Ca., USA, Jun 1982. IEEE Computer Society Press.
4. B. Hendrickson and R. Leland. A multi-level algorithm for partitioning graphs. In *Proceedings of Supercomputing'95*, San Diego, CA, Dec 1995. ACM/IEEE.
5. J. Hromkovic and B. Monien. The bisection problem for graphs of degree 4 (configuring transputer systems). In A. Tarlecki, editor, *Proceedings of Mathematical Foundations of Computer Science. (MFCS '91)*, volume 520 of *LNCS*, pages 211–220, Berlin, Germany, Sep 1991. Springer.
6. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, Jan 1999.
7. B. W. Kernighan and S. Lin. An efficient heuristic for partitioning graphs. *Bell Systems Technical Journal*, 49:291–308, Feb 1970.
8. C. Walshaw and M. Cross. Mesh partitioning: A multilevel balancing and refinement algorithm. *SIAM Journal on Scientific Computing*, 22(1):63–80, 2000.

Mesh Partitioners for Computational Grids: A Comparison

Sili Huang , Eric Aubanel, Virendrakumar C. Bhavsar

Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada, E3B 5A3
{q27ac, aubanel, bhavsar}@unb.ca

Abstract. Computational grids are emerging as a new infrastructure for high performance computing. Since the resources in a grid can be heterogeneous and distributed, mesh-based applications may require a mesh partitioner that considers both processor and network heterogeneity. We have developed a heterogeneous mesh partitioner, called PaGrid. PaGrid uses a multilevel graph partitioning approach, augmented by execution time load balancing in the final uncoarsening phase. We propose that estimated execution time of the application is a better metric for the performance of a mesh partitioner than communication cost. The main objective of this paper is to compare the performance of PaGrid with two existing mesh partitioners, METIS 4.0 and Jostle 3.0, for mapping applications to a heterogeneous computational grid. We have found that PaGrid produces comparable or better partitions than Jostle. Although the performance of METIS is worse, in terms of total communication cost, than PaGrid and Jostle, it is comparable to them in terms of the estimated execution time of the application.

1 Introduction

High performance distributed computing is rapidly becoming a reality. Applications that are either data intensive or computationally intensive can exploit a high performance distributed computing environment such as a grid. In computational science, there are many applications based on a three-dimensional (3D) mesh model, for example computational fluid dynamics and computational mechanics. Many such 3D applications require large amounts of processing time and are often executed on a parallel computer. Computational grids are attractive for large-scale problems in these and other areas. Such grids are usually heterogeneous in computing and communication resources, and the performance of these resources may vary greatly. Consequently, the successful deployment of parallel mesh-based applications in a grid environment must involve efficient mesh partitioning.

Given a mesh-based application and a computational grid, the mesh partitioning problem can be defined as follows. The mesh is modeled as a weighted graph $G = (V, E)$, consisting of V vertices and E edges [1]. The computational grid is modeled as a weighted processor graph $S = (P, C)$, consisting of a set, P, of processors connected by the set, C, of edges. The mesh partitioning problem is to map V onto P, that

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 60-68, 2003.
© Springer-Verlag Berlin Heidelberg 2003

is $\pi : V \to P$, such that a specified cost function is minimized. The cost function can be based on total interprocessor communication cost, and/or estimated execution time of the application.

Currently, grid core middleware such as Globus [5] and Legion [6] provides its own implementations of MPI [7], such as MPICH-G2 [9], to support message passing. However, it does not provide a service for data partitioning. On the other hand, existing partitioning tools may not be suitable for heterogeneous grids. There are many partitioning tools available for homogeneous environments, such as METIS [11]. There has been recent work on partitioning for heterogeneous environments, such as PART [2, 3], Jostle [15, 17, 18] and MiniMax [12]. In particular, Walshaw and Cross [17] recently studied the partitioning problem for a heterogeneous communication network. Another approach is used in SCOTCH, which involves recursive bisection of both the mesh and processor graphs [19].

In this context, we have developed a mesh partitioner, PaGrid [8], for partitioning unstructured meshes onto heterogeneous grids, taking consideration of heterogeneity in both network and processor performance. Section 2 briefly describes PaGrid. Subsequently, in Section 3, we compare the performance of PaGrid to METIS and Jostle for a set of test graphs mapped to a heterogeneous processor graph. Finally, we give concluding remarks in Section 4.

2 PaGrid

The PaGrid mesh partitioner makes use of multilevel graph partitioning to minimize the total cost of processor mapping. The graph partitioning proceeds in three phases: coarsening, initial partitioning and uncoarsening. Load balancing among processors is accomplished by a multilevel diffusion algorithm taking into account vertex weights and the estimated execution time in the final uncoarsening level. We use the cost function based on quadratic path length proposed by Walshaw and Cross [17].

During the coarsening phase, we use modified heavy edge matching proposed in METIS [11] and we iteratively coarsen the original graph down to where the number of vertices equals the given number of processors. During the initial partitioning phase, as in JOSTLE [16] but with a different quadratic assignment problem (QAP) solver [4], we assign each vertex in the coarsest graph to a different processor, while minimizing the total cost of mapping. During the uncoarsening phase, the coarsest graph is iteratively projected backward to the original graph with a multilevel refinement algorithm to minimize the cost of mapping. In each refinement stage, as in METIS [13] and JOSTLE [16], we perform vertex weight load balancing, which involves calculating the required load that should be transferred among processors. As in Jostle [16], Kernighan-Lin-like refinement is used to refine the partitions. The degree of imbalance allowed is largest for the coarsest graph, and it decreases as the graph is uncoarsened, down to 3% for the original graph, as in [17].

To reduce the total execution time for an application, it may be desirable to allow some load imbalance [14], depending on the computational and communicational characteristics of both the computing environment and the application. In PaGrid we assume that an estimate of the ratio of the communication time to computation time for a given application is known, and use this ratio to perform execution time load

balancing in the final uncoarsening stage to minimize estimated total execution time of the application.

We now give the details of our cost function. Let t_{comp}^{p} represent the computation time of processor p for processing a vertex that has the smallest weight in the application graph, and $t_{comm}^{(p,q)}$ denote the communication time for a vertex from processor p to processor q. Let π_q denote the set of vertices mapped to processor q, and let $|\pi_q|$ represent the total weight of these vertices. Note that this weight represents the total computational time for processor q, in units of the time required to process a vertex. Let $E_q(v)$ denote the set of edges from vertex v to vertices assigned to processor q, $E_q(v) = \{(v,u) \mid u \in \pi_q\}$, and $|E_q(v)|$ represent the sum of the weights of these edges; this sum represents the total communication time for processor q, in units of the time required to process a vertex.

For a processor p and one of its edges (p, q) in the processor graph, let $R(p,q) = t_{comm}^{(p,q)} / t_{comp}^{p}$. Further, let R_p represent the ratio of the minimum over all the edges in the processor graph of the communication time for a vertex over the computation time for a vertex in processor p, that is $R_p = \min_{p,q \in P, p \neq q} \{t_{comm}^{(p,q)}\} / t_{comp}^{p}$. The execution time t_p of processor p, in units of the time required to process a vertex, is given by:

$$t_p = |\pi_p| + \sum_{v \in \pi_p} \sum_{r \in P} |E_r(v)| R(p,r)$$

The execution time of the application is given by $t = \max_{p \in P} \{t_p\}$. This metric is used in the execution time load balancing stage of PaGrid. Moreover, we use it to compare the performance of various mesh partitioners. Note that owing to the complexity of the machines that are used (e.g. memory hierarchy, characteristics of interconnection networks) it is extremely difficult to devise an efficient metric. Further, our metric assumes that data corresponding to a single vertex is sent at a time. Our metric may be modified to take account grouping of communication for multiple vertices in a single message.

Another metric we consider is total communication cost given by $\Omega = \sum_{(u,v) \in E_c} |(u,v)| \cdot |(\pi(u), \pi(v))|$, where E_c denotes the set of all cut edges; each cut edge connects a pair of vertices assigned to different processors, $\pi(v)$ represents the processor to which vertex v is assigned in the mapping, $|(\pi(u), \pi(v))|$ represents the weight of the shortest path between $\pi(u)$ and $\pi(v)$, and $|(u, v)|$ denotes the weight of the edge (u, v).

3 Experimental Results

We evaluate the performance of PaGrid, METIS, and Jostle on six test graphs. The test graphs have been chosen from [20] to be representative of small to large-scale problems, including 2D and 3D node graphs and dual graphs. Since most mesh-based applications do not require weighting of the vertices and edges, none of these test graphs are weighted. The characteristics of these meshes are given in Table 1 in increasing order of graph size.

Table 1. Characteristics of the test graphs.

Graph	$\|V\|$	$\|E\|$	$\|E\| / \|V\|$	Description
3elt	4720	13722	2.91	2D finite element mesh
tri10k	10027	14806	1.48	-
crack	10240	30380	2.97	2D finite element mesh
t60k	60005	89440	1.49	-
wing	62032	121544	1.96	-
598a	110971	741934	6.69	3D finite element mesh

We consider two Grid systems in our experiments:

- Grid-1: heterogeneous communication network and identical processor weights (Figure 1).

The processor graph of Grid-1 consists of two clusters connected together through a slow link (p_5, p_6). All edges in Cluster 1 have identical weight equal to 1; all edges in Cluster 2 have weight equal to 2; the edge connecting the two clusters is assigned a weight of 4, that is $|(p_5, p_6)| = 4$. All processors are assigned an identical weight equal to 1. We choose identical communication to computation ratios R_p for all processors. We select R_p equal to 0.25 and 2.0 for all test application graphs. Note that the case where $R_p = 0.25$ represents coarser grained computation than the other case.

- Grid-2: heterogeneous communication network and processor weight varying by a factor of ten (Figure 1).

The processor graph of Grid-2 is the same as Grid-1, except that processors in Cluster 1 are assigned a weight equal to 1, while the processors in Cluster 2 are assigned a weight equal to 10. This means that the processors in Cluster 2 are ten times slower than the processors in Cluster 1.

In our implementation, a random number generator is used to guide the selection of vertices in both the matching stage in the coarsening phase and the refinement stage in the uncoarsening phase. During our extensive experiments [8], we observed that the choice of seed for the random number generator affects the partitioning results by 20% on average.

We ran PaGrid 200 times on the test graphs taking different random number seeds and compare our results with METIS and Jostle. For PaGrid we give the minimum and average of results over 200 trials, to illustrate the spread of results one might obtain. In practice, we expect that normally PaGrid would only be run once for an application, although more than one run could be used to select the best partition (at

the cost of extra time). METIS 4.0 includes pmetis and kmetis: pmetis is based on multilevel recursive bisection described in [10], and kmetis is based on multilevel k-way partitioning described in [11]. Jostle 3.0 is based on multilevel partitioning described in [15, 17]. Note that the results with METIS and Jostle are for only one run.

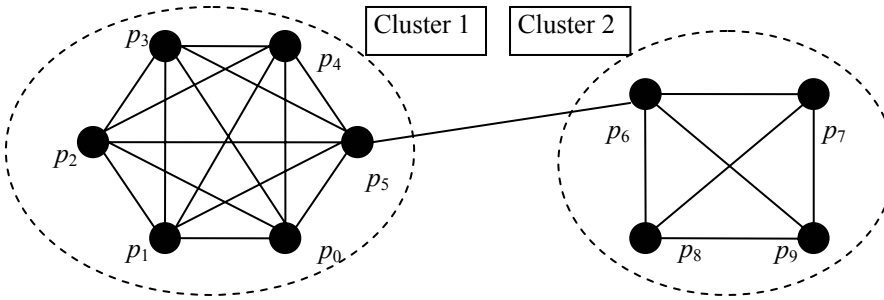


Fig. 1. Processor graph for Grid-1 and Grid-2.

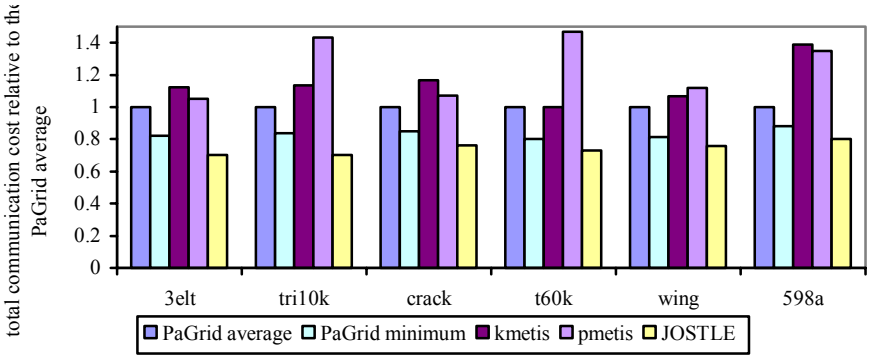


Fig. 2. Performance of PaGrid, METIS and JOSTLE for Grid-1 system graph: total communication cost relative to the PaGrid average.

Figures 2 and 3 show a comparison of total communication cost (Ω) for PaGrid (without execution time load balancing) with METIS and Jostle for Grid-1 and Grid-2 respectively. Since METIS did not permit us to specify different processor weights, we do not show results for METIS for Grid-2.

As can be seen in Figure 2, both the average and minimum values of Ω of PaGrid are better than METIS, but worse than Jostle in most cases. For Grid-2, Figure 3 shows that the PaGrid minimum is always better than Jostle, but the PaGrid average is only better for the two largest graphs.

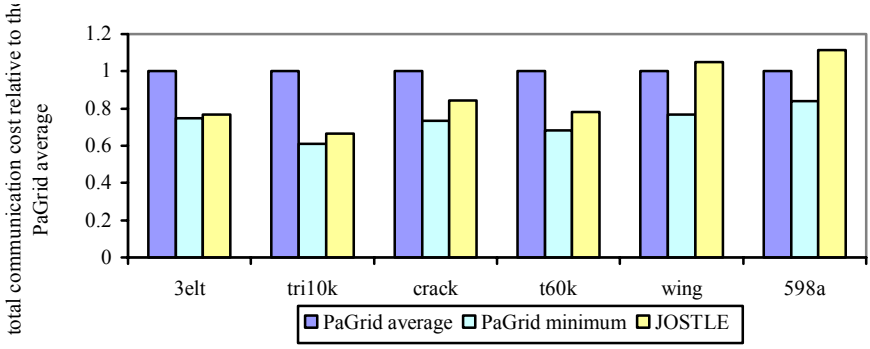


Fig. 3. Performance of PaGrid and JOSTLE for Grid-2 system graph: total communication cost relative to the PaGrid average.

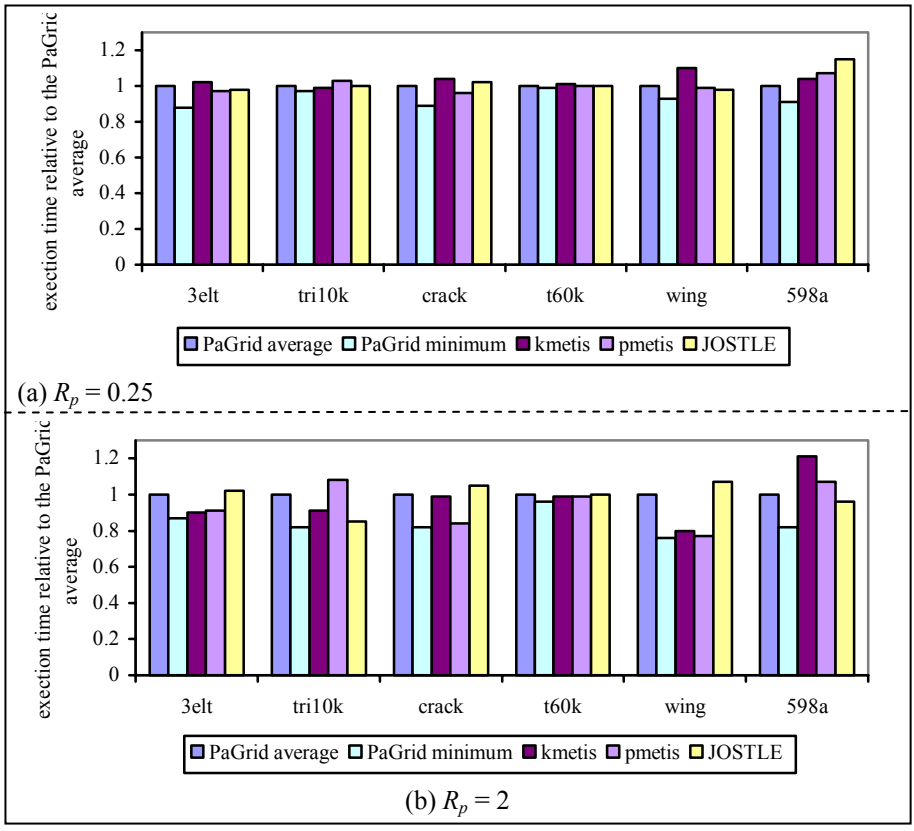


Fig. 4. Performance of PaGrid, METIS and JOSTLE for Grid-1 system graph: execution time relative to the PaGrid average.

Now we compare PaGrid with execution time load balancing against METIS and Jostle, and measure the quality of partitions with respect to the estimated execution time of the application (t). Figure 4 shows the performance of the three partitioners for Grid-1, with R_p equal to 0.25 and 2. The minimum values of execution time for PaGrid are always lower than for the other partitioners. Even though our results are worse than Jostle in terms of average Ω (see Figure 2), our execution time load balancing generates comparable or better results of average execution time than Jostle in most cases. This demonstrates that our execution time load balancing can improve the quality of a given partition.

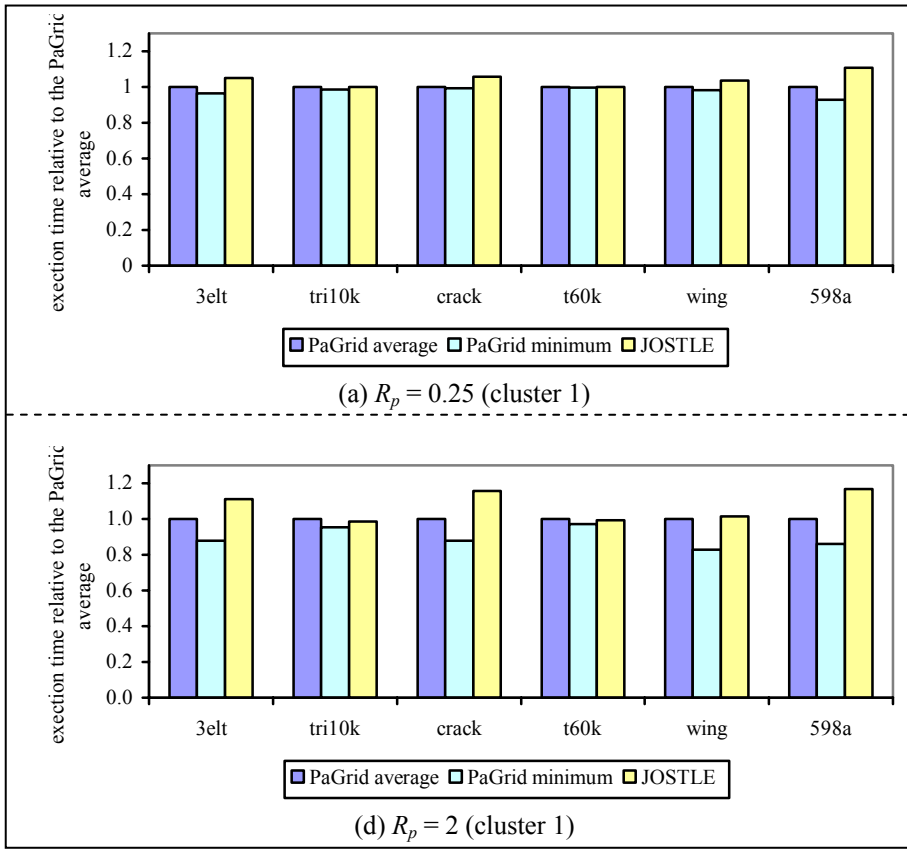


Fig. 5. Performance of PaGrid and JOSTLE for Grid-2 system graph: execution time relative to the PaGrid average.

One notable feature of Figure 4 is that METIS produces partitions with similar or lower estimated execution times than the others in most cases. Note also that METIS does even better for the fine grained case (R_p = 2) than the coarse grained case (R_p = 0.25). Compare this with Figure 2, where METIS produces partitions with higher

communication cost than the others. This suggests that while a partition may be optimal in terms of communication cost, it may not be optimal in terms of execution time, particularly for fine-grained applications. PaGrid, in its final stage, attempts to improve the partition by minimizing the estimated execution time. Since the results from PaGrid are better than Jostle, we infer that this execution time refinement is successful to some extent.

As can be seen in Figure 5, our results are better than Jostle in all test graphs, saving as much as 17% for the graph "598a" when R_p =2. Compared to the results for Grid-1, the PaGrid results for Grid-2 are relatively better. This could be explained as follows. The effect of the slow link between p_5 and p_6 is accentuated in Grid-2 because the disparity in processor speeds is higher than in Grid-1. Since Jostle assigns vertex weights based on only the computational load, it cannot take into consideration the disparity, whereas PaGrid can assign vertex weights taking into account communication cost as well.

The results demonstrate that our execution time load balancing generates partitions with better quality in terms of estimated execution time, which means applications would benefit from the partitions generated by our mesh partitioner. It is also worth pointing out here that partitions of better quality could be achieved by allowing more imbalance for the final partition in the uncoarsening phase preceding the execution time load balancing.

4 Conclusion

We have presented a comparison of the performance of our mesh partitioner, PaGrid, against two existing partitioners, METIS and Jostle. PaGrid makes use of some of the techniques used by the other two partitioners, and adds an execution time load balancing stage in the final phase. We have found that PaGrid produces comparable or better partitions that Jostle. Although the performance of METIS is worse in terms of total communication cost than PaGrid and Jostle, it is comparable to them in terms of estimated execution time. We have found that estimated execution time is a better metric for the performance of a mesh partitioner than communication cost. In future we plan to evaluate both the partitions obtained with PaGrid and our metric based on estimated execution time by executing applications on computational grids.

References

1. Schloegel K., Karypis G., Kumar V., "Graph Partitioning for High-Performance Scientific Simulations", in "Sourcebook of Parallel Computing", Dongarra, J., et al. (eds.), Morgan Kaufmann, chap. 18, 2003.
2. Chen J., Taylor V. E., "PART: A partitioning tool for efficient use of distributed systems", In *Proceedings of the 11th International Conference on Application Specific Systems, Architectures and Processors,* Zurich, Switzerland, pp. 328-337, July 1997.
3. Chen J., Taylor V. E., "ParaPART: Parallel Mesh Partitioning Tool for Distributed Systems", in *IRREGULAR'99, Sixth International Workshop on Solving Irregularly*

Structured Problems in Parallel, in Conjunction with IEEE IPPS/SPDP'99 (13th International Parallel Processing Symposium), San Juan, Puerto Rico, April 1999.

4. Connolly D. T., "An improved annealing scheme for the QAP", European Journal of Operational Research, vol. 46, pp. 93-100, 1990.
5. Foster I., Kesselman C., "The Globus Project: A Status Report". *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998.
6. Grimshaw A. S., Wulf Wm. A., "The Legion Vision of a Worldwide Virtual Computer", *Communications of the ACM*, vol. 40, No.1, pp. 39-45, January 1997, http://www.cs.virginia.edu/~legion/papers/cacm.ps, June 10, 2001.
7. Gropp W., Lusk E., Skjellum A., "Using MPI, 2nd Edition: Portable Parallel Programming with the Message-Passing Interface", MIT Press, 1999.
8. Huang, S., "PaGrid: a Mesh Partitioner for Computational Grids", MCS thesis, Faculty of Computer Science, University of New Brunswick, Canada, 2003.
9. Karonis N., Toonen B., Foster I., "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface", *Journal of Parallel and Distributed Computing,* to appear, 2003.
10. Karypis G., Kumar V., "A fast and high quality multilevel scheme for partitioning irregular graphs". *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359-392, 1998.
11. Karypis G., Kumar V., "Multilevel k-way Partitioning Scheme for Irregular Graphs", *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 96-129, January 1998.
12. Kumar S., Das S., Biswas R., "Graph Partitioning for Parallel Applications in Heterogeneous Grid Environments", *International Parallel and Distributed Processing Symposium*, Florida, April 2002.
13. Schloegel K., Karypis G., Kumar V., "Multilevel Diffusion Schemes for Repartitioning of Adaptive Meshes", *Journal of Parallel and Distributed Computing*, vol. 47, No. 2, pp. 109-124, December 1997.
14. Taylor V.E., Schwabe E.J., Holmer B.K., Hribar M.R., "Balancing Load versus Decreasing Communication: Parameterizing the Tradeoff", *Journal of Parallel and Distributed Computing*, vol 61, pp. 567-580, 2001.
15. Walshaw C., Cross M., Johnson S., Everett M., "JOSTLE: Partitioning of Unstructured Meshes for Massively Parallel Machines", in*, editor, Parallel Computational Fluid Dynamics: New Algorithms and Applications*, N. Satofuka et al. (eds.), Elsevier, 1995.
16. Walshaw C., Cross M., "Mesh Partitioning: A Multilevel Balancing And Refinement Algorithm", *SIAM Journal on Scientific Computing*, vol. 22, no. 1, pp. 63-80, 2000.
17. Walshaw C., Cross M., "Multilevel Mesh Partitioning for Heterogeneous Communication Networks", *Future Generation Comput. Syst*., vol. 17, no. 5, pp. 601-623, 2001.
18. Walshaw C., Cross M., "Parallel Optimisation Algorithms for Multilevel Mesh Partitioning". *Parallel Computing*, vol. 26, No. 12, pp.1635-1660, 2000.
19. Pellegrini F, Roman J., "SCOTCH: A Software Package for Static Mapping by Dual Recursive Bipartitioning of Process and Architecture Graphs", *Proceedings of HPCN'96*, Brussels, Belgium. LNCS 1067, pp. 493-498. Springer, April 1996.
20. Walshaw C.. "University of Greenwich Graph Partitioning Archive". Retrieved January 10th 2003 , from http://www.gre.ac.uk/~c.walshaw/partition/.

CONFIIT: A Middleware for Peer to Peer Computing

Olivier Flauzac[1], Michaël Krajecki[1,2], and Jean Fugère[2]

[1] LERI - EA2618, Université de Reims Champagne-Ardenne,
BP 1039, F-51687 Reims Cedex 2, France
{olivier.flauzac, michael.krajecki}@univ-reims.fr
[2] Royal Military College of Canada
PO Box 17000, Station Forces
Kingston, Ontario CANADA K7K 7B4
{michael.krajecki,fugere-j}@rmc.ca

Abstract. Once applications with Finite number of Independent and Irregular Tasks (FIIT) have been introduced, CONFIIT is presented. This is a fully distributed peer-to-peer environment designed to compute FIIT problems. This Java written middleware aims at setting a logical ring organization for network's resources, such as PCs, workstations or parallel architectures. It also offers a low-cost communication solution to share, and compute, the different tasks of a FIIT problem over the structured system. The efficiency of this solution, is shown, through experiments, using the Langford's problem.

1 Introduction

The requirement for computation power and storage capacity always increases. At the same time , the capacity of computation also increases. But, the demands are still exceeding the supply. During many years, computation and storage were done in a centralized way. A few years ago, a new approach has emerged: the distribution of computation and storage over a network. This new approach tends towards the utilization of all available computational aspects of a local network, and to a larger scale of the Internet. This new way to design computational programs is called GRID computing. In [5], authors discussed arising issues related to this new paradigm, and explain how to design efficient GRID based solutions.

CONFIIT, introduced in this paper, is a fully distributed peer to peer middleware, designed to cope with sharing the computation means to be used on a local network or on the Internet.

The rest of this paper is organized as follows : firstly, we present actual issues in GRID computing. Then FIIT applications are introduced, and CONFIIT is described. Then first experimental results are then discussed. Further extensions are mentioned at the end.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 69–78, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Related Work

GRID applications are actually designed for two main purposes: data sharing and distributed computation.

Data sharing GRID applications aim is to allow users to share or recover data in different formats through a network. Those applications are based on several components : file searching, file publication, statistics gathering, and of course, file downloading. We can find in these categories Napster, FreeNet, Gnutella...

Distributed computation GRID applications aims to share a computation over a network, to "build" a general solution. In the remaining of this paper, we will focus almost exclusively on Distributed computation GRID applications.

With the GRID paradigm, came different applications, middlewares or environments. The related issues can be classified in three main categories. The first one is the family of applications specially designed to compute a particular problem. The most known application of that category is the seti project[1], designed to Search for Extraterrestrial Intelligence. The second one is the family of protocols and libraries, put together to help in developing a dedicated GRID application. JXTA project[2] is an example of this category. Finally , we find middlewares, offering different services (grid construction and management, tasks sharing management and results gathering). In this last family we find XtremWeb[3] [3] or Globus[4] [4,1].

Whatever the application is, and, whatever the means are, a GRID based application has to verify some properties to be efficient. In [7,6], authors expose desirable properties for an efficient grid system : sharing hardware and software capacities and capabilities, scheduling the use of those and the capability of peer to peer computation over Internet. F. Capello presents the topical issues in peer to peer globalized computation system.

In [2], Cappelo sheds light on the use,in most cases, of a centralized, or semi centralized architecture. We can notice that Freenet is designed as a distributed decentralised information Storage and retrieval system [12].

3 The FIIT Applications

In [10], we have defined the FIIT application. It is made up of a Finite number of Independent and Irregular tasks. We assume that each task satisfies three features: first, a task cannot make any hypothesis about the execution of any other task. Hence, there is no communication between tasks. Secondly, the execution time of each task is unpredictable. In other words, we cannot have a good estimate of the execution time before the task is completed. Thirdly, The same algorithm is applied to compute all the tasks of the application. Hence, two tasks are distinguished by the set of data they process.

[1] http://setiathome.ssl.berkeley.edu/
[2] http://www.jxta.org/
[3] http://www.xtremweb.net
[4] http://www.globus.org

Due to the task irregularity, we have to propose some suitable load balancing strategies in order to effectively parallelize the FIIT applications. Due to the lack of information on the task behavior, we introduce dynamic load balancing strategies. The communication overhead induced by the load balancing strategy should be reasonable if we want to get a good acceleration of a FIIT application.

To help the FIIT environment to bring an effective parallelization, the user must answer two questions: how to redefine his problem into a finite number of tasks and also has to provide a function that solves a task identified by its number. The choices carried out by the user are significant. He must choose a granularity (a number of tasks) adapted to the parallel machine he wants to use. Indeed, the granularity influences the quality of the load balancing but also impact on the level of use of the interconnected network.

4 CONFIIT: A Middleware for Peer to Peer Computing

CONFIIT (Computation Over Network for FIIT), is a middleware for peer to peer computing. It aims, to distribute, over the network, all the tasks obtained by the decomposition of a FIIT problem, to solve each distributed task, and to spread, over the network, computed results of those tasks. This middleware is fully distributed, and is designed to be applied to a heterogeneous network architecture : networks based on different operating systems (Unices,WinXX ...) and/or hardware (Intel, Sparc ...) and/or architecture (mono,multi-processor).

4.1 Global Architecture

Each computer of the network, using the CONFIIT middleware, is called *node*. This node is set up with three main threads basic components : a topology and communication manager, a tasks manager and a tasks solver, dedicated to one problem.

All the nodes are set up as an oriented ring, each node knowing its next neighbor in the ring. Communication between nodes is done using a token, carrying the state of the computation, around the ring.

A *task manager* can cope with several *tasks solvers*, if, and only if, all the *tasks solvers* are launched on the same node as the *task manager*. We can notice that this node can be a monoprocessor node or a shared memory multiprocessor one. Over and above these three main threads elements, each node owns a set of data.

4.2 Tasks Management

Once inserted in the ring, a node owns the different parameters of the actual computation. This node is able to set up its local parameters of the tasks to compute (list of tasks and associated results).

To avoid a parallel computation of the same task by all the nodes of the ring, the task manager gives to the tasks solver(s) a number, randomly chosen, associated to a task. This task is, of course known at that time, as uncomputed.

Some tasks may be computed in parallel by different nodes. Those tasks, called *replicated tasks*, increase the load of the system. Those *replicated tasks* are necessary to guarantee a fully distributed asynchronous computation without any centralized server providing tasks distribution, or any centralized scheduler. Even more, that heuristic is adapted to the GRID computation with heterogeneous hardware and network. Using that strategy, no node is waiting for the token, in order to compute a task.

5 Experimental Analysis

5.1 The Langford's Problem

C. Dudley Langford a mathematician, gave his name to a classic problem of permutation. While observing his son manipulating blocks of different colors, he noticed that it was possible to arrange three pairs of blocks of different colors (yellow, red, blue) in such a way that only one block separates the red pair, two blocks separate the blue pair and finally three blocks separate the yellow pair (to see the figure 1).

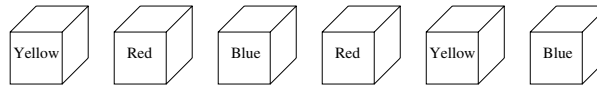


Fig. 1. L(2,3): arrangement for 6 blocks of 3 colors: yellow, red and blue

The problem has been generalized to any number of colors n and any number of blocks having the same color s. $L(s, n)$ consists in searching for the number of solutions to the Langford's problem. In November 1967, Martin Gardnker presented $L(2, 4)$ (two cubes and four colors) as being part of a collection of small mathematical games and stated that $L(2, n)$ has solutions for all n such that $n = 4k$ or $n = 4k - 1$ for $k \in \mathbb{N}$. Actually, the instances solved in practice, in a merely combinative manner, limit themselves to a small number of color. In this case, one mentions the instance $L(2, 19)$ that was solved in 2,5 years on a DEC Alpha to 300MHz in 1999. In 2002, $L(2, 20)$ was solved with the help of a specific algorithm and the intensive use of a cluster of 3 PCs during one week.

The reader wishing more information on this problem could consult John E. Miller's page[5].

In [9,8], we proposed to formalize the Langford's problem as a CSP (*Constraint Satisfaction Problem*) and showed that an efficient parallel resolution is possible. The

experimentations performed on a SGI Origin'3000 using OpenMP permitted to get an efficiency superior to 80 % with 256 processors.

[5] `http://www.lclark.edu/~miller/langford.html`

In this section, we are going to show that it is also possible to use the CONFIIT environment to solve this problem.

First we propose a modeling of the Langford's problem as a tree search problem. In order to solve $L(2, n)$, we consider the tree with $2(n-1)$ sons of height n according to:

- every node of the tree corresponds to the place in the sequence of the cubes of a determined color;
- to the depth p, the first node corresponds to the place of the first cube of color p in first position and it ith node corresponds to the investment of the first cube of color p in position $i,\ i \in [1,\ 2(n-1)]$;
- every leaf of the tree symbolizes the positions of all cubes;
- a leaf is a solution if it respects the color constraint defined by the Langford's problem.

It is now sufficient to propose a walk through the search tree, in depth first, to get a simple sequential algorithm solving the problem of Langford.

In a second time, this tree traversal can be considered as a FIIT application while introducing the following definition for the notion of task. A task is associated to the traverse of a particular subtree. While choosing to develop all subtrees to a depth k, we define $(2(n-1))^k$ independent tasks accessible using a unique identifier (numbering of the nodes in $2(n-1)$ basis).

It is noticed that, when introducing a backtracking scheme on the inconsistent branches (for which the first placed cubes do not already respect the color constraint), we observe that the computations associated to these tasks are especially irregular.

Finally, it is easy to verify that these tasks are independent and can be solved in any order.

5.2 The Java Based Implementation of CONFIIT

The prototype of CONFIIT, developed in Java, supports itself with six main classes and an interface. The class *CONFIIT* contains the main program.

When the initiator of the computation starts its execution, it first gathers the data necessary to the application (in the case of the Langford's problem, it is about the number of color and the depth of the division). If the node already joins one existing logical ring, it gets back the parameters of execution by the distant node that it had contacted parameters that will to be inserted in the logical ring.

After its initialization, the main program creates a thread of the class *Param*. This object gathers the set of information shared by the threads of a node. In particular, this object memorizes the local address of the node, the address of the next node in the logical ring, and the table of the tasks.

Then, the threads corresponding to the management of the ring (an instance of the class *ThreadRing*), to the management of the token (an instance of the class *ThreadToken*) and to the resolution of the tasks (one or several threads

of the class *Solves*) are created. The reader may notice that the constructors of the classes require an object of type *Param* that allows the collaboration of the different processes.

Every instance of the class *Solves* creates an object of type *Langford* that conforms to the interface *FIIT*. Thus, the user wishes to use CONFIIT for another FIIT application, will only have to provide such a class. The objects of the class *Solves* will be in charge to make the link between the CONFIIT environment and the problem at hand to be solved.

5.3 Experimentations

The first experimentations have been achieved using a set of PCs on a local area network as well as the parallel machine Roméo of the university of Reims Champagne-Ardenne (SUNFire 6800, 24 processors).

We have chosen the Langford's problem as example of a case of irregular applications. We have solved in practice the problems $L(2, n)$ with $n \in [11, 16]$. The serial times obtained on the SunFire 6800 are presented in table 1.

Langford's problem	L(2,11)	L(2,12)	L(2,13).	L(2,14)	L(2,15)	L(2,16)
Execution time	1s	7.5s	60s	526s	4933s (1h12min)	49139s (13h40min)

Table 1. Serial execution time of L(2,n), $n \in [11, 16]$, on a SUNfire 6800

Calculations on a Local Area Network On a first trial, we used up to 16 PCs (Celeron 433 Mhz and 466 Mhz) under WINDOWS NT. For the problem $L(2, 15)$, we did measures on different parameters according to the number of processors : execution time, number of *replicated tasks* as well as speed-up and efficiency. For the second trial, we repeated the set of these tests using 8 PCs Celeron (633 Mhz and 667 Mhz).

The sequential resolution time for $L(2, 15)$ is of 10582 seconds on a celeron 433 Mhz and 7733 seconds on a celeron 633 Mhz.

Table 2 presents the observed speed-ups and efficiencies. We can notice that CONFIIT behaves very well : we notice that using 16 celeron 433 Mhz divides the computation time by 14. These first experiments show that CONFIIT provides an efficient solution to share the computation on a classic local area network. We can ascribe these good performances to two skills of CONFIIT : the virtual ring and token management, although rather expensive, and the tasks random distribution to avoid *replicated tasks*. This last point has also been verified experimentally.

The resolution of $L(2, 15)$ generates 2186 tasks to a depth of 3 (we place the first 3 cubes arbitrarily). We observe experimentally that the number of *replicated tasks* increases slowly (it raises by a factor close to 2 from 8 to 16 processors). Besides, for 16 processors, less than 9 % of the tasks are *replicated tasks*. In practice, most *replicated tasks* could be avoided. Indeed, to avoid results loss originated from a potential breakdown of a node, the token is only updated

L(2,15)	1 proc.	2 proc.	4 proc.	8 proc.	16 proc.
Execution time	10582	5372	2931	1365	745
Speed-up	1	1,97	3,61	7,75	14,2
Efficiency	100%	98,5%	90,2 %	96,9 %	88,75 %

Table 2. Resolution of the problem $L(2, 15)$ on a network of PCs (celeron 433 Mhz)

when a task has been completely computed, and not when it has been affected to a thread. Such a strategy implies possible *replicated tasks*. Thus updating the token using a preemptive way (as soon as a task is distributed) will reduce the number of *replicated tasks* significantly, especially when the ring contains a large number of CONFIIT nodes.

Finally, we measured the necessary time to solve $L(2, 16)$ using the set of available PCs (a total of 30 machines). The complete resolution of the problem has been obtained in 55 minutes.

Calculations on Parallel Machine The university of Reims Champagne-Ardenne owns a parallel computer SUNFire 6800 with 24 UltraSparc III 900 Mhz processors and 24Go of core memory. We did performance tests to evaluate the efficiency of the multi-threads mechanism proposed by Java in the context of CONFIIT.

L(2,15)	1 proc.	2 proc.	4 proc.	8 proc.	10 proc.	12 proc.	16 proc.
Execution time	4933	2471	1239	626	499	415	331
Speed-up	1	1.99	3.98	7.88	9.88	11.89	14.90
Efficiency	100%	99.5%	99.5%	98.5%	98.9%	99.0%	93.1%

Table 3. Resolution of the problem $L(2, 15)$ on SUNFire 6800

CONFIIT's efficiency was proved: an almost linear acceleration was observed, depending of the number of processors we allocate (see table 3). We can notice that these results are very close to those obtained with the FIIT using a message passing environement [10].

Those computations did share the power of the parallel computer with other computations farmed by other users. This is the reason why we could not use the 24 processors of the SUNFire 6800 at the University of Reims Champagne-Ardenne.

We also made some experiments on a cluster of IBM SP3 and Regatta owned by the CINES[6] in Montpellier (France). We solved $L(2, 15)$ using up to 96 processors. Table 4 exposes obtained execution times. Results show, as well, that CONFIIT is a suitable solution to take advantage of parallel machines, in the context of FIIT applications. The reader may notice that the SP3 nodes exploit the Power3+/375 Mhz processor and that the Regatta node takes advantage of

[6] Centre Informatique National de l'Enseignement Supérieur, `http://www.cines.fr`

the new Power4/1,3 Ghz processor. For this reason, it is hard to provide classical speed-up and efficiency values.

Configuration	Execution time (in seconds)
1 node SP3	498.38s
1 node Regatta (16p)	194.88s
1 node Regatta (32p)	93.048s
2 nodes SP3	329.59s
3 nodes SP3	203.25s
4 nodes SP3	142.81s
4 nodes SP3 + 1 node Regatta (16p)	90.8s
4 nodes SP3 + 1 node Regatta (32p)	66.6s

Table 4. Execution times (in seconds) on IBM SP3/Regatta for L(15,2), depth = 3

Finally, some experiments were conducted using three different servers located in two different sites.

Thanks to HPVCL[7], a Sun Fire 15K (72 Sun UltraSPARC III 900 Mhz processors, located at Queen's University, Kingston, Ontario) and two sun entreprise servers (one with 4 processors and the last one with 16 processors, located at RMC, Kingston) were used to solve $L(2, 15)$.

Three main experiments were conduced using the three servers. For the first experiment, $L(2, 15)$ was solved using 16 processors (from the 15K server), 8 processors and 4 processors from the other two servers. Experiment 2 also solved $L(2, 15)$ using respectively 8, 4 and 2 processors. For the last experiment, $L(2, 15)$ has been solved using 32+8+4 processors.

Table 5 sums up, for each server, the execution times, the number of tokens exchanged and the number of tasks locally resolved.

216.22 seconds are necessary to solve $L(2, 15)$ using 32 processors of the Sun Fire 15K server, and 429.8 seconds using 16 processors. Thus, the results obtained using CONFIIT on two different sites are quite interesting. The efficiency observed between experiment 1 and 2 seems to be quite linear. Moreover, CONFIIT has achieved a good load balancing among the three servers : the most powerful (Sun Fire 15K) has solved the majority of the tasks in each experiment.

6 Concluding Remarks and Future Work

This survey did aim at presenting the first step of a Peer to Peer computing environment dedicated to FIIT applications. CONFIIT makes it possible to easily distribute the computations associated to the application through wide-area networks. The main characteristic, and benefit, of this environment is the fact that it does not require a centralized server. It is a fully asynchronous distributed environment.

[7] High Performance Computing Virtual Laboratory, `http://www.hpcvl.org`

Experiment 1	**Sun Fire R15**	**Sun Entreprise**	**Sun Ultra 450**
Nb of processors	16	8	4
Execution time	348.3	347.7	348.0
Nb of received token	1081	1081	1081
nb of tasks	1676	300	215
Experiment 2			
Nb of processors	8	4	2
Execution time	789.5	789.0	789.2
Nb of received token	2093	2093	2093
nb of tasks	1493	453	245
Experiment 3			
Nb of processors	32	8	4
Execution time	198.8	198.8	198.8
Nb of received token	639	639	639
nb of tasks	1894	175	120

Table 5. Execution times (in seconds) for L(15,2) in Kingston, depth = 3

The load balancing scheme is little sensitive to the performances of the network (network's capacities influence the token exchange speed) because it uses, in addition to the information transmitted within the token, a heuristic of random choice in order to avoid *replicated tasks*.

CONFIIT, developed in Java, performs the communications in TCP mode and uses the multi-threaded paradigm. Thus, CONFIIT may be applied to a large variety of material architectures : PCs, workstations or parallel machines. To achieve the first experiments, we chose to solve the Langford's problem. The computed results showed, using a network of PCs and on a parallel machine, the efficiency of the propositions made in the setting of the CONFIIT system. Large scale experiments, using several parallel machines accessible by internet are in progress and the preliminary results seem to confirm the efficiency of such a solution. Unfortunately, we are not able, at this point to provide them.

Finally, the object's approach,chosen to develop CONFIIT with, offers many advantages. Thus, it is easy to spread the CONFIIT environment to solve all applications, extending the FIIT interface. Users do not have to worry about underlying management, such as load balancing or logical structuration; the system manages, on its own, those tasks.

The perspectives of this first paper are numerous. Many interesting aspects of grids computing related to this work have not been discussed in this paper.

For example, every CONFIIT node must locally have the specific classes in the FIIT application to be able to join the logical ring. We plan to use the mechanisms of dynamic loading the classes offered by Java to allow a node to download, and instantiate these classes, during its insertion in the ring.

There are also the problems of authentication and security that should be to studied: management of the group of CONFIIT users, confidence in a result calculated on a distant site, are illustrations of these problems.

How many CONFIIT nodes can be managed efficiently by this middleware while using the logical ring-shaped structuring and taking into account the ca-

pacities of the different networks? When the number of sites becomes large, it is maybe desirable to use an array with a smaller diameter. A modelling using *OMNet++*[8] [11] should allow us to obtain a first answer to these questions.

Acknowledgements

This job has been achieved with the support of the ARGE[9] animation of the GRID ACI.

The authors wish also to thank the CINES (national computer center of the higher education in France) and HPCVL (High Performance Computing Virtual Laboratory).

References

1. W. Allcock, A. Chervenak, I. Foster, L. Pearlman, V. Welch, and M. Wilde. Globus toolkit support for distributed data-intensive science. In *Computing in High Energy Physics (CHEP '01)*, September 2001.
2. F. Cappello. Calcul global pair à pair : extension des systèmes pair à pair au calcul. *La Lettre de l'IDRIS*, 4:14–25, apr 2002.
3. G. Fedak, C. Germain, V. Néri, and F. Cappello. Xtremweb : A generic global computing system. In *CCGRID2001, workshop on Global Computing on Personal Devices*. IEEE Press, 2001.
4. I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Supercomputer Applications*, 11(2):115–128, 1997.
5. I. Foster and C. Kesselman. *Computational Grids*, chapter 2 in The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufman, 1998.
6. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6), 2002.
7. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 15(3), 2001.
8. Z. Habbas, M. Krajecki, and D. Singer. Parallelizing combinatorial search in shared memory. In *Fourth European Workshop on OpenMP (EWOMP'2002)*, page to appear, Roma, Italy, sep 2002.
9. Z. Habbas, M. Krajecki, and D. Singer. Étude de performance de la mémoire partagée pour les applications irrégulières. In *quatorzième édition des Rencontres Francophones du Parallélisme (RENPAR 2002)*, pages 91–96, Hammamet, Tunisie, apr 2002.
10. M. Krajecki. An object oriented environment to manage the parallelism of the FIIT applications. In V. Malyshkin, editor, *Parallel Computing Technologies, 5th International Conference, PaCT-99*, volume 1662 of *Lecture Notes in Computer Science*, pages 229–234. Springer-Verlag, St. Petersburg, Russia, Sept. 1999.
11. A. Varga. The omnet++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, 2001.
12. I. Clarke A Distributes Decentralised Information Storage and Retrieval System University of Edinburgh

[8] `http://www.hit.bme.hu/phd/vargaa/omnetpp.htm`
[9] http://www-r2.u-strasbg.fr/rge/

On MARS's s-boxes Strength against Linear Cryptanalysis

Carlos Javier Hernández Castro[1], Luis Javier García Villalba[2],
Julio César Hernández Castro[3], José María Sierra Cámara[3]

[1] Complutense University, Servicio Informático de Gestión, Madrid, Spain
chcastro@pas.ucm.es
[2] Complutense University, Departamento de Sistemas Informáticos y Programación,
Facultad de Informática, Madrid, Spain
javiergv@sip.ucm.es
[3] Carlos III University, Computer Security Group, Computer Science Department,
28911 Leganés, Madrid, Spain
jcesar@inf.ucm.es, sierra@inf.ucm.es

Abstract. MARS's s-boxes were generated using a new algorithm developed by the IBM team, which was supposedly able of producing secure s-boxes against both differential and linear cryptanalysis. In this paper we show this is not the case, because their strength against linear cryptanalysis is not better (in fact, it seems to be worse) that what could be expected if generated randomly.

1 Introduction

MARS's s-boxes were obtained after a week of calculations using a new algorithm developed by IBM [1] which supposedly guaranteed the generated s-boxes had excellent properties against both differential and linear cryptanalysis. The designers of MARS did in fact conjecture that its maximum bias was not greater than 2^{-3}, a value that was, afterwards, proved to be too optimistic in [3] and [4].

In this paper we propose a mathematical model for the number of masks for a given bias value which, in particular, will have shown that the IBM's conjecture was extremely improbable and that the final results presented in [5] are not best than what could have been obtained at random. That simply means the new algorithm for producing s-boxes is not better, at least from the linear cryptanalysis point of view, than using a simple random generation.

In fact, we suggest that the generation procedure, basically a random generation followed by a series of tests and an optimization search between the s-boxes that were found to pass the filters, would probably never produce cryptographically sound s-boxes.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 79-83, 2003.
© Springer-Verlag Berlin Heidelberg 2003

1.1 MARS's s-boxes

As mentioned in [1], MARS's s-boxes were generated "in a pseudorandom fashion" (by using 32 bit words of the output of the SHA-1 algorithm with some fixed constants, an index and another 32 bits as input) and then filtered by testing they have good differential and linear properties. The properties tested were:

1. The S-box does not contain the all-zero or the all-one word.
2. Within each of the two s-boxes S_0, S_1 every two entries differ in at least three of the four bytes
3. S does not contain two entries S[i], S[j] with $i \neq j$ such that S[i]=S[j], S[i]=not(S[j]) or S[i]= - S[j]
4. S has $\binom{512}{2}$ distinct xor-differences and $2\binom{512}{2}$ distinct subtraction differences.
5. Every two entries in S differ by at least four bits.

Additionally, the algorithm tried to minimize the following values, in order to make the resulting s-boxes stronger against linear cryptanalysis:

6. Parity bias $|Pr_x[parity(S[x])=0]-1/2|$, requiring the bias to be at most 1/32
7. Single bit bias. For every j $|Pr_x[(S[x]_j)=0]-1/2|$, being it 1/30 at most
8. Two consecutive bit bias. For every j $|Pr_x[(S[x]_j \text{ XOR } S[x]_{j+1})=0]-1/2|$, requiring the bias to be at most 1/30
9. Single bit correlation: For every generated s-box which satisfies conditions 1-8, minimize $|Pr_x[(S[x]_j)=x_i]-1/2|$

Where the thresholds set above were calculated experimentally. The generation and testing proceeded for "around a week" studying 2^{26} values, and after that, the value which minimized the single bit correlation bias was selected. However, in [2] it its pointed out that the S-box actually fails to meet all the criteria shown above.

2 MARS's s-boxes Linear Probabilities Complete Distribution and Model

In [5] the author presented the complete distribution of the linear probabilities of MARS's s-boxes, calculated after using about 2 months idle time of our processors (5 processors ranging from 500 to 266 MHz.). In Table 1, we show both the real number of masks for each bias and the expected number of masks, as calculated by the Formula in **(1)** below:

$$\#\{m \in Z_2^{9+32} \mid bias(m) = n\} \approx 2 \frac{\binom{512}{256+n}}{2^{512}} 2^{41} \qquad (1)$$

Table 1. Number of masks for a given bias, with the expected number given by the formula in **(1)**, for some selected values

Bias * 2^9	Frequency	Expected
0	77498737588	77503773066,8126
1	154403399557	154404403930,770
2	152613000039	152609003885,064
3	149664380489	149662884118,943
4	145634858332	145633498777,279
5	140608578512	140611653991,855
64	14136	15183,8400327846
77	6	9,889082203093
78	4	5,299837468124
79	3	2,816033042764
80	0	1,483445977885
81	1	0,774737365304
82	2	0,401121416947
83	1	0,205885329052
84	1	0,104759299782

As shown in Table 1, the goodness of the formula given in (1) is quite good, except for the higher biases. However, the real values will not pass a chi-square goodness of fit test In particular, we can observe that the conjecture of the developers of MARS

about the higher bias being at most 2^{-3}, which translates to a value of 64, was very optimistic, as one could expect around 20,000 masks above this value. The discoveries in [3], which first pointed out this conjecture was too optimistic and exhibited biases over 80, and later the findings in [5], which showed biases over 82, would have been easily predictable by simply using the proposed formula.

Recalling that this formula reflects the distribution that could be expected if the s-boxes were generated randomly, one is tempted to conclude that not only the algorithm proposed by the developers of MARS is not clearly better (with respect to linear cryptanalysis, at least) than the much simpler random generation, but indeed worst for high biases, which are precisely those that matter to find the best linear approximations for the s-boxes. So it seems that, instead of spending a hole week or more in generating and testing 2^{26} s-boxes, the MARS team would have done better (on average) just by generating a single s-box at random.

3 Conclusions

The claim of a bias not higher than 2^{-3} by the MARS's team was spectacular and rather improbable, as far as there should be more than 18,000 masks with biases higher than 64 if the s-boxes were generated completely at random. Their conjecture would only hold if, by means of using a brand new s-box generation algorithm, they were able of obtaining much better s-boxes than those one could obtain at random, which would have been a very important achievement in the field but is obviously not the case, as the final s-box was not better with respect to its linear approximations than what could be expected of a random s-box.

Additionally, we do not believe the general procedure for obtaining s-boxes proposed in [1] nor the supposedly better method shown in [2] have much future in cryptology. Although there are previous works on generating Boolean functions by using heuristic optimization methods such as hill-climbing [6] and genetic algorithms [7], we believe that the best cryptographic primitives could not be obtained by maximizing/minimizing any simple value as in those proposals. Improving the strength against linear or differential cryptanalysis will surely imply weaknesses against new kinds of cryptanalytic methods, some perhaps yet to be discovered.

References

1. Burwick, C., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C., Matyas,S., O'Connor, L., Peyravian, M., Safford, D., Zunic, N.: MARS- a candidate cipher for AES. Proceedings of the First AES Conference (1999). Revised September 22, 1999
2. Burnett, L., Carter, G., Dawson, E., Millan, W.: Efficient Methods for generating MARS like S-boxes. Proceedings of the Fast Software Encryption 2000 (FSE'2000)
3. Robshaw, M., Yin, Y.L.: Potential flaws in the conjectured resistance of MARS to linear cryptanalysis: Proceedings of the 3rd AES Conference

4. Knudsen, L., Raddum, H.: Linear Approximation to the MARS S-box. NESSIE Deliverable, April 2000
5. Aoki, K.: The Complete Distribution of Linear Probabilites of MARS's s-box. Cryptology e-print n°33, June 30, 2000
6. Millan, W., Clark, A., Dawson, E.: Boolean Function Desing using Hill-Climbing Methods. Proceedings of the Symposium on Applied Cryptography (SAC'97). LNCS 1587
7. Millan, W., Burnett, G., Carter, G., Clark, A., Dawson, E.: Evolutionary Heuristics for finding Cryptographically strong s-boxes. Proceedings of the Information and Communication Security, Second International Conference, ICICS'99, Sydney, Australia, November 9-11, 1999. Lecture Notes in Computer Science 1726

A Secret Sharing Scheme Using Matrices

Candelaria Hernández-Goya, Pino Caballero-Gil, and Carlos Bruno-Castañeda

Dept. Statistics, Operations Research and Computing
Universidad de La Laguna
38271 - La Laguna, Tenerife, Spain
mchgoya@ull.es, pcaballe@ull.es

Abstract. This work includes the description of a new multiparty protocol devoted to the sharing of secrets whose main application is key distribution. The particularity of this scheme consists of its basis on a problem classified as $DistNP - Complete$ under the average-case analysis, the so called *Distributional Matrix Representability Problem*.
One of the most critical questions in Cryptography is referred to the misunderstanding equivalence between using a difficult problem as basis of a cryptographic application and its security. Problems belonging to NP according to the worst-case analysis are frequently used in Cryptography but when random generated instances are used, then many times there are fast an efficient algorithms to solve them. So, using the search version of the Distributional Matrix Representability problem we guarantee the security of the proposed scheme.

1 Introduction

In this paper the use of problems catalogued as $NP - complete$ from the point of view of the average-case analysis is proposed for the design of a new Secret Sharing Scheme (SSS).

When basing a cryptographic application on a determined problem the main characteristic we look for is that fixing one of its instances to find a solution will be computationally impracticable, whereas to generate pairs formed by $(instance, solution)$ can be efficiently accomplished. Other usual property of the selected problem is that the verification procedure for any solution should be as simple as possible.

The three previous reasons justify the so extended utilization of problems belonging to the worst-case NP and $NP-complete$ classes in the area of the design of cryptographic applications. However, with the development of the Computational Complexity Theory, and concretely thanks to the advances made on the Average-Case analysis, it has been proved that some $NP - complete$ problems may be efficiently solved when the inputs are randomly generated [1]. One of the most immediate conclusions of this fact is to choose as base of cryptographic applications problems whose difficulty is guaranteed by the average-case analysis. The problem selected for the scheme here described is the designated as *Distributional Matrix Representability Problem*, which possesses that characteristic [2].

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 84–90, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The structure of the present work is as follows. The next section is devoted to the introduction of some necessary concepts, as well as to the location of the proposal in the existing context of Secret Sharing. In section 3, the proposed Secret Sharing Scheme is described in detail. Section 4 ends the paper with the presentation of the conclusions and some explanatory commentaries.

2 Preliminaries

Average-Case Complexity The average-case analysis is based on the concept of distributional decision problem, which is formed by a decision problem and a probability distribution defined on the set of instances [3].

In such a context the choice of the probability distribution plays an important role since it has a direct influence on the practical complexity of the problem. In fact, it has been proved that many $NP - complete$ problems may be solved in polynomial time when the instances are randomly generated under certain distributions.

The distributional class analogous to NP in the hierarchy associated to the average-case analysis is the $DistNP$ class. It is formed by pairs $< D, \mu >$ containing a decision problem D belonging to the NP class, and a probability distributions μ that is polynomially computable.

A distributional problem is said to be average-case $NP - complete$ (or complete for $DistNP$) if it it is in $DistNP$ and every distributional problem in $DistNP$ is reducible to it. The first problem catalogued as $DistNP - complete$ is the distributional tiling problem. Its formal proof of membership may been found in [3]. Later, on other works, descriptions of several new average-case $NP - complete$ problems have been published, [2], [4], [5], [6].

The main difficulty that we have found when trying to use problems belonging to this category for practical cryptographic applications is the artificiality of their specifications. So, the principal reason why we have chosen the Distributional Matrix Representability Problem as base of the proposed Secret Shared Scheme is its naive formulation. The original problem can be roughly defined in the following way: given a matrix A and a set of matrices with same size $M = \{A_1, A_2, ..., A_k\}$ it should be decided whether A can be expressed as a product of matrices belonging to the given set or not. This problem was shown to be indecidable for 6×6 matrices [7] and was later improved to 4×4 matrices [8]. On the other hand, a bounded version of this problem for 20×20 matrices defined in [6] is used here. In this case the instance consists of a matrix A, a set M of k distinct matrices and a positive integer n. All the matrices intervening in the problem are square and with integer entries. Now, the question to answer may be stated as follows: is it possible to express A as a product belonging to M^n, where M^n is formed by all the products of n matrices from M, and $n \leq k$. The distribution considered to generate the integers k and n, and the integer entries of the matrices is the uniform distribution.

The proposed scheme uses the search version of the above distributional problem. The difficulty of this such a version is equivalent to that of the distributional

decision problem as may be deduced from the general result stated in [9], according to which, search and decision distributional problems are equivalent from the average-case analysis point of view.

Secret Sharing Schemes Secret Sharing multiparty protocols solve usual practical situations in which it is necessary the distribution of a particular secret S among a set of users P. Such context may be illustrated with the problem of secret keys protection. So, the main objective of Secret Sharing Schemes is to guarantee that only pre-designated subsets of participants are able to reconstruct the secret by collectively combining their shares (or shadows) of S. The specification of all the subsets of participants which are authorized to recreate the secret is called the access structure of the SSS. It is said to be monotone if any set which contains a subset that can recover the secret can itself recover the secret. A methodology to design SSS for arbitrary monotone access structure was given in [10] and [11]. Such results were not useful for the SSS proposed here because the access structure is not monotone.

The first Secret Sharing Schemes were independently proposed in 1979 [12], [13]. Later it was demonstrated that both proposals can be considered in a most general scheme due to their basis on the same principles of linear algebra, [14]. Many different mathematical structures such as polynomials, geometric configurations, block designs, Reed-Solomon codes, vector spaces, matroids, complete multipartite graphs, orthogonal arrays and Latin squares have been used to model secret sharing schemes.

The scheme here described is based on the generation of a secret matrix as product of matrices with the same size.

A common model of SSH is based on two phases. In the initialization phase, a third trusted party called the dealer, distributes shares of the secret to authorized participants through a secure channel. In the reconstruction phase, the authorized participants of a subset in the access structure combine their shares to reconstruct the secret. In the initialization of the SSS proposed here the dealer will publish all the shares and the only secret information that is revealed to each participant is a pointer to a concrete share and the names of the other parties in the same subset of the access structure. Secret sharing schemes which does not reveal any information about the shared secret to unauthorized individuals are called perfect. So, the SSS proposed here is perfect and its security is unconditional.

3 A SSS on an Average-Case Intractable Problem

It is possible to establish two variants of the proposed protocol depending on whether the participation of a Third Trusted Party (TTP) is considered or not. If its intervention is decided then it will be in charge not only of the generation of the secret but furthermore it will reconstruct the secret, maintaining it protected from the rest of participants.

In order to facilitate the general description of the protocol with TTP it will be split into four stages: set-up, distribution, verification and recovery stages.

Set-Up Stage. This stage consists of the generation of the secret, task that is equivalent to the generation of an instance of the underlying problem. It should be carried out under privacy by the TTP.

This stage starts with the random generation of two integers k and n verifying the inequality $n \leq k$.

Once done, this k matrices $A_i, i = 1, \ldots, k$ with integer entries and size 20×20 are randomly generated such that two products of n matrices coincide. All the matrices form the set denoted by M and are identified by an index determined by its position in this set, the two subsets of n matrices constitutes the access structure and the product of the n matrices in each subset is considered the secret information.

The main parameter of this kind of protocols is the number of indispensable participants to recover the secret, known as the cardinality of the privilege users in the access structure. In the current scheme it is denoted by n and should be large enough to avoid that an exhaustive search attack could be successful. Here the search space for this sort of attacks is formed by all the possible products built with n matrices in the set M, so the cardinality of this search space is $\binom{k+n-1}{n}$.

The set M is published in an indexed directory with only lecture permission where each authorized participant can have access to his or her pointer share.

Distribution Stage. This phase requires the existence of a secure communication channel, or either the use of some cipher.

The usual main problem of this stage is the bandwidth necessary to transfer the shadows. However, such as difficulty is here avoided by sending to each authorized participant user the index assigned to the corresponding matrix in the access structure.

Verification Stage. This stage allows to detect the presence of cheaters among the shadow holders and to guarantee the correctness of the secret construction. The Monte Carlo algorithms described by Freivalds [15] for the verification of the product of two matrices is here used to achieve the fraud detection process. The error probability in these algorithms is bounded by 2^{-t}, where t is the number of iterations to be performed. Furthermore, all the necessary matrices products will be carried out using the algorithm proposed in [16] due to its efficiency.

The first step of the verification stage is the generation of a random binary vector U with the same dimension that the matrices intervening in the protocol. It may be done both by the TTP that holds the secret or through a random public generator. The generated vector U is multiplied by the secret matrix A obtaining U', that is a new vector containing random linear combinations of the

rows of the secret matrix. This vector U' and the random vector U are located in the previously mentioned public directory.

At this point, a permutation of the participants set $\{P_{(1)}, P_{(2)}, \ldots, P_{(n)}\}$ is randomly chosen. Such a permutation establishes the order in which the verification is developed. The participant designated in first place according to the previous permutation ($P_{(1)}$) privately computes the product of his or her shadow by the random binary vector obtaining $A_{(1)}U$, and sends this result to the next participant ($P_{(2)}$) determined by the permutation. In this way, $P_{(2)}$ computes the product determined by his or her shadow and the vector provided by $P_{(1)}$, $(A_{(2)}A_{(1)}U)$, and so on. Only if all the participants have been honest, $P_{(n)}$ obtains U', which communicates to the others.

According to the previous process, if some of the participants forge his or her shadows then it will be detected with probability strictly greater than 1/2. If a better security level is desired, then the complete verification process may be repeated a sufficient number of times.

Recovery Stage. In order to recover the secret every participant should access to the directory where the set M is available to obtain his or her shadow. Afterwards, and depending on the existence of a TTP, there are two possible actions:

- the TTP intervention is considered: In this case, each participant send his or her shadow to the TTP, who will reconstruct the secret
- the TTP is avoided: Then, first participant in the permutation reveals his or her shadow $A_{P_{(1)}}$ to the following user, who multiplies his or her shadow by the information transferred and so on (taking into account that the intermediate products should be sent using secure means). This way to proceed allows the equitable distribution of computation complexity among all the participants, and to reveal only one of the shadows.

Notes and Comments. Another scheme based also in matrices was proposed in [17], but there the secret is a solution to a system of linear equations.

In the generation of the vector U the binary vectors with Hamming weight 1 should be discarded, because otherwise the vector U' will coincide with a column of the secret matrix A. Thus, the cardinality of the set of possible binary vectors is $2^{20} - 20$.

An advantage of this SSS is that the secret is not revealed in any time to the participants, so the scheme allows the reusability of the secret and shadows..

As mentioned before, a concept extensively used in SSS is the unconditional security of perfect schemes. A SSS is considered unconditionally secure against cheaters if the probability of successfully cheating does not depend on the computational abilities of the cheaters. In this sense, and thanks to the described verification procedure, the SSS included in this paper may be considered as unconditionally secure.

4 Conclusions

In this paper we have proposed a secret sharing scheme based on an average-case NP-complete problem, the so-called distributional matrix representability problem. The proposal does not reveal any information about the shared secret matrix to unauthorized parties, and the size of each share equals the size of the secret, so the scheme is ideal. Although it is not the first sharing secret scheme connected with combinatorial structures, the main advantage of the proposed scheme is that its security is guaranteed by its average-case complexity. The study of concrete constructions of difficult instances of the problem that are adequate according to the design of the scheme is part of a work in progress, so we hope that a forthcoming version of this work will include it. Also a complete analysis of the security of the scheme and a comparison with other known schemes are questions that deserve further research.

References

1. Karp, R.: The Probabilistic Analysis of Some Combinatorial Search Algorithms. Academic Press, NY (1976) Referencia cruzada de artículo Levin Venkatesan.
2. Wang, J.: Average-Case Intractable NP Problems. In Du, D., Ko, K., eds.: Advances in Languages, Algorithms and Complexity. Kluwer Academic Publishers (1997) 313–378
3. Levin, L.: Average Case Complete Problems. SIAM Journal on Computing (1986) 285–286
4. Venkatesan, R., Levin, L.: Random Instances of a Graph Colouring Problem are Hard. In: ACM Symposium on Theory of Computing. (1988) 217–222
5. Gurevich, Y.: Matrix decomposition problem is complete for the average case. In: Proc. 31stAnnual Symposium on Foundations of Computer Science, IEEE Computer Society Press (1990) 802–811
6. Venkatesan, R., Rajagopalan, S.: Average case intractability of diphantine and matrix problem. In: Proc. Of the 24th Annual Symposium on Theory of Computing, ACM Press (1992) 632–642
7. Markov, A.: Theory of algorithms. (1954)
8. Markov, A.: On the problem of representability of matrices. Z. Math. Logik Grundlagen Math (1958) 157–168
9. Ben-David, S., Chor, B., Goldreich, O., Luby, M.: On the Theory of Average Case Complexity. Journal of Computer and System Sciences **44** (1992) 193–219
10. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. In: Proceedings IEEE Globecom '87. (1987) 99–102
11. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In Goldwasser, S., ed.: Crypto '88, Berlin, Springer-Verlag (1989) 27–36LNCS Vol. 403.
12. Shamir, A.: How to share a secret? Communications of the ACM **22** (1979) 612–613
13. Blakely, G.R.: Safeguarding cryptographic keys. In: Proceeding of the National Computer Conference. Volume 48., AFIPS Conference Proceedings (1979) 313–317
14. Kothari, S.C.: Generalized linear threshold scheme. In Blakley, G.R., Chaum, D., eds.: Proceedings of Crypto '84, Berlin, Springer-Verlag (1985) 231–241 LNCS Vol. 196.

15. Freivalds, R.: Fast probabilistic algorithms. In Becvár, J., ed.: Proc. LNCS Vol. 74., Olomouc, Czechoslovakia, Springer (1979) 57–69
16. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progresions. In: Proc. Nineteenth Annual ACM Symposium on Theory of Computing, New York (1987) 1–6
17. Karnin, G., Hellman: On secret sharing systems. IEEE Transactions on Information Theory **IT-29** (1983) 35–41

A Modular Architecture for Distributed IDS in MANET

Ricardo S. Puttini[1], Jean-Marc Percher[2], Ludovic Mé[3], Olivier Camp[2], Rafael de Sousa Jr.[1], Cláudia J. Barenco Abbas[1], L. Javier García-Villalba[4]

[1] Department of Electric Engineering, University of Brasilia (UnB)
Campus Universitário Darcy Ribeiro – CP 4386 – Asa Norte
Brasilia – DF – 70919-970, Brazil
puttini@unb.br, desousa@unb.br, barenco@unb.br
[2] École Supérieure d'Électronique de l'Ouest (E.S.E.O.), France
4, rue Merlet de la Boulaye – BP 926, 49009 Angers Cedex 01, France
jean-marc.percher@eseo.fr, olivier.camp@eseo.fr
[3] Supélec, BP 81127, Cesson Sévigné Cedex, France
ludovic.me@supelec.fr
[4] Department of Computer Systems and Programming
Complutense University of Madrid (UCM)
Ciudad Universitaria s/n, 28040 Madrid, Spain
javiergv@sip.ucm.es

Abstract. In this paper we propose a distributed and modular architecture for an intrusion detection system (IDS) dedicated to a mobile ad hoc network (MANET) environment. The main feature of our proposition relies on the use, on each node of the MANET, of a local IDS (LIDS) cooperating with other LIDSes through the use of mobile agents. The modular design is needed as a response to the extensibility requirements related to the complex contexts of MANET. The proposed solution has been validated by a proof-of-concept prototype, which is described in the paper. Two different types of attacks are presented and have been implemented, at the network level and at the application level. The detection of such attacks are formally described by specification of data collection, attack signatures associated with such data and alerts generation, emphasizing the relation of each of these detection steps with the modules in the designed architecture. The use of the management information base (MIB) as a primary data source for the detection process is discussed and modules for MIB data extraction and processing are specified and implemented in the prototype. Experiments exhibit fairly good results, the attacks being collaboratively detected in real-time.

1 Introduction

Mobile ad hoc networks (MANET) are wireless networks in which the mobile nodes exchange information without the help of any predefined network infrastructure [1]. In such networks, also called spontaneous networks, the nodes collaborate with each other to provide the basic network services. For instance, in the case of routing, the lack of a network infrastructure implies that the service is usually provided in a peer-to-peer fashion and that all the nodes of the network need to act as collaborating

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 91-113, 2003.
© Springer-Verlag Berlin Heidelberg 2003

routers. Nodes in a MANET may, at any time, disappear from, appear into or move within the network. The resulting dynamic nature of the network topology, along with the unreliability of the wireless links, require for the configuration of MANET services to be highly adaptable. Moreover, the availability of an individual node cannot be assured and therefore, services cannot rely on a central entity and must be provided in a distributed and adaptative manner [2].

Security services are not an exception to this general rule and the traditional approaches, such as authentication servers or firewall based access control, are usually unsuitable for the MANET context [3]. Moreover, the wireless nature of the access links makes the network more vulnerable to a large range of attacks (e.g. passive eavesdropping, active impersonation, denial of service) than wired networks. These MANET features make the provision of security services in MANETs particularly challenging and tricky to achieve.

Nevertheless, some preliminary solutions for security services in MANET have been presented in recent literature. Most existing solutions consist in some kind of preventive security, such as distributed authentication services for the *ad hoc* context discussed in [3-5] or the use of personal firewalls as a distributed access control mechanism [6]. However, the evaluation of such solutions in terms of efficiency, performance and costs remains an open issue. These preventive security mechanisms can usually be reinforced by proactive security services, such as intrusion detection. Once more, intrusion detection systems (IDS) have specific requirements in MANETs and should not follow a traditional IDS design.

In this paper, we are interested in the design of an IDS for MANETs. This is not a completely new issue, as the subject has already been treated recently [7, 8]. This previous work introduces the basic requirements for such a special type of IDS [7] and discusses some preliminary architecture concepts [8]. Our objective is to contribute to this discussion by presenting a new IDS architecture design, derived from a careful requirements analysis. The requirements considered in our approach are the following: (1) an elementary functional design based on the IETF IDWG (Intrusion Detection Working Group) intrusion detection model [9], (2) distribution of intrusion detection activities among nodes that need to collaborate in the MANET context [8], (3) the interoperability with other IDS or other security resources, and (4) a flexible and adaptable configuration mechanism based on the formal specification of IDS messages and processing rules in external configuration files. The use of mobile agents [20] is the salient feature of our proposition, in response to the distribution requirement. Note that we allow a complete distribution of the intrusion detection tasks and not only of the data collection, which is the common approach in most of existing distributed IDS (a brief survey is given in section 2). The results of our analysis is a global distributed architecture, where the intrusion detection service is provided by local IDS (LIDS) entities, located on each MANET node, and collaborating through the use of mobile agents. Another result is a modular design for each local IDS (LIDS), following the decomposition requirement presented briefly above. The definition and description of such a distributed and modular architecture is the first contribution of our paper.

The second contribution in this paper relates to the implementation of a prototype LIDS enforcing our conceptual architecture. The prototype is used to validate the functionalities and suitability of the proposed IDS model. We point out an important feature of our modular approach, which allows the coexistence of existing software

(from commercial and research software libraries and from previous implementations) and of newly developed modules. The prototype realizes a misuse detection [21] and uses the management information base (MIB) as its primary data source [22]. The use of such type of data in intrusion detection is also emphasized.

Finally, we describe the formal specification of messages and detection rules, which should be supplied as the external configuration of each LIDS. Two examples of complex attack detection are presented: an attack against the MANET routing protocol (network level service) [5] and the stepping stone (session chain) attack (application level service) [23]. In both cases, formal specifications of the messages and rules concerning the intrusion detection process are given.

The rest of this article is organized as follows: Section 2 presents the distribution requirements and the resulting distributed architecture for a MANET IDS. In section 3, the modular LIDS architecture with its requirement conception rationale is discussed. Section 4 is dedicated to the description of the design choices in the practical prototype implementation. In section 5, we present the formalization of messages and rules used for detecting attacks against the MANET routing protocol and the stepping stone attack. Next, in section 6, we analyze the experimental results obtained with the prototype and finally, in section 7, we conclude and point out future directions to explore this field.

2 Toward a Distributed IDS with Mobile Agents

The general objective of an IDS is to identify any violation of the security policy in the monitored system. In order to be able to detect such violations, system and network audit data needs to be collected and analyzed. Upon detection, intrusions should be reported to the security management. Finally, an automatic reactive response (aiming to eliminate the causes and/or effects of the detected violation) may be triggered.

Given the lack of centralization, the mobility of the nodes and the wireless nature of link connections in the MANET environment, some (if not all) of the tasks required for the intrusion detection process described above should be executed in a distributed and cooperative manner [7].

Current literature concerning distributed IDS is not vast. Table 1 presents the existing distributed IDS systems that have reached sufficient maturity in their development to allow the validation of the principles and directions considered in each design.

All the tools presented in Table 1, apart from CSM, EMERALD and SPARTA, are hierarchically organized around a central node. The central node is the kernel of the IDS and uses information collected in a distributed manner to detect intrusions. In such architectures, distribution is restricted to the data collection process and they are, therefore, not suitable on the MANET context.

In contrast, the CMS [16] architecture is completely distributed. A local IDS is installed on each cooperating node to provide a collaborative identification of the originator of network connections. Furthermore, the EMERALD [18] architecture has been designed to suite scalability needs in large networks. This IDS is made up of generic communicative nodes, called monitors, installed on each system. Among the architectures cited above, SPARTA [20] is the only one to have been specifically designed for wireless environments. This system, developed concurrently with ours, is

also based on a mobile agents platform. However, it is designed to detect attacks against distributed applications and, to our best knowledge, it does not consider network layer attacks. In our design, a LIDS is placed on each node of the MANET. The LIDS intercommunicate using a mechanism that takes into account the restrictions resulting from the MANET context; e.g. limited bandwidth or poor connectivity. Such architecture has already been identified in [7] as a basic requirement for an IDS for the MANET environment. To provide a flexible means of interaction and communication between the LIDS we propose to use mobile agents [8]. The proposed distributed architecture is presented in Figure 1.

IDS	Data Source	Detection Method	Distributed Pre-processing	Centralized Detection	Real-time Analysis	Response Type
AAFID [12]	System	Misuse	Yes	Yes	Yes	Passive
DIDS [13]	System Network	Hybrid	Yes	Yes	Yes	Passive
DPEM [14]	System	Anomaly	Yes	Yes	Yes	Passive
Grids [15]	System Network	Hybrid	Yes	Yes	No	Passive
CSM [16]	System	Anomaly	Yes	No	Yes	Active
JiNao [17]	MIB Network	Hybrid	Yes	Yes	Yes	Passive
EMERALD [18]	System Network	Hybrid	Yes	No	Yes	Active
IDA [19]	System	Misuse	Mobile Agents	Yes	Yes	Passive
SPARTA [20]	System Network	Misuse	Mobile Agents	No	Yes	Passive

Table 1. Distributed IDS Survey

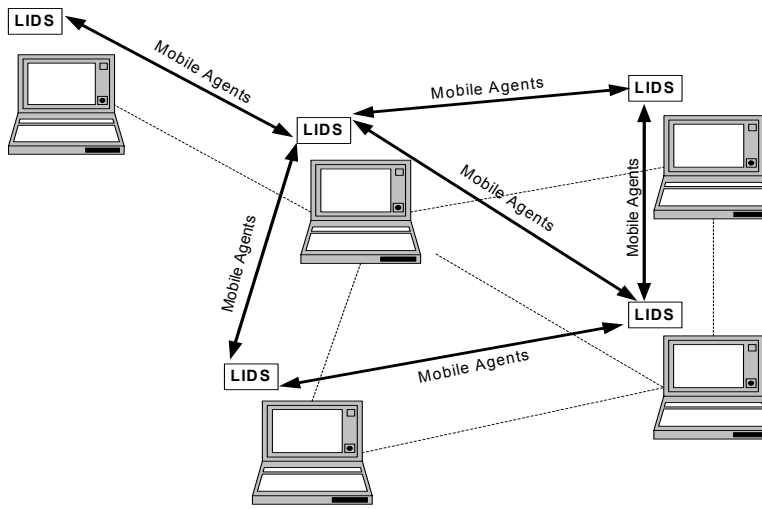


Fig. 1. Global Architecture of a Distributed IDS for MANET

Mobiles agents are an alternative to the client-server distribution model. Their use is particularly interesting in wireless networks where bandwidth is limited and links are unreliable [24]. The use of mobile agents, opposed to traditional approaches where data are transported towards the computation location, allows the code to move to the data. Carefully designed agents can reduce the amount of data exchanged through the network, while providing a flexible way of distributing computation. A node dispatching an agent, may not have to wait for it to return, before it can continue its processing. Moreover, an agent can be dispatched and even destroyed by other nodes, without having to move back to its creating node.

Mobile agents can also provide a first element of response to the problem of the scalability of the global intrusion detection process. Indeed, when a node joins the network, it does so with a running LIDS and a mobile agent platform. It can therefore, immediately take part in the global cooperative intrusion detection process.

3 Modular Architecture Description

Present intrusion detection concerns are usually divided in three main processes: data collection, detection algorithm design and alert management [27]. Figure 2 below shows the model proposed in the requirement specification from the IETF Intrusion Detection Exchange Format Working Group (IDWG)[1] [9]. Each of the elements in the model, **Sensor**, **Analyzer** and **Manager,** are related with one of the intrusion detection processes. More precisely, a Sensor collects data from a data source, an Analyzer processes the collected data for detecting signs of events that might have security concerns and the Manager stands for the management interface of whole process, besides of doing alert correlation and response initiation. The IDWG model also defines messages (or data flows) exchanged by these elements. Thus, activities monitored by

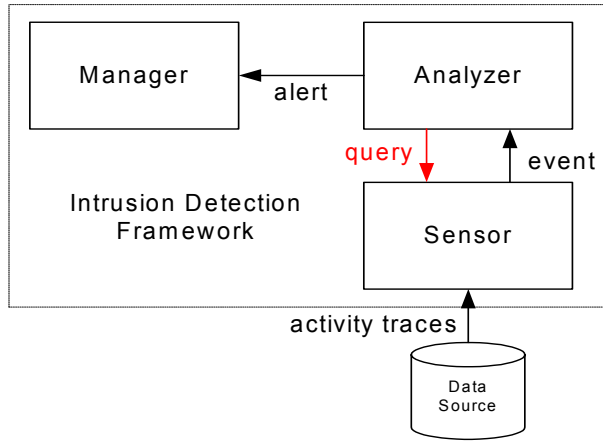


Fig. 2. IDWG IDS Model

[1] Official charter can be found at: http://www.ietf.org/html.charters/idwg-charter.html.

the Sensor in the data source may be mapped in **events**, which are passed to the Analyzer, where they are submitted to the detection algorithm. When the Analyzer finds events with relevant security concerns, **alerts** are generated to the Manager.

The model [9] serves as a baseline for architecture design of any IDS, and is adopted in our design. However, the IDWG model doesn't explicitly specify any message flowing from the Analyzer toward the Sensor. We have introduced a new message type, named **query**, going from Analyzer to Sensor, which is also showed in Figure 2. The new message enables the Analyzer to query the Sensor whenever a more refined data collection is needed. In this section we present the proposed modular architecture for each LIDS used in the distributed IDS, shown in Figure 3. The architecture results from the analysis of the detection process. The analysis proceeded in three phases. First, a basic intrusion detection framework is analyzed by functional decomposition of the IDWG model (Figure 2). Second, the constraints of the MANET context are considered and the requirements for distributed intrusion detection are introduced. The result is the addition to the architecture of the module related with the processing distribution and of the mobile agents platform. Finally, interoperability requirements are considered by addition of a standard communication interface that should be use to interoperate with other IDS systems and with other security entities (e.g. firewall or authentication service) in the intrusion response processes. The following sub-sections detail each one of these steps.

3.1 Intrusion Detection Framework

The modularization of Sensor, Analyzer and Manager elements are presented in this section.

Two important aspects concern the Sensor entity and the data collection process. First, data can be collected from different audit sources, which can be, for example, a network packet capture interface (network level), a log system (host level) or a MIB (network, host and/or application levels). Second, data describing the activity traces (collected by the sensor entity) are usually raw and has poor semantics. The data collected directly from the audit source is hereafter referred as **raw data**. Also, raw data is available in a format that depends on the data source. Some pre-processing is usually applied to this raw data, aiming to translate it into semantically richer information or **events** used by the detection algorithm. This transformation on raw data is referred as the event abstraction. Event abstraction processes can be of many different types, such as a pattern matching based on (domain knowledge) heuristic rules [21], a data-mining technique [25] or a statistical correlation [7, 26].

Different choices for the aspects mentioned above result in distinct scenarios for data collecting. We have decomposed the sensor in two modules: **Event Abstractor** and **Data Collector**. These modules separate the data retrieving and the event abstraction features in two different entities. The idea is to enable multiple implementations for the Data Collector module, which may operate simultaneously in the data collection from different data sources (e.g. network probe, host log or MIB), while enabling the event abstraction process to have abstraction rules that use information originated from multiple data sources. In the same way, we should have implementations of Event Abstractor with different abstraction principles.

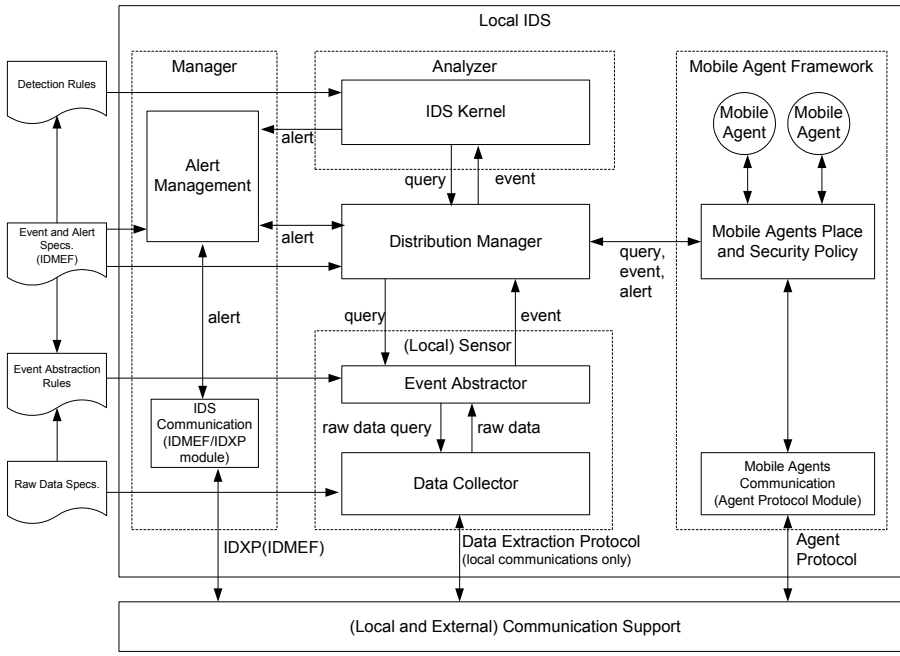


Fig. 3. Proposed Modular Architecture for LIDS

We also define the messages exchanged between the two entities composing the Sensor architecture. Thus, **raw data query** is used by the Event Abstractor to query the Data Collector and **raw data** supplies the Event Abstractor Entity with formatted data coming from the data source.

The Analyzer processes the events according to some defined detection strategy. At least two detection methodologies are currently in discussion: misuse and anomaly detection [27]. It seems to us that both methodologies have complementarities. It is our goal to have a hybrid (misuse and anomaly detection) intrusion detection strategy, as such design should be able to deal with a more large spectrum of attacks. Once more, the modular architecture is important to enable independent development of detection modules for each approach. In our architecture, each detection algorithm implementation is encapsulated in an **IDS Kernel** module, and it is possible to have multiples instances of such module, each one with specific detection algorithms.

Finally, the **Alert Manager** module is designed to accomplish with the alert management tasks, such as alert interpretation and validation (false positive elimination, usually by alert correlation).

3.2 Distribution Requirements

As we have argued in section 2, a distributed IDS architecture is required for *ad hoc* networks. In almost all distributed IDS architectures that have been evaluated, the distribution is achieved by distributing the data collection process, which is, almost invariably, executed locally. In IDS for ad hoc network this is highly recommended, as

the limited bandwidth requires that communications related with LIDS information exchanges are kept to a minimum. The remote collection of important volumes of raw data is prohibitive. In our design, data collection and event abstraction is always kept local. This means that all raw data collection and pre-processing is performed locally, in the same LIDS. Thus, data being exchanged is restricted to concise information (events) resulted from locally pre-processed of raw data.

In our design, when a LIDS needs to recover information from other node, it queries the remote node by sending a query message. This request is traduced in the destination node and the raw data related to the query is collected locally. Then, the data is processed and summarized as an event that is sent back to the querying node. Alternatively, the LIDS should decide if a locally generated event could be useful to a remote node, even if the LIDS hasn't been explicitly queried for the occurrence of such event by the remote node. In such case, the event should be dispatched to the remote node interested in receiving it. In any case, only high-level messages (queries, events and alerts) are exchanged. No raw data is transmitted in the network.

Besides of local data collection, we want to make a complete distribution of IDS tasks, enabling intrusion detection and alert management to be equally realized in a distributed manner. The objective is to enable nodes in the *ad hoc* network to detect attacks against some of its neighbors (and not only against itself) while executing all raw data collection and abstraction locally. Nodes should cooperate to provide alert correlation and attack response, as well.

We propose that distribution should be accomplished by means of mobile agents. Thus, whenever some (high level) message needs to be processed remotely, a mobile agent is dispatched to the remote node carrying the data and, possibly, the code needed for the remote message processing. The use of mobile agents enables the mobility of code related with message processing, making the distribution in a flexible and convenient way.

Mobile Agents are created, received (from a remote host) and managed in the **Mobile Agent Framework**. This mobile agent platform should also provide security services (e.g. server authentication, agent and server code integrity, access control to local resources, etc.) related to agent activities [24]. Thus, a Mobile Agent Place module implements the directives to create, dispatch and receive mobile agents while a Security Police module implements the required security services. The mobile agent communication protocol is also implemented as a separate module, the **Agent Protocol Module**. The communication overhead imposed by of the agent protocol is an important design feature. The protocol should remain as lightweight as possible, while maintaining the communication reliable and providing basic security services.

Finally, we introduce a **Distribution Manager** module. All high level messages generated locally or received from remote nodes are sent this module, which is responsible for deciding if the message should be consumed locally or if it should be dispatched inside a mobile agent to execute some processing in a remote node. The Distribution Manager also maintains a library with all types of alert, event and query, which is available for the other modules.

3.3 Interoperability with Other Security Devices and IDS

The last addition into our proposed architecture lies on the interoperability of the IDS. Thus, each LIDS should be able to communicate with other IDS in a standardized manner. The objective of such interaction can be information sharing and correlation. Also, especial interest should be giving in the interaction with devices that are related with preventive security services, such as authentication and resource access control. Such interaction is important in the definition and implementation of automatic response to intrusion being detected. As an example, consider that a fully compromised node[2] is used to launch an attack against some network entity (node or service). If the attack is detected and its source is identified, the detecting IDS should communicate with the access control service, which should be reconfigured to deny access to the compromised node. Also, some interaction with authentication service would request the revocation of the compromised private key or certificate.

Although some level of automatic response is possible, as shown by the above example, the collaboration among security devices and even among different IDS is yet an open issue. However, some effort has already been done in this direction. Especially, the IETF IDWG is currently discussing standard data format (IDMEF [10]) and communication protocol (IDXP [11]) related with intrusion detection processes and system collaboration. These standards will be important in the definition of collaboration extension for many IDS and security services, as they can create a compatibility level between them. For this reason, we have previewed in our architecture design an **IDS Communication** module as part of the Manager. The applicability of such module is yet conceptual, as the standardization process is not finished. This should be a general purpose IDMEF/IDXM communication module that should be used to alert exchange and interoperation with other IDS, especially for cooperative attack detection and response.

3.4 Configuration Information

For flexibility and adaptability, LIDS should be configured by some configuration information (files). In the proposed architecture, external configuration should be provided as:

- Detection Rules: configuration information containing a set of detection rules and parameter associated with the behavior of the IDS Kernel module as well as the formal relation between high-level messages (query, event, alert) and the detection process.
- Event and Alert Specification: configuration information containing the formal specification of all types of alerts, events and respective queries. This information is coded in the IDMEF [10], which relies in the XML standard.
- Event Abstraction Rules: configuration information containing the formal specification (e.g. in some procedural language) of the abstraction event processes. This should relate to the formal specifications of events, queries, raw data queries and raw data.

[2] A node whose private key or any other identity protection have been exposed to the intruder.

– Raw Data Specification: configuration information containing the formal specification (e.g. in a previous syntax definition language such as ASN1) of the raw data queries and raw data being collected.

The modules affected by the configuration information should have the respective parsers included in their feature.

4 Implementation Description

For validation of the proposed architecture, a prototype implementation was built, using some specific features in the design of each module, which are discussed in this section.

4.1 Analyzer

The IDS Kernel module implements a misuse intrusion detection strategy. Attack signatures should be supplied for all attacks that are aimed to be detected. Signatures are described as finite state diagrams (FSD) [21], which are composed by a finite number of states with defined transitions between then. Each transition is executed as a consequence of one event being triggered. Whenever passing from one state to another (e.g. in a transition), an appropriate action can be activated, which may result in generation of one or more messages among: (1) new (more specific) queries, (2) new events or (3) an alert, the last one in case of positive attack detection.

Event triggering is related to some particular entity (nodes or service) that has been targeted by an attack (target). The LIDS detecting the attack (not necessarily placed in the node targeted in the attack) is always signaled in the event/alert generation (originator). Whenever possible, the target of the attack and the entity that originates the attack (attacker) is also signaled by the detection procedure.

4.2 Sensor

In the present implementation, intrusion detection is exclusively based in data collected from the Management Information Base (MIB), maintained locally in each *ad hoc* network node by an SNMP agent. Raw Data is extracted directly from the local MIB by simple SNMP queries (raw query $\Rightarrow$ MIB query and raw data $\Rightarrow$ MIB data). The use of this data source is doubled justified. First, because raw data and raw data retrieving has standardized format and process respectively given by the MIB format, which is coded in ASN1, and the SNMP. Moreover, these standard are extensively implemented and adopted in a variety of platforms, providing good portability for the IDS designed in such way. Second, MIB information describes entities of different layers of the node architecture. Thus, we should be able to monitor simultaneously network, host and even application levels, as exemplified in the next section.

In such design, the Data Collector module is simply a software module that can execute local SNMP queries to the SNMP agent. This module is recalled **MIB Browser**. Following the same rationale, the Event Abstractor module is recalled **MIB**

Event Abstractor, as all processing executed in the module is based in data provided from the local MIB.

Concerning MIB data collection (query execution), each high level query should be mapped into respective MIB variables to be queried (**MIB query**[3]). Mapping between queries and MIB variables to be evaluated is supplied in the configuration file providing query specification. When scheduled by the Distribution Manager, the query is mapped into the respective MIB query in the MIB Event Abstractor, which is executed by the MIB Browser. The resulting raw data (**MIB data**) corresponds to the value(s) of the queried MIB variables at the moment of the MIB query execution.

Data collection is accomplished by explicitly querying the data source. Two types of queries are used: queries that are to be executed only once and are trigged by a transition in the finite state machine, as described above, and queries that should be executed periodically[4] by the sensor elements.

4.3 Configuration Information

The configuration information for the prototype is the following:

- Detection Rules: consist of attack signatures provided as a FSD for each attack.
- Event and Alert specifications: specifications of alerts, events and respective queries, coded in IDMEF.
- Event Abstraction Rules: specification of mapping (high level) queries into MIB variables to be collected and of rules for mapping collected data (may include simple processing of collected data and temporary store of local data) into (high level) events.
- Raw Data Specifications: consist of proprietary and standard MIB definitions.

5 Attacks Description and Signatures Specification

In this section we provide examples of complete signature specification (including high level and low level message specifications) for two types of attacks. These signatures have been loaded in the implemented prototype and the detection of the de-

[3] MIB query contains all information needed to perform the data collection in the local MIB. Specially, the whole data to be collected (MIB data) is specified as a list of MIB variables to be collected as an ensemble of basic SNMPv1 operation (get, for single variable retrieving or getnext for table retrieving). Each of these basic query to the MIB is specified by its unique OID. A "type" attribute specifies if the data gathering is to be realized with a unique "get" operation (type =MIBVariable) or successive "getnext" operation (type=MIBTable).

[4] This is implemented rather as special queries (scheduled to be executed systematically within a defined period of time) managed by the Distribution Manager than by the Data Collector module. This design choice is mainly related with the SNMP agent that maintain the relevant MIB information up-to-date, used in the prototype. Thus, the IDS MIB Data Collector module always queries the SNMP agent (with GET and GETNEXT SNMP messages), instead of being trapped (TRAP SNMP message) by it.

scribed attacks was possible, as discussed in the next section. The attacks are placed at different levels in the node architecture.

The first attack described here is an attack against the MANET routing protocol (network level). This attack has basically denial of service (DoS) effects at the target node, and so, must be detected by some of the target's neighbors running the LIDS. The detection of such attack with the presented attack signature is completely done locally, so the distribution functions of the architecture are not used. As there is no presently MANET routing protocol standard[5], an experimental MIB and the respective support for it in the SNMP agent have been developed.

The second attack described is a stepping stone attack [23] (application level). All nodes participating in the session chain should be running the LIDS and the attack detection is done collaboratively. Information is collected directly from standard MIB-II [27] variables, and no experimental MIB or SNMP agent patch were needed for this case.

5.1 Attack against the Routing Protocol

The most common types of attack against routing protocols are related with the generation of bogus routing information. This is the case for modification (malicious modification of routing protocol messages), impersonation (masquerading as another node) or fabrication (generation of false routing messages) attacks. It is usually possible to identify attack signatures from the traces of abnormal behavior of malicious node (attacker), such as anomalies in the protocol scheduling (in the case of impersonation attacks) or inconsistencies in the routing tables and paths maintained by different nodes (in the case of modification and fabrication attacks) [5].

The attack described concerns the Optimized Link State Routing protocol (OLSR) [29], one of the MANET routing protocols that is being considered to standardization by IETF.

5.1.1 OLSR Background

OLSR [29] operates as a table driven and proactive routing protocol, which means that it is based on regular exchange of network topology information between nodes. The topological information is used for updating the routing table of participating nodes, by the means of a link-state routing algorithm. The routing metric is always hop-distance. Thus, the protocol gives minimum hop distance routing when the network is in a stable state. Optimization over a pure link state algorithm is obtained by reducing the size of control messages and minimizing flooding of control traffic (flooding is executed only by some selected nodes, called MPR, Multipoint Relay).

HELLO messages are used by nodes to detect and update their neighbor set. Each node periodically broadcasts HELLO messages, containing information about heard neighbor interfaces and their link status. The link status may either be "symmetric" (link has been verified to be symmetrical), "heard" (link is asymmetrical), "mpr" (a node is selected by the sender as a MPR and the link must be symmetric) or "lost"

[5] Standardization process has been carried on by the IETF MANET working group. http://www.ietf.org/html.charters/manet-charter.html

(indicates that the link with this neighbor interface is now lost). HELLO messages are periodically broadcasted from all nodes to all one-hop neighbors.

Routing through a neighbor node is possible only if there is a symmetric link between sender and receiver. Thus, making the status of a link "heard" intentionally should stop routing through that link.

5.1.2 OLSR Protocol Attack (NHOP)

One attacker should generate spoofed HELLO messages, leading neighbor nodes to have the status of their link with some target node (which is being spoofed) changed to "heard" (asymmetric). Figure 4 describes the attack. Note that attack effects will persist as long as the attacker generates a fake message for each legitimate received message. Before attack, nodes A and B have learned a "symmetric" link:

1. Node A broadcasts a Hello message announcing B with "symmetric" link status.
2. After receiving the HELLO message from A, the attacker sends a fake message spoofing A announcing B with "lost" link status. After receiving the fake message, B changes the status of the link with A to "heard". Symmetry is lost and B stops routing traffic through A. A correct HELLO message will be send by A only a HELLO_INTERVAL period after the HELLO message transmission of step 1.
3. B broadcasts a HELLO message announcing A with "heard" link status.

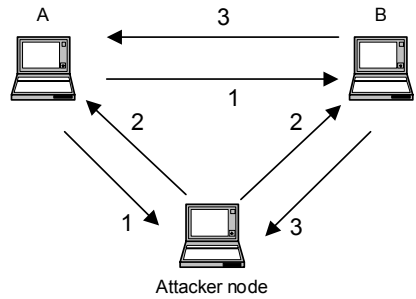


Fig. 4. DoS by Faking Hello Messages

5.1.3 Attack Signatures

The attack can be detected by the anomalies in the OLSR scheduling during a HELLO_INTERVAL period, which is set to be the data collection period. Therefore, in-between such interval, one of the following can happen (for each neighbor): (1) No information has been received in HELLO_MESSAGE (during normal operation), (2) One update has been received in HELLO_MESSAGE (during normal operation), or (3) Two updates has been received in HELLO_MESSAGES and one of them must advertise the "lost" state (during attack).

Thus, one of the following events should happen at each HELLO_INTERVAL period[6]:

- NHOP_E0: Link type has not been "heard" in the last HELLO_INTERVAL period (no "lost" update, and no attack occurring).
- NHOP_E1: Link type has been "heard" and "symmetric" or "mpr" in one or more HELLO_INTERVAL period ("lost" update has been received).
- NHOP_E2: Link type has been only "heard" for more than or equal one HELLO_INTERVAL period (only "lost" update received, no attack occurring).

To consider the abstraction of these events, we must define what information must be kept in the designed experimental MIB. This information is collected whenever a query is generated. From the event definition we can deduce straightforward that this information should be related to the values the link type variable have assumed. Recalling that queries are generated at each HELLO_INTERVAL and knowing that link status can change twice during this sampling period (whenever an attack is taking place), we have designed an experimental MIB that contains a "olsrNeighborTable" variable, with the following information for each neighbor of an OLSR instance (daemon):

- olsrNeighborAddress (integer, 32 bits): IP address of neighbor.
- olsrNeighborState (integer, 16 bits): current status of neighbor ("heard", "symmetric", "mpr" or "lost").
- olsrPreviousNeighborState (integer, 16 bits): status of neighbor before receiving of last HELLO message.

The use of both olsrNeighborState and olsrPreviousNeighborState assures that no information will be lost in the values assumed by the link status variable during the query execution period. The events can be abstracted in function of the collected information by the abstraction rules bellow:

IF (olsrNeighborState != ASYM_LINK) && (olsrPreviousNeighborState != ASYM_LINK)
  NHOP_E0
ELSEIF (olsrNeighborState == ASYM_LINK) && (olsrPreviousNeighborState == ASYM_LINK)
  NHOP_E2
ELSEIF (olsrNeighborState == ASYM_LINK) || (olsrPreviousNeighborState == ASYM_LINK)
  NHOP_E1
ENDIF

As the information needed in all events of interest are the same, only one periodical query (NHOP_Q0) is locally executed. The formal (simplified) definition of (NHOP_Q0) is:

[6] Events are defined for each possible target from the attack. Due to a space limitation, only a simplified definition is supplied for all messages (events, queries, alerts, MIB queries and MIB data), containing the ID and description of each one. We admit that, during the attack, the attacker will generate only one fake message for each HELLO message received from the target node, e.g., the attacker will generate only one message at each HELLO_INTERVAL.

– NHOP_Q0: query for NHOP_E0, NHOP_E1 and NHOP_E2, executed as NHOP_MIBQ0.

The MIB query (NHOP_MIBQ0) associated with this high level query (NHOP_Q0) is defined by the MIB variable olsrNeighborTable, specifying a query for the MIB table represented by the OID of the MIB variable. The formal (simplified) definition of the MIB query and respective MIB data (NHOP_MIBD0) are:

– NHOP_MIBQ0: OID of olsrNeighborTable.
– NHOP_MIBD0: values (table) of MIB variable specified by OID of olsrNeighborTable.

The alert to be generated is similarly defined as:

– NHOP_Attack: N_HOP attack has been detected.

Figure 5 presents the automata for the attack signature. Initially (state NHOP_S0), the detecting node hadn't received any advertisement from the target node with a "lost" type. When receiving one of such advertisement (event NHOP_E1), there is a transition to state NHOP_S1. Before raising an alert, the detecting node must be sure that the received advertisement is not a normal one. If only "lost" is received, NHOP_E2 is generated again and the behavior is considered normal. However if other NHOP_E1 is received, it is possible to be sure that the link type is changing twice during one HELLO_INTERVAL time, thus characterizing the attack. Any occurrence of NHOP_E2 indicates that there is no attack and the symmetry is established, resulting in transitions to the NHOP_S0 state.

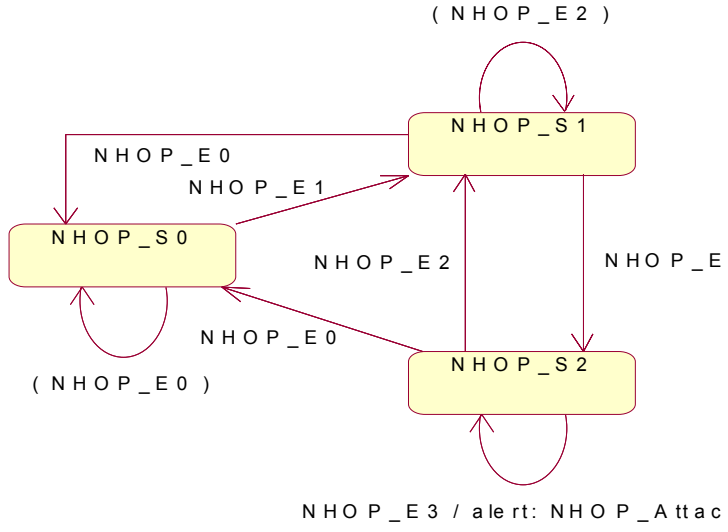


Fig. 5. NHOP Attack Signature

5.2 Stepping Stone Attack

The stepping stone attack corresponds to creating of a connection chain (e.g. telnet). This type of attack precedes often some more evasive or destructive actions as it represents increased difficulty for tracking the attack source [23].

Figure 6 illustrates the attack. The root node (attacker) start a telnet connection with some other node (node 2) and, from there, to another one (node 3), and so on, building the whole chain until the target node.

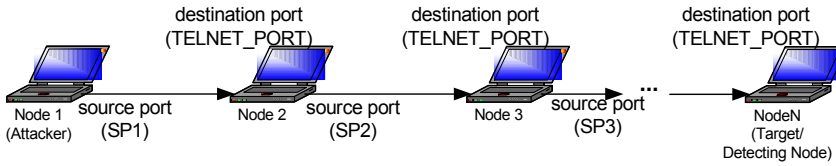


Fig. 6. Stepping Stone Attack (Telnet Chain)

5.2.1 Attack Signature

The detection of a telnet chain will be divided in two steps. First, whenever one node receives an incoming telnet connection (local event STEPSTONE_E0), this node should query the calling node about its incoming telnet connections. If the calling node has no incoming telnet connection, it will generate an event to the querying node telling that there it is the root of the telnet chain (remote event STEPSTONE_E1). The node receiving such event should create the recently formed chain, composed by exactly two nodes (the calling node and itself) and having the calling node as root of the chain. The source address (IP address and port number) of the calling node is stored as the root of the chain path. Otherwise, the node will generate an event (remote event STEPSTONE_E2) for each active chain it stores, informing the querying node the path of the chain. Whenever receiving one of such event, the node should create itself a chain formed by the same chain path in the received event with the source addresses of the incoming connection added in the end of the path. Chains are excluded whenever the incoming telnet connection from its last node is closed (local event STEPSTONE_E3). The reception of STEPSTONE_E2 event characterizes the detection of a possible telnet chain attack (formed by, at least, three nodes) and finishes the first step of detection.

The missing point is that there is no ensured relation between telnet connections in the path of each chain. The second step in the stepping stone attack detection consists in evaluating the existence of such relations (e.g. showing that one telnet connection in the chain path was indeed generated inside the section of the telnet connection in the precedent hop of the telnet chain). Identifying such relation is not a straightforward task. As we are rather interested in presenting the rationale of attack signature identification, for space limitation reasons, we'll describe the attack signature for the first step only. One possible solution to accomplish the second step of detection in our design would be the use of a statistical correlator based event abstraction such as in [30].

From the above description, we should have one local query executed periodically (STEPSTONE_Q0), which results in one of the two possible corresponding local events (STEPSTONE_E0 or STEPSTONE_E3). One remote query (STEPSTONE_Q1) is also defined to look for occurrences of the remote events (STEPSTONE_E1 and STEPSTONE_E2). This query is not executed periodically but triggered by a state transition whenever an STEPSTONE_E0 event is received.

Information collected from the MIB in both queries (STEPSTONE_Q0 and STEPSTONE_Q1) is the table with active TCP connections, which is defined as a single MIB query, SETSTONE_MIBQ0, expressed by the OID representing "tcpConnTable", a standard MIB variable [28]. This MIB table contains the following information, for each active TCP connection:

- tcpConnLocalAddress (32 bit integer): local IP Address
- tcpConnLocalPort (16 bit integer): local TCP port
- tcpConnRemAddress (32 bit integer): remote IP Address
- tcpConnRemPort (16 bit integer): remote TCP port

Note that, besides of information retrieved from the MIB at each query, the Event Abstractor must keep locally some additional information, e.g. the list of all chains the node is participating in. From information kept locally and new information collected from the MIB query, each of the events can be easily abstracted.

Finally, the finite state diagram representing the attack signature (for each telnet chain) is showed in the Figure 7 below.

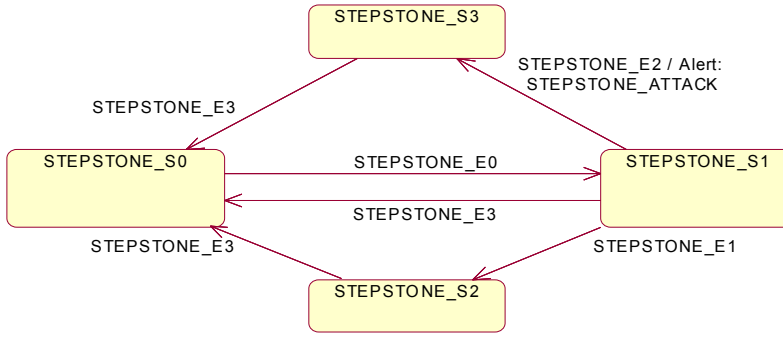


Fig. 7. Stepping Stone Attack Signature (Telnet Chain)

6 Results and Analysis

We have implemented a prototype of the local intrusion detection system specified in section 4. The design was realized using an object-oriented conception and was implemented using the Java2 language. In the prototype design, the modules **MIB Event Abstractor**, **Distribution Manager**, **IDS Kernel**, **Alert Manager** and **Mobile Agent Framework** were implemented as individual threads, providing relative independence in the scheduling of the tasks related with each module. This design choice is re-

lated with the requirement of low coupling between the modules. The MIB Browser module was implemented using a simple SNMP java library, available in [30]. The **Mobile Agent Framework** was implemented using the IBM *Aglet* platform [31]. A special SNMP agent was needed, to define the experimental MIB defined for the OLSR protocol. This agent was implemented as an add-on module to the general-purpose agent proposed by the NetSNMP project [32]. The daemon implementing the OLSR, the MANET routing protocol used, was based on the only publicly available implementation of the protocol, which is conform to version 3 of the IETF draft (the last IETF draft version is 7 [29]). Since the SNMP agent and the OLSR daemon are executed as separated processes, the OLSR daemon was also extended to provide process intercommunication with the SNMP agent. The NHOP attack was designed as three distinct modules: (1) a packet capture module, developed using the libpcap library (a packet capture library developed as part of the tcpdump project [33]), (2) an OLSR parsing module, implemented using some code from the OLSR daemon, and (3) a packet generation module, which was realized using standard UNIX-like socket API, by creation of *raw sockets* [34].

A MANET with six Linux nodes was used in our experiments, each of them executing the LIDS prototype, an instance of the extended SNMP agent and of the OLSR daemon. An additional node, consisting of a PDA running Linux was used to generate both attacks described in section 5. Figure 8 illustrates one of the topologies assumed by the MANET network during the NHOP attack execution, its DoS effects (nodes B and C are unreachable by D, E and F, as the attacked node A is the only available path to them). Nodes D, E and F detects the NHOP attack against node A. Nodes B and C detects the stepping stone attack coming from the PDA, accusing the PDA IP address as the attack originator.

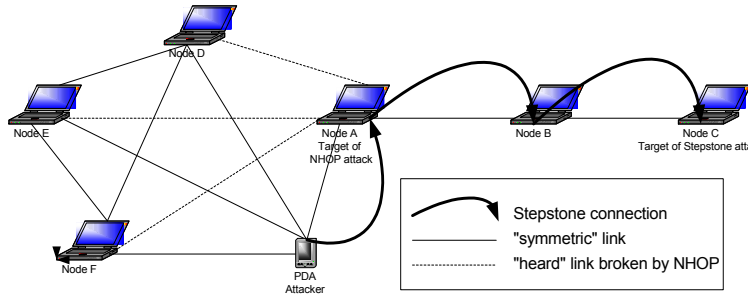


Fig. 8. Experimentation Platform

The two attacks have been played and detected. In the first one (NHOP), the detection is always local and no mobile agent is dispatched. In the second (STEPSTONE), the telnet chain was detected collaboratively by dispatching mobile agents from the node receiving the incoming telnet connection toward the node initializing each telnet section in the chain. The detection of these attacks intends to provide a functional and feasibility validation for our design, which is shown to be flexible in detecting attacks by different methods (locally and collaboratively) and in different levels (network and application) of a MANET node architecture. The choice of the MIB as a data source

is not original [22]. The contribution of our approach for this subject is, however, the use of the semantically rich information in the MIB to enable detection of different kinds of attacks on multiple layers (network, system and application). Moreover, standard MIB information is now largely implemented in the many available SNMP agents making MIB data a particularly standard data source. Finally, an SNMP agent needs to know the formal specification of the MIB it uses. Such a specification can simplify the formal specification of event abstraction and of detection rules. The designed IDS seems also to have good scalability, as the intrusion detection process is not distributed in the entire MANET but rather in a small number of nodes, usually located near each other. Another important factor to consider when analyzing the scalability and performance of the LIDS concerns the mobile agent platform. We have noticed that the Aglet platform is quite "heavy" for the purpose of the designed LIDS, as it contains lots of features that won't be used in the LIDS. The specification of a "lightweight" intrusion-detection customized agent platform should be an important requirement. We are now carrying out a precise evaluation of the scalability and performances of our architecture and will publish our results in a future paper.

Concerning the formal specification of high-level messages (events, alerts and queries), some important results are being obtained by the standardization process of the IDMEF and IDXP protocols, by the IETF IDWG [9-11]. However, the process has not reached finalization and, the instability of the existing drafts do not allow these technologies to be deployed in a large scale on security devices. It should be noted that, high-level messages between LIDS do not need to use the IDMEF format, as this would require each message to be parsed before being processed. Finally, we are also working on the implementation of an **IDS Kernel** module based on anomaly detection. The anomaly detection module will l be aggregated to the prototype, and thus provide a hybrid anomaly/misuse detection approach. We expect that, the complementarities between these approaches will broaden the spectrum of attacks detected by the system.

7 Conclusions

In this paper, we proposed a new distributed and modular architecture for MANET IDSes. The proposed model was validated by a prototype implementation and a formal description of IDS messages and rules concerning two different attacks. The experimental results presented have demonstrated both the feasibility and the efficiency of our IDS. Moreover, our experimentation shows that the proposed architecture is perfectly suited to the MANET context. A complete prototyping cycle has been realized and the results obtained are encouraging.

We are currently considering performance aspects, which depend mainly on the design of a lightweight mobile agent platform, customized for the needs of the IDS. Further extensions to the implemented prototype include the design of an anomaly based IDS Kernel module and a full-featured implementation of the IDMEF and IDXP, which are still in draft form.

Acknowledgments

The authors would like to thanks partners from the RAHMS (Réseau Ad Hoc Multiservice Sécurisé) RNRT project (funded by the French Ministry of Industry) and CAPES (Brazilian government program for research support), which funded Ricardo Puttini during his stay in France.

Javier García is supported by the Spanish Ministry of Science and Technology (MCYT, Spain) under Projects TIC2002-04516-C03-03 and TIC2000-0735. This author would like to express his appreciation to the Programa Complutense del Amo for providing him a grant to stay at IBM Research Division. During this work he was with the Information Storage Group at the IBM Almaden Research Center, San Jose, California, USA (javiervi@almaden.ibm.com).

References

1. S. Corson and J. Marker – Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation consideration. RFC 2501 (informational), IETF, 1999.
2. Y. Chun, L. Qin, L. Yong and Shi MeiLin – Routing protocols overview and design issues for self-organized network. *Proceedings of IEEE International Conference on Communication Technology–(ICCT 2000)*, pp. 1298-1303, 2000.
3. L. Zhou and Z. J. Haas - Securing ad hoc networks. IEEE Network, Vol. 13, Nov.-Dec. 1999, pp. 24 –30, 1999.
4. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang - Self-securing Ad HocWireless Networks, *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, 2002.
5. Reference anonymized for the review process.
6. T. Droste - Weighted communication in a security compound,. *Proceedings of the 5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service, 2001 (TELSIKS 2001)*, pp. 463-466, vol.2, Yugoslavia, Sept. 2001.
7. Y. Zhang and W. Lee – Intrusion detection in wireless ad hoc networks. *Proceedings of 6th Annual International Conference on Mobile Computing and Networking, MOBICOM 2000*, ACM, ACM Press New York, pp. 275-283, 2000.
8. Reference anonymized for the review process.
9. Wood and Erlinger, "Intrusion detection message exchange requirements". IETF Internet draft. June 2002.
10. D. Curry, H. Debar, and Merrill Lynch - Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML). IETF Internet draft. June 2002.
11. B. Feinstein, G. Matthews, and J. White - The Intrusion Detection Exchange Protocol (IDXP). IETF Internet Draft. October 2002.
12. J. Balasubraniyan, J. Fernandez, D. Isacoff, E. Spafford, D. Zamboni - AAFID - Autonomous Agents For Intrusion Detection, Technical report 98/05, COAST Laboratory Purdue University, June 1998.
13. Steven R. Snapp, , James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukkherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur - DIDS-Distributed Intrusion Detection System, Computer Security Laboratory, Department of Computer Science, University of California, Davis, June 1992.

14. C. Ko, M. Ruschitzka, and K. Levitt. Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach. Proceedings of the 1997 IEEE Symposium on Security and Privacy, 1997.
15. S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle - GrIDS- A Graph Based Intrusion Detection System for Large Networks, Computer Security Laboratory,Department of Computer Science,University of California, Davis, 1996.
16. Gregory B White, Eric A. Fish and Udo Pooch - CSM - Cooperating Security Managers : a peer based intrusion detection system, IEEE Networks, pages 20-23, January/February 1996.
17. Y. F Fou, F. Gong, C. Sargor, X. Wu, S. F. Wu, H. C. Chang, F. Wang - JINAO-Design and Implementation of a Scalable Intrusion Detection System for the OSPF Routing Protocol, Advanced Networking Research, MCNC Computer Science Dept, NC State University, February, 1999.
18. Phillip A. Porras, Peter G. Neumann - EMERAL - Event Monitoring Enabling Responses to Anomalous Live Disturbances, Conceptual Overview, December, 1996.
19. Midori Asaka, Atsushi Taguchi, Shigeki Goto - IDA-The Implementation of IDA : An Intrusion Detection Agent System, IPA Waseda University, 1999.
20. Christopher Krügel, Thomas Toth - Flexible, Mobile Agent Based Intrusion Detection for Dynamic Networks, Distributed Systems Group, Technical University Vienna, 2002.
21. K. Ilgun, R. A. Kemmerer, and P. A. Porras – State Transition Analysis: A Rule-Based Intrusion Detection Approach. IEEE Transactions on Software Engineering. pp 181-199.March 1995.
22. J. Cabrera, L. Lewis, R. Prasanth, X. Qin, W. Lee, and R. Mehra - Proactive detection of distributed denial of service attacks using MIB traffic variables – a feasibility study, in *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, WA, USA, may 2001.
23. S. Satinford-Chen, and L. Heberlein – Holding Intruders Accountable on the Internet. *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, 1995.
24. S. Martino - A mobile agent approach to intrusion detection, technical report, Joint Research Centre Institute for Systems, Informatics and Safety, Italy, June 1999.
25. W. Lee; S. J. Stolfo; and K. W. Mok - A data mining framework for building intrusion detection models. *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.
26. Reference anonymized for the review process.
27. H. Debar, M. Dacier and A. Wespi - A Revised Taxonomy for Intrusion-Detection Systems, IBM Research Report, Zurich, 1999.
28. K. McCloghrie; and A. Bierman - Entity MIB (Version 2), IETF Request for Comment 2737, December 1999.
29. T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot - Optimized Link State Routing Protocol - IETF draft, MANET working group, version 7, July 2002.
30. http://edge.mcs.drexel.edu/GICL/people/sevy/snmp/snmp_package.html.
31. J. Kiniry and D. Zimmerman - Special Feature: A Hands-On Look at Java Mobile Agents, IEEE Internet Computing, Vol. 1, No. 4, July/August 1997.
32. http://www.netsnmp.org.
33. http://www.tcpdump.org.
34. D. Comer; and D. L. - Internetworking with TCP/IP, Vol. 3: Client-Server Programming and Applications, Linux/Posix Sockets Version LINUX/POSI, Prentice-Hall, 2000.

Appendix A ASN1 OLSR Experimental MIB Specification

```
RAHMS-OLSR-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE
    experimental, IpAddress        FROM SNMPv2-SMI ;

rahmsOlsrMIB MODULE-IDENTITY
    LAST-UPDATED "0207051145Z"
    ORGANIZATION "ESEO"
    CONTACT-INFO "rahms@eseo.fr"
    DESCRIPTION  "The MIB module for RAHMS networks"
    ::= { experimental 6060 }

rahms     OBJECT IDENTIFIER ::= { experimental 6262 }

-- the olsr group
olsr      OBJECT IDENTIFIER ::= { rahms 1 }

-- OLSR Neighbor Table
-- The OLSR Neighbor Table contains information
concerning this entity's existing neighbors and
the status of the link between this host and
each of its neighbors

olsrNeighborTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF olsrNeighborEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "A table containing OLSR neighbor
                information”
    ::= { olsr 1 }

olsrNeighborEntry OBJECT-TYPE
    SYNTAX      olsrNeighborEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "A conceptual row of the
                olsrNeighborTable containing
                Information about the connection
                towards a particular OLSR neighbor.
                Each row of this table is transient, in
                that it ceases to exist when (or soon
                after) the connection with a neighbor
                is lost"
    INDEX   { olsrNeighborAddress}
    ::= { olsrNeighborTable 1 }
```

```
olsrNeighborEntry ::= SEQUENCE {
        olsrNeighborState     INTEGER,
        olsrPreviousNeighborState INTEGER,
        olsrNeighborAddress   IpAddress
    }

olsrNeighborState OBJECT-TYPE
    SYNTAX     INTEGER { ASYM(1),SYM(2),MPR(3),LOST(4)}
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "The state of this OLSR neighbor
                connection"
    ::= { olsrNeighborEntry 1 }

olsrPreviousNeighborState OBJECT-TYPE
    SYNTAX     INTEGER {ASYM(1),SYM(2), MPR(3),LOST(4)}
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "The previous state of this OLSR
                neighbor connection"
    ::= { olsrNeighborEntry 2 }

olsrNeighborAddress OBJECT-TYPE
    SYNTAX      IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "This neighbor IP address"
    ::= { olsrNeighborEntry 3 }
END
```

Characterization of the Security Implications of Alternative E-procurement Topologies

E. Ponce; A. Dur□n

University Carlos III of Madrid.
Avda. Universidad 30, 28911, Legan□s, Madrid, Spain.
eponce@ing.uc3m.es

Abstract. In this communication we propose a classification of alternative e-procurement topologies that can be characterized in terms of their security implications. In order to establish these alternative topologies we consider how the two main participants (buyer and supplier) could access to the e-procurement solution (i.e. through the Internet or through a Virtual Private Network □VPN-). From the point of view of the buyer, we consider two main possibilities, to use a third party intermediary that hosts the solution or to host it in the buyer information system. In the first case, we describe different situations when a third party intermediary is used, and we characterize the security implications in each case. Finally, we use this classification of topologies to analyze the security implications of different e-procurement solutions offered by *Opciona* (Spanish e-procurement solution provider).

1 Introduction

The advent of Internet has prompted numerous changes in the way companies do business; particularly these changes have affected the technological solutions developed for the inter-organizational relationships (also known as Business to Business, B2B) that supports the procurement process. The use of these solutions allow the reduction of time and costs for the procurement process, and what is even more important, allow the organization to develop new strategic procurement capacities due to the simplification of administrative duties.

However, when companies use solutions based on the Internet, they feel more comfortable if the information exchanged is related to non-production materials (for example, parts for maintenance, repair and operations materials □MRO-). In case they need to exchange more sensitive information (related to direct materials), companies are usually reticent to Internet based solutions. We consider one of the main reasons for this behavior is the lack of reliability and security of the transactions over the Internet. In this way, when companies purchase key products these requirements are especially important for choosing the technical solution more adequate for their needs.

In this paper we propose a classification of alternative e-procurement topologies aimed at helping companies to establish the security implications of using different e-procurement commercial solutions.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 114-122, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Justification

Due to the importance of e-procurement solutions in order to improve supply management, managers need tools that help them to choose the best solution for their organization. In that sense, one of the important considerations that managers need to take into account is related to the security requirements that assure the information exchange among supplier and buyer organizations. Aspects related to reliability, confidentiality, and integrity of procurement transaction when companies purchase key products are especially important for choosing the technical solution more adequate for their needs.

In that sense, we propose a classification of alternative e-procurement topologies elaborated from the security requirements perspective, whose objective is to assist managers who have to choose a suitable e-procurement solution.

3 Alternative E-procurement Topologies

There are two main participants in an e-procurement solution, the supplier and the buyer. There might be another one, the third party intermediary. An intermediary in this context is a company that provides a service that promotes trade between buyers and suppliers in some way, including [1]: matching buyers and sellers; ensuring trust among participants by maintaining neutral position; facilitating market operations by supporting certain transaction phases; and aggregating buyer demand and seller information.

In this communication we proposed a classification of alternative e-procurement topologies. The following sections describe and characterize alternative e-procurement topologies from a security point of view.

3.1 Supplier Side

Firs of all, we are going to analyze the supplier side. This participant could access the e-procurement solution (directly to the buyer if it is hosted there, or through a third party intermediary when it is used by the solution) through the Internet or through a Virtual Private Network (VPN). The cost associated to the first possibility is less than the second one, but VPNs are more secure than the public Internet. For instance, the e-procurement solution provides by *Covisint* (the vertical e-marketplace for automotive industry) allows supplier access through the Internet or through a VPN such as ANX (Automotive Network eXchange) [2]. The ANX infrastructure was launched by the automotive industry in 1998. It is an extranet infrastructure supported by a VPN that enables the automotive manufacturers (Chrysler Corp., General Motor Corp., and Ford Motor Corp.) to link with their suppliers in conjunction with their Internet Service Providers [3].

3.2 Buyer Side

Secondly, we are going to analyze the buyer side. There are two possibilities. The e-procurement solution could be hosted in the buyer information system, so the buyer could access the solution through his Intranet, and we assume this infrastructure has appropriate mechanisms to offer an adequate level of security. Thus, we consider this alternative is appropriate when the buyer has strong security requirements. An example of this type of solution is a "buy-side solution" that the e-procurement module is hosted at the buyer's (see figure 1). These are solutions oriented to support purchasing processes between an important buyer and their suppliers. These are private solutions, which restrict the access to their suppliers only, in this way the company will do any purchase operation through the system (electronic catalog, auctions, request for quotation "RFQ-). Among those operations it is possible to include direct or indirect procurement. In these cases, the buy-side solution is hosted into the buyer information system, and hence the buying organization is in control of catalog content, data format, and back-end system functionality. Large companies commonly adopt this option because they are able to manage this e-commerce solution internally due to a powerful information system. The use of this choice implies a more direct integration with suppliers allowing the integration of the procurement process into the ERP (Enterprise Resource Planning) system. One clear example of this type of solutions is the ValueChain[1] website employed by Dell Computers, where suppliers connects to Dell information system through the Internet. The figure below shows an example of this kind of solution.

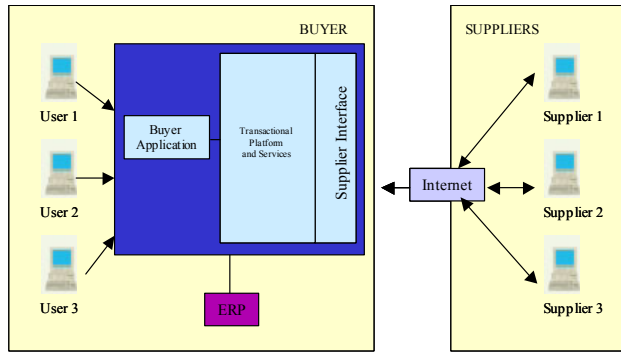


Fig. 1. Buy-side solution hosted into the buyer Information System (without intermediary)

The second possibility includes e-procurement solutions that work through a third party intermediary. In this option, the intermediary could host total or partially the solution. An example of e-procurement solution that is totally hosted in the third party intermediary is the electronic marketplace. This solution usually provides support for buying processes for multiple buyers and multiple suppliers. In any case, the solution is operated by a third party intermediary that typically controls catalog content, aggregate suppliers input, and provides additional functionality and standardized data

[1] Dell's suppliers website: valuechain.dell.com

access to buyers [4]. Some systems such *AribaNet*, *Marketsite* and *Opciona* are used for horizontal markets (supporting a specific functions such as MRO purchasing). Others, such as *Covisint*, *Exostar*, *Metalsite*, etc. support vertical markets (oriented to specific industries).

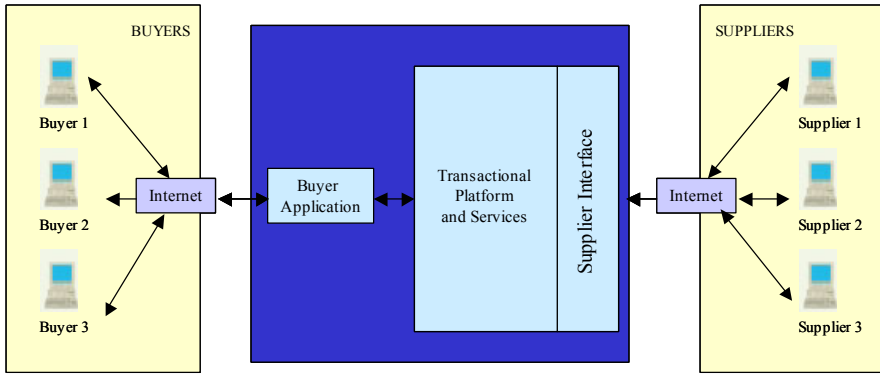


Fig. 2. Intermediary e-marketplace solution

Buy side solutions could also include an intermediary; in these cases, the e-procurement system is hosted, totally or partially, in this third party intermediary.

3.2.1 Buyer Access to a Third Party Intermediary E-procurement Solution

Once we have chosen an e-procurement solution involving an intermediary, we need to analyze how the buyer could gain access to this intermediary. In that case, it is possible to use the Internet or to use a VPN infrastructure (as previously discussed for the suppliers). In the buyer case, sometimes security requirements could be more important than in the case of supplier, since the information flow could be more sensitive, for instance when it links directly to the buyer ERP system. In that case, an infrastructure like ANX could be more appropriate than the access through the public Internet.

3.2.2 Type of Application Used in an Intermediary E-procurement Solution

When a third party intermediary e-procurement solution is chosen, it is important from a security implications perspective to consider which type of application we are using at both sides. In other words, if it is generic or specific, and in the later case, if it is from the same or from different software providers.

In this case, we have different possibilities. First of all, the buyer could use a generic application (i.e. standard browser) to access the application hosted in the intermediary and developed by a specific software provider (i.e. *CommerceOne*, *Oracle*, etc. and represented as product A in figure 3). In that case, we consider that this option opens a set of security considerations that need to be addressed, in order to achieve an adequate level of security.

The second possibility involves the case in which the buyer has its own application (because they bought it from a different provider than the provider of the application hosted in the intermediary or because they developed it internally) and the provider of

the application hosted in the intermediary is different (represented in figure 3 as product B vs. product A). In this case, it is necessary to take into account security considerations to solve specific security problems related to the information exchange between two different applications.

Third, it is possible that the same provider had developed the buyer application hosted in the buyer side, and the application hosted in the intermediary side. In that case, we consider that they could develop specific mechanisms to protect the information flows to be exchanged between these two participants.

Lastly, it is possible to find an e-procurement solution that combines two or more of the alternatives described before. For instance, the possibility of buyer users to access through a standard application and the possibility of exchange some information among the ERP buyer system and the procurement application hosted in the intermediary, that might be developed by different software providers (for more details, see section 5). In that case, it is recommended to take into account several security considerations that have been commented previously.

As a result of this study, we propose a classification that allows to characterize different e-procurement solutions from the point of view of the security implications of the topology. The following figure shows this classification of alternative e-procurement topologies.

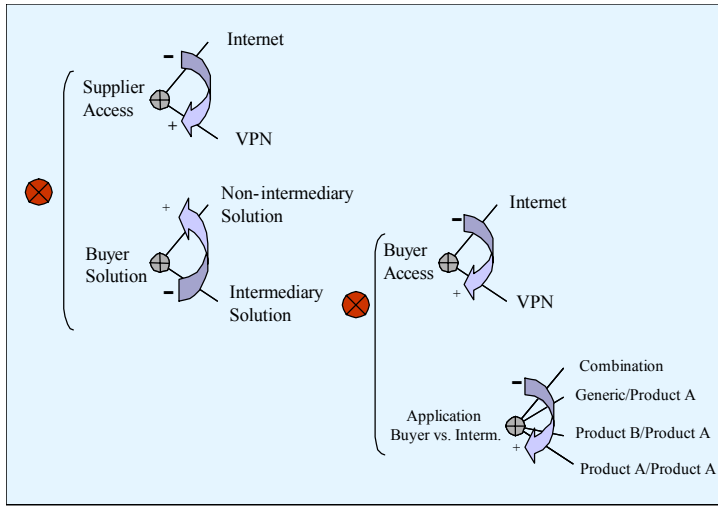


Fig. 3. Classification of alternative e-procurement topologies proposed

In figure 3, it is represented a summary of the proposed e-procurement solution classification. We use the plus sign and arrows to show different possibilities for each case, and we use the multiply sign to indicate that both sides must be considerate. There is a curved arrow over the different alternatives, in which the minus sign at the extreme is indicating in which alternative is needed more security considerations. The alternatives are classified according to their demanding security requirements, which are higher in the minus extreme of the arrow. The classification proposed represented 18 combinations, as we can extract from the figure 3 (2*[1+(2*4)]).

4 Intranet/Extranet-VPN/Internet and Security Mechanisms to Use

This study characterizes different e-procurement topologies, and as we can see in the classification proposed (see figure 3), depending on where is hosted the procurement solution, an inter-organizational infrastructure must be required for the information exchanged.

For instance, when the procurement solution is completely hosted at the buyer side, users access the buyer application through an Intranet infrastructure. In that case, we assume that this infrastructure has appropriate mechanisms to offer an adequate security level.

But, in the case that the buyer application is hosted in an intermediary, it needs to access to an external infrastructure, some organizations choose a Virtual Private Network (VPN). A VPN is an extension of a private network that encompasses links across shared or public networks such as the Internet. VPN connections leverage the IP connectivity of the Internet and use a combination of tunneling and data encryption to securely connect remote clients and remote servers. The underlying technologies that make it work such as the Point-to-Point Tunneling Protocol (PPTP), the Layer Two Tunneling Protocol (L2TP), and Internet Protocol Security (IPSec). In this case the security mechanism are implemented in a lower communication layer than application layer, supplying confidentiality, integrity and authentication for every Internet Protocol packet. This type of networks establishes a cryptographic tunneling which permits to get the security level of a private network but running over a public one. Although typical VPNs utilize the public Internet to connect users, it also could run over an Internet separate network, as ANX infrastructure [5]. The ANX Network is private (although it is a shared network), meaning that it is isolated from the public Internet. This network is intentionally separate and inaccessible from the public Internet to ensure the security, speed, and performance required to conduct mission-critical transactions[2]. Lastly, organization could use its own VPN (not share with other networks). In this case, the access is more restricted and as same as the ANX network, it is no expose to public Internet vulnerabilities (i.e. denial of service attacks) and also could offer greater reliability and faster data transmission.

When the infrastructure chosen is the Internet, the possibilities go from no protection to using some security mechanism for coding the protocol, as the HTTPS protocol (HiperText Transfer Protocol Security). Its major advantage is that all browsers include this protocol and hence no additional applications are needed. This protocol uses internally another protocol called Secure Socket Layer (a.k.a. SSL), which supplies confidentiality, integrity and authentication for the communications between the machine that runs the browser and the machine that runs the web server.

[2] More details in URL: http://www.anx.com/faqs/

5 Application of the Proposed Topologies to E-procurement Solutions Offered by *Opciona*

The proposed classification of e-procurement topologies could be useful for managers that have to choose an e-procurement solution to support the purchasing process for their organization to tackle the decision taking into account security considerations.

Opciona is a Spanish e-procurement solutions provider. They provide solutions from horizontal intermediary marketplaces (see fig.2) to different possibilities of □buy side□solutions (one of those are represented in fig. 1). In order to show the utility of the classification proposed aimed at helping companies to establish the security implications of e-procurement solutions, we have studied the following solution represented in figure 4.

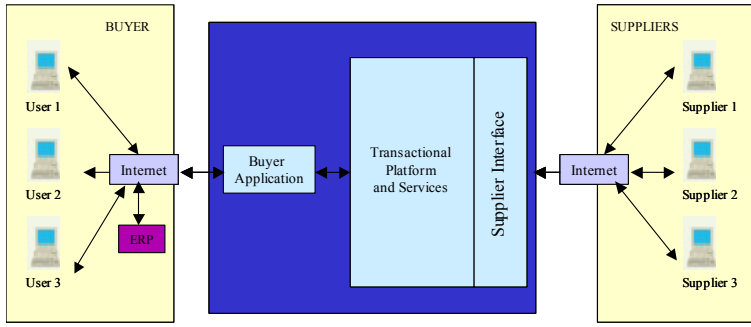


Fig. 4. Intermediary solution: ERP accessing to buyer application hosted in the intermediary

We are interested in analyzing security implications related to the use of this solution in order to support the purchasing process of the organization. The classification proposed in this communication aimed at helping in this purpose as it is shown in the following figure.

The solution provided by *Opciona* and represented in figure 4, presents an Internet access to the supplier and the buyer. In both cases, and according to our classification, a more secure option could be chosen (as the curved arrow represented in figure 5).

In the buyer side, there is an intermediary solution, and from the point of view of security, this option requires more security considerations than in the other one. The intermediary alternative presented in this particular solution is a combination of two. The first one is from generic application (browser) to the application provides by *Opciona* (product A in the chart) and the second one is the access from a different application (ERP of the company, product B in the chart) to the application offered by *Opciona* (product A in the chart). As we can observed in the classification proposed, this solution is exposed to security vulnerabilities, and several security considerations must be taken into account if we need high level of security for the transactions.

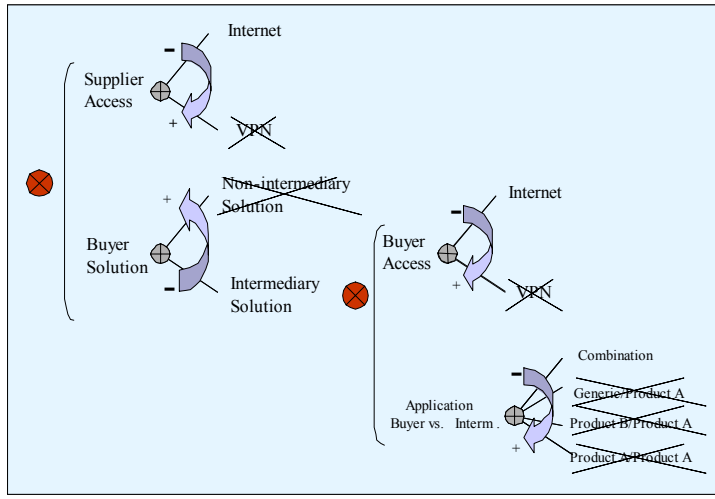


Fig. 5. Example of application of the proposed classification

6 Conclusions

The classification of alternative e-procurement topologies proposed in this communication allows the characterization of e-procurement solutions in terms of their topology security implications. In order to establish these alternative topologies we have considered how the two main participants (buyer and supplier) could access to the e-procurement solution (i.e. through the Internet or through a VPN). From the point of view of the buyer, we have considered two main possibilities, to use a third party intermediary that hosts the solution or to host it in the buyer information system. In the first case, we describe different situations when a third party intermediary is used (depending on the type of application at both sides: generic, commercial product, etc.), and the security implications have been characterized in each case.

The application of the classification of topologies proposed in order to analyze the security implications of an e-procurement solutions offered by *Opciona* have illustrated one of the possible utilities of the classification.

We can conclude that this classification could be useful for managers to tackle the decision of choosing an appropriate e-procurement solution according to their security requirements. In this sense, our classification is aimed at helping them in choosing the most adequate topology.

Nevertheless, our classification could be useful in the designing of the security mechanism because this classification characterize the topologies from the point of view of its security requirements.

7 References

1. Bailey, J., Bakos, Y.: An exploratory study of emerging role of electronic intermediaries. *International Journal of Electronic Commerce*. Vol. 1 N□3 (1997) 7-20.
2. Garretson, D., Mines, C.: The Net Revs Up Auto Making. Forrester. June (2000). Retrieved from URL: www.forrester.com/ER/Research/Report/Summary/0,13338,9484,FF.html
3. Merkov, M.: Extranet examples: extraordinary extranets. August 27 (1999). Webreference Web site: http://webreference.com/content/extranet/examples.html.
4. Archer, N., Gebauer, J.: Managing in the context of the new electronic marketplace. 1st *World Congress on the Management of Electronic Commerce*. Hamilton, Ontario Canada. January 19-21 (2000).
5. Brennan, M.: ANX Launches Auto Industry Portal Site. *Michigan Technology News* 09 August (2000). Retrieved from URL: www.mitechnews.com/news/show_article.asp?category=34&id=24.

Protection of Multiagent Systems

J.M. Sierra, J.C. Hernández, E. Ponce, and A. Ribagorda

IT security Group. Univeristy Carlos III of Madrid. Spain
sierra@inf.uc3m.es

Abstract. The use of intelligent agents to improve business to business (B2B) communications is starting to be adopted by more and more companies. The main reason is that intelligent agents are able to multiply the number of agreements between suppliers and customers. However, due to the type of information managed by agents those negotiations must be protected in order to avoid fraud and misrepresentation. Communications between agents must be authenticated and, once an agreement is signed, none of the parties should be able to repudiate its signature, and hence their agreement. Furthermore, some confidentiality services must be supplied, allowing to hide some message contents or identities.Security has important implications to the participants of the market, because they will trust the system according to the level of security supplied. This paper shows how RETSINA agents may obtain those security services using trust management and security mechanisms. Our approach uses the trust infrastructure defined by [3] for RETSINA and the security services supplied by the new IPSEC protocols [1].

1 Background

The RETSINA infrastructure is designed for the asynchronous collaboration among intelligent agents in support of performing a variety of decision-making tasks. We have chosen RETSINA because it was designed specifically for the Internet and for being a general and flexible multi-agent infrastructure. In RETSINA framework each user is associated with a set of agents that collaborate to support him in various tasks and to act on user's behalf. These are the components of the RETSINA model:

- ***Interface agents*** work together with users, receiving specifications from the users and transmitting them to the task agents.
- A ***Task agent*** is able of performing specific task defined by the interface agents. These agents collect the information needed from the information agents.
- ***Information agents*** are associated with the sources of information and supply it to other agents of the system.
- ***Agent Name Servers (ANS)*** are databases where agent names and their addresses are stored. When an agent A gets into the system he sends his ID and address to the ANS and when another agent B needs to contact agent A, he requests that information from the ANS.
- ***Matchmakers (MM)*** maintain a mapping of agent names and capabilities. When an agent gets into the system, he sends his capabilities to the matchmaker, which store them. When an agent needs services from the system, he sends an information

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 123-128, 2003.
© Springer-Verlag Berlin Heidelberg 2003

request to the matchmaker, which looks into his database and sends the answer to the agent.

Over this infrastructure we can define an scenario were consumers and suppliers can do their business through the Internet. The first stage take place when a supplier wants to put up for sale the '*good X*'. In this case, the supplier agent sends a registration message to known ANSs. The agent also sends a capabilities registration message to the known matchmakers. After this, a customer agent gets into the system looking for the '*good X*'. This agent sends a request of information to the Matchmaker asking for the good and the matchmaker responds with the supplier agents that are selling it. Once the customer agent has the name and capabilities of the supplier agent, he requests to the ANS the IP address of this agent. In the last stage, the customer directly contacts the supplier to establish an agreement for '*good X*'.

Our approach also uses the trust structures proposed by Wong and Sycara in [3], they include two new elements of the RETSINA scenario: Deployer Key Certificates (DCA) and Agent Certification Authority (ACA). These elements make possible the creation and management of agent certificates.

IPSEC protocol [1] provides a suite of security mechanisms that supply confidentiality, integrity and authentication to IP datagrams. IPSEC define two new IP level protocols: the Authentication Header (AH), which supplies integrity and authentication check, and the Encapsulated Security Payload (ESP) which supplies confidentiality, integrity and authentication to IP packets. Parameters needed to accomplish both protocols (called Security Associations) are securely established by the use of IKE protocol (Internet Key Exchange). IKE is a two phases protocol, which could develop authentication mechanisms based on the agent certificates described in [3].

2 RETSINA Security Vulnerabilities

In the next subsections we will describe threats that could be exploited by an agent to take advantage of the RETSINA system.

2.1 Message Eavesdropping

Messages exchanged through a public network, such as the Internet, can be easily eavesdropped. In fact, under certain situations, messages can be also modified and, in this case, their content can be forged. Furthermore, if a message is not authenticated, there is no way to ensure who is the sender of it, and hence, an agent could repudiate any not authenticated message because there is no proof of its origin.

Most of these vulnerabilities are inherited from TCP/IP architecture and, unless some security mechanisms are used, we must assume that contents of not ciphered messages are public, their integrity of a message can be forged and that their sender impersonated. According to the type of messages exchanged into the RETSINA scenario, some of them must be protected to achieve their confidentiality, integrity, authentication and non-reputation.

2.2 ANS Misrepresentation

When an agent comes up to the system, misrepresentation must be escaped. A 'supplier agent F' could send a registration message to the ANS saying that he is the 'supplier agent A', in this case all the customers that want to contact with the 'agent A' will receive the physical address of 'agent F'.

2.3 Matchmaker Misrepresentation

Let's suppose that a 'supplier agent A' already exist in the system (included into the ANS and the matchmaker databases) and another 'supplier agent F' sends a message to the matchmaker saying that the capabilities of 'agent A' must be changed (for example increasing the price of the goods offered by the 'agent A'). In this case, agent F could use misrepresentation to take advantage of the system

2.4 Matchmaker Speculation

Suppose a supplier who wants to create a forged market and uses a bunch of false supplier agents. All this agents can offer several products at a very low/high price, poisoning the bids. Once all these false agents have been included into the databases, if a customer asks for one of the products offered by false agents, he will receive all the bids for that product and this customer will select a supplier from a forged market.

2.5 ANS and Matchmaker Impersonation

All the threats described above can also be exploited if the ANS or the Matchmaker server is locally corrupted. In this case, honest agents (customer or supplier) can be cheated and the market can be easily forged.

2.6 Message Replay

Messages emitted by ANS or matchmaker can be easily replayed. Let's imagine an evil agent who has stored a set of old matchmaker answers, and suppose that a customer agent requests information from the matchmaker (information that was requested before). In this case, evil agent could send the answer more appropriate to his self-interest.

2.7 Agent Authorization

Agents will negotiate on behalf of their users, but sometimes an agent could be supplanted during the negotiation, provoking a forged and non valid agreement. For this reason agents must include a secure mechanism that permits to protect his impersonation, and hence the repudiation of their agreements.

2.8 Bids and Agreements Accountability

In this type of environment suppliers and customers must trust on unknown agents. However, agents may be dishonest and it is necessary to establish a technological infrastructure that allows the accountability of agent agreements.

3 IPSEC Protection of Retsina

As we said into the Background section, the IKE protocol must be accomplished before AH and ESP can be used. For the next sections, we consider that the parameters have been previously negotiated by IKE and we only describe the utilization of AH and ESP to protect RETSINA.

3.1 Registration Messages

Following subsections describe messages exchanged for the registration at the ANS (registration/unregistration at the matchmaker is not shown because is analogous to the ANS registration).

3.1.1 Registration into the ANS

Every agent who wants to enrol into the system must accomplish two registration steps, one into the ANS and other into the Matchmaker. The registration into the ANS needs agent ID and IP address, but also the ANS and the Matchmaker must be sure that the agent is who say he is. For that reason the registration messages must be authenticated and their integrity checked (confidentiality is not required, since the information maintained into the ANS is public). The answer of the ANS must also be protected because it could be sabotaged. For this reason the registered message must be also protected by authentication and integrity check.

Before the development of this exchange of messages, the agent and the ANS must agree the AH Security Association. Based on this security association the parties are able of doing the authentication and integrity check. The reply attacks are avoided by the use of the AH. The exchange of messages is :

```
1) Agent→ANS: AH {request(agent_ID, IPaddr)}

2)  ANS→Agent: AH {registered(agent_ID, IPaddr,
expiration)}
```

3.1.2 Unregistration into the ANS

Unregistration messages can be also used to block an agent. For this reason the exchange of this type of messages must be also protected. The protection is made in the same way than registration, using the AH. These are the messages exchanged:

```
1)  Agent → ANS :  AH {unregister(agent_ID)}

2)  ANS   → Agent :  AH {unregistered(agent_ID)}
```

3.2 Look Up Messages

There are two types of look up messages: address search and capabilities search. Both are described bellow.

3.2.1 Address Search into the ANS

When an agent A wants to find out the physical address of another one (in this case Agent_X) he must send a look up message to the ANS. Into this message he must include its certificate (Agent_Cert), a fresh request number (Req_Nonce) and the IDs of the ANS and the agent in search. This message is public and it is not necessary to protect it (every agent can access to the physical address of another). However the answer of the ANS must be protected, and the agent who request the information can be sure that the ANS is answering to his request and that the answer is fresh. These are the two messages exchanged:

```
1)  Agent_A→ANS:  search(ANS_ID, Agent_X_ID,
Agent_Cert, Req_Nonce)

2) ANS → Agent A: AH { answer(Agent_Cert, Req_Nonce ,
Agent_X_ID, Agent_X_address) }
```

3.2.2 MatchMaker Capabilities Search

The message exchanged for a capabilities search is very similar to the exchange described before. The first message is public and the second one must be protected because the answer must be authenticated and integrity checked by the solicitor. These are the messages:

```
1) Agent →MM: capabilities(ANS_ID, Agent_X_ID,
Agent_Cert, Req_Nonce)

2) MM→ Agent: AH { answer(Agent_Cert, Req_Nonce,
Agent_X_ID, capabilities_list) }
```

3.3 Negotiation Messages

For this type of messages is where our approach is more powerful because once a customer has found its supplier, they can establish a simple VPN through the Internet using IPSEC. In this way all their negotiations will be protected by data confidentiality, data integrity, data source authentication and anti-reply protection.

In the scenario we have described, once agents find one each other, they will develop the IKE protocol for establishing one or more security associations, which will facilitate the establishment of end-to-end cryptographic tunnelling to communicate securely.

4 Conclusions

In this paper we have described a security architecture to protect the RETSINA multiagent system. RETSINA is designed to be used in an open environment such as the Internet where there is no security service provided and where the amount of entities involved can be high. Both reasons induce the use of security mechanism widely accepted and standardized. We choose IPSEC because there is no doubt about its importance in the Internet future and because its protocols have been evaluated extensively by the Internet community.

IPSEC protocols, AH and ESP, are able of supplying authentication based on public key cryptography and for that reason we needed an infrastructure for the creation and management of agent certificates. Wong and Sycara work [3] added security and trust to multiagent systems and we have used the certificates management proposed by them to base our architecture. In contrast to their contribution our solution is not based on socket based security –such as SSL-, which is a significant approach because IPSEC virtual private networks are everyday more used by companies to develop their business.

Our work describes the security vulnerabilities of the RETSINA system and identifies what are the messages that must be protected. These vulnerabilities can be easily positioned on other similar multiagent systems and, in the same way, protection proposed in our paper can be used to protect their communications.

In our proposal every agent needs cryptographic capabilities, which includes a certificate agent and algorithms implementation. In a future work we will propose a new element that simplifies our architecture, the SESAMO agent (Secure Sub-Agent Module). SESAMO will operate like a proxy assuming the cryptographic capabilities of those agents that use it.

5 References

1. Doraswamy N. and Harkins D. IPSec, The new Security Standard for the Internet, Intranets, and Virtual Private Networks. Prentice Hall. 1999

2. Sycara K., Pannu A., Williamson M., Zeng D. and Decker K. Distributed Intelligent Agents. In IEEE expert, pages 36-45, 1996

3. Wong H. and Sycara K. Adding Security and Trust to Multi-Agents Systems. Carnegie Mellon University. 1999

On the Number of Equivalence Classes in Certain Stream Ciphers

L. Javier García-Villalba

Department of Computer Systems and Programming (DSIP)
Complutense University of Madrid (UCM)
Ciudad Universitaria s/n
28040 Madrid, Spain
javiergv@sip.ucm.es

Abstract. This paper discusses the relationship between the so-called Sequential Decomposition in Cosets representation and the non-linear filter of m-sequence representation. The aim is to get some knowledge about the different nonlinear filter equivalence classes, where these filters are partitioned according to cyclic shift and conjugacy. This equivalence also indicates an identical linear complexity for all sequences in the same equivalence class. The motivation of this paper is the desire to design stream ciphers using nonlinear filters which ensure high linear complexity. The paper ends giving a suggestion to construct nonlinear filters from different equivalence classes. Emphasis is on the cosets associated with a nonlinear filter since they determine these classes.

1 Introduction

Stream ciphers form an important class of secret-key encryption schemes. They are widely used in applications since they present many advantages: they are usually faster than common block ciphers and they have less complex hardware circuitry. Moreover, their use is particularly well-suited when errors may occur during the transmission because they avoid error propagation. In a binary additive stream cipher the ciphertext is obtained by adding bitwise the plaintext to a pseudorandom sequence, called the running-key (or the keystream). The running-key is produced by a pseudorandom generator whose initialization is the secret key shared by the users. Most attacks on such ciphers therefore consist in recovering the initialization of the pseudorandom generator from the knowledge of a few ciphertext bits (or of some bits of the running-key in known-plaintext attacks) [1] [10] [11] [16] [17] [18] [19] [20] [21] [22] [26].

Linear feedback shift registers (LFSRs) are the basic components of most keystream generators since they are appropriate to hardware implementations, produce sequences with good statistical properties and can be easily analyzed.

Most common sequence generators in stream cipher systems are based on a combination of LFSRs and nonlinear functions. Depending on whether the keystream involves one or more than one LFSR, the sequence generators are

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 129–138, 2003.
© Springer-Verlag Berlin Heidelberg 2003

commonly classified into filter generators and combination generators. In both cases the linear complexity is a measure of the suitability of a keystream for its cryptographic application. In fact, the linear complexity of sequences obtained from a nonlinear combination of LFSR-sequences is mostly predictable. Such is the case of many well-known generators proposals [25] (e.g. clock-controlled generators, alternating step generators, cascade generators, etc.) whose linear complexity is either linear or exponential in the number of storage cells employed.

On the other hand, the linear complexity of the filter generators depends exclusively on the particular form of the filter and the LFSR minimal polynomial. Generally speaking, there is no systematic method to predict the resulting complexity. This is the reason why only a few authors have faced the problem of the determination of the linear complexity for filter generators. At any rate, several fundamental contributions must be quoted.

Groth [9] concentrated in the use of 2nd-order products and presented the linear complexity as a controllable parameter which increases with the order of the nonlinear function. Nevertheless, any reference to the degeneracies which may occur in the linear complexity of the resulting sequence is completely omitted.

Kumar and Scholtz [13] derived a general lower bound for the class of bent sequences, although the LFSR length is restricted to be a multiple of 4. Rueppel [24] obtained a quite large lower bound on the linear complexity of nonlinearly filtered m-sequences when a linear combination of products of equidistant phases is applied to the LFSR stages.

Massey and Serconek [15] has focussed on the use of the Discrete Fourier Transform Technique to analyze the linear complexity. The former applies the DFT technique to the case of 2nd-order products exclusively while the latter derives a new way of the Rueppel's root presence test which can be applied to the case or 'regular shifts'.

Finally, Fúster and Caballero [2] based on the concept of fixed-distance coset to obtain a general lower bound for any arbitrary nonlinear function with a unique term of maximum order.

The present work is concerned with the problem of the determination of the number of nonlinear filters with different linear complexity.

The paper is organized as follows. Section 2 introduces some definitions and basic concepts that are needed in the work. Section 3 defines the so-called 2^j-distant function nonlinear filters. Section 4 computes the number of classes of nonlinear filters of m-sequences. In section 5, the form of these classes for second order nonlinear filters is showed. Finally, conclusions in section 6 end the work.

2 Basic Concepts and Definitions

Some definitions and different basic concepts are introduced as follows:

$\{a_n\}$ is the binary output sequence of a maximal-length LFSR. In the sequel and without loss of generality we assume that $\{a_n\}$ is in its characteristic phase, then the generic element a_n can be written as

$$a_n = \alpha^n + \alpha^{2n} + \cdots + \alpha^{2^{(L-1)}n} \tag{1}$$

$\alpha \in GF(2^L)$ being a root of the LFSR characteristic polynomial.

F denotes a *kth*-order nonlinear filter applied to the L stages of the LFSR. That is, F includes at least a term product of k distinct phases of the sequence $\{a_n\}$, $a_{n+t1}a_{n+t2}...a_{n+t_k}$, where the symbols t_j $(j = 1, 2, \ldots, k)$ are integers satisfying $0 \leq t_1 < t_2 < \cdots < t_k < 2^L - 1$.

$\{z_n\}$ is the sequence obtained at the output of the nonlinear filter F.

Let Z_{2^L-1} denote the set of integers $[1, \ldots, 2^L-1]$. We consider the following equivalence relation R defined on its elements: q_1 R q_2 with $q_1, q_2 \in Z_{2^L-1}$ if there exists a j, $0 \leq j \leq L-1$, such that

$$2^j \cdot q_1 \;=\; q_2 \mod 2^L - 1 \tag{2}$$

The resultant equivalence classes into which Z_{2^L-1} is partitioned are called the *cyclotomic cosets* modulo $2^L - 1$. All the elements q_i of a *cyclotomic coset* have the same Hamming weight, that is, the same number of 1's in their binary representation.

E, the leader element of the cyclotomic *coset* E, is the smallest integer in such an equivalence class. If L is a prime number, then the cardinal of every cyclotomic *coset* E is L (except for *coset* 0 whose cardinal is always 1). If L is a composite number, then the cardinal of a cyclotomic *coset* E may be L or a proper divisor of L. For a maximal-length LFSR, we can give the following definitions:

Definition 1. *The characteristic polynomial of a cyclotomic coset E is a polynomial P_E in the indeterminate D defined by*

$$P_E(D) \;=\; (D + \alpha^E)(D + \alpha^{2E}) \cdots (D + \alpha^{2^{(r-1)}E}) \tag{3}$$

where the degree r ($r \leq L$) of $P_E(D)$ equals the cardinal of the cyclotomic coset E.

Definition 2. *The characteristic sequence of a cyclotomic coset E is a binary sequence, denoted by $\{S_n^E\}$, that obeys to the expression*

$$\{S_n^E\} = \{\alpha^{En} + \alpha^{2En} + \cdots + \alpha^{2^{(r-1)}En}\}, \quad n \geq 0 \tag{4}$$

Note that the sequence $\{S_n^E\}$ is in its characteristic phase and satisfies the linear recurrence relation given by $P_E(D)$. Moreover, $\{S_n^E\}$ is a decimation of $\{a_n\}$ formed from this sequence by taking one out of E terms.

If *coset* E is a proper coset [8], then $P_E(D)$ is a primitive polynomial of degree L and its *characteristic sequence* $\{S_n^E\}$ is a m-sequence. If *coset* E is an improper coset [8], then $P_E(D)$ is either a primitive polynomial of degree

$r < L$ or an irreducible polynomial of degree L, consequently the period of its *characteristic sequence* $T(\{S_n^E\})$ is a proper divisor of $2^L - 1$.

In brief, every cyclotomic *coset* E can be characterized by its *leader element* E or its *characteristic polynomial* $P_E(D)$ or its *characteristic sequence* $\{S_n^E\}$.

LC is the linear complexity of the sequence $\{z_n\}$, whose value is given by the sum of the roots of the minimal polynomial of $\{z_n\}$.

3 2^j-Distant Function Nonlinear Filters

Let F a *kth*-order function applied to the stages of a LFSR of length L

$$F = a_{n+t_1} \cdot a_{n+t_2} \cdot \ \cdots \ \cdot a_{n+t_k}$$

and let F' be the 2^j-distant functions associated with F

$$F' = a_{n+t_1'} \cdot \ a_{n+t_2'} \cdot \ \ \cdots \ \ \cdot \ a_{n+t_k'}$$

with

$$t_i' = 2^j \cdot t_i \qquad \forall i, \qquad j = 1, \ldots, L-1$$

According to the Sequential Decomposition in Cosets [3] [4] [5] [6] [7], the sequence $\{z_n\}$ generated by the filter F can be written as

$$\{z_n\} = \left\{ \prod_{i=1}^{k} \left[\sum_{l=0}^{L-1} \alpha^{2^l \cdot t_i} \ \cdot \ \alpha^{2^l \cdot n} \right] \right\} = \sum_{i=1}^{N} C_{E_i} \{S_n^{E_i}\} \tag{5}$$

where N is the number of cosets of weight $\leq k$, $C_{E_i} \in GF(2^L)$ are constant coefficients and $\{S_n^{E_i}\}$ is the characteristic sequence of *coset* E_i.

By analogy, the sequence $\{z_n'\}$ generated by the filter F' can be represented as

$$\{z_n'\} = \left\{ \prod_{i=1}^{k} \left[\sum_{l=0}^{L-1} \alpha^{2^l \cdot t_i'} \cdot \alpha^{2^l \cdot n} \right] \right\} = \sum_{i=1}^{N} C_{E_i}' \{S_n^{E_i}\} \tag{6}$$

where the coefficients C_{E_i} and C_{E_i}' in (5) and (6) are related through the expression

$$C_{E_i}' = (C_{E_i})^{2^j} \qquad \forall \ i \tag{7}$$

since in $GF(2^L)$ it is verified $\forall\, j$

$$\left(\alpha^{a \cdot 2^j} + \alpha^{b \cdot 2^j} + \ldots + \alpha^{p \cdot 2^j} \right) = \left(\alpha^a + \alpha^b + \ldots + \alpha^p \right)^{2^j} \tag{8}$$

as is well known, for a field of characteristic p and any integer r

$$(a+b)^{p^r} = a^{p^r} + b^{p^r} \tag{9}$$

Consequently, if some C_{E_i} is zero in (5), then the correspondent $C_{E_i}{}'$ in (6) will also be zero and viceversa. Therefore, both filters have the same number of nondegenerate cosets, i.e., the same linear complexity.

In addition, the period of both filters is the same since they contain the same characteristic sequences, except for a phase shift.

Finally, this result is valid for all primitive feedback polynomials since (7) is independent of $GF(2^L)$.

The 2^j-distant nonlinear filters are easily characterized in the Fourier Transform Dominion [15]. In fact, these filters produce sequences whose transformed sequences, when those sequences are in their characteristic phase, are exactly identical as they contain the same frecuencies (or cosets).

4 On the Number of Equivalence Classes of Nonlinear Filters of m-Sequences

It is known [24] that a filter F and their shifted versions, that is, the next family of filters

$$F' = a_{n+t_1'} \cdot a_{n+t_2'} \cdot \cdots \cdot a_{n+t_k'}$$

with

$$t_i' = t_i + \delta \qquad \forall\, i, \quad \delta \in N$$

have the same period and linear complexity.

Also it is known by previous section that a filter F and the family of 2^j-distant functions associated with this F have exactly the same period and linear complexity.

If we consider in the set of the nonlinear filters of order k with $k < L$ and L prime (which is the most common case), a class that contains the shifted filters of a certain filter f and its 2^j-distant filters, the number of possible classes, that is to say, the number of possible nonlinear filters with different cryptographic properties (of period and linear complexity) is

$$N = \frac{\binom{2^L - 2}{k-1}}{L \cdot k} \tag{10}$$

Proof. First of all, we are going to see that the numerator of (10), denoted by Num, is always integer since it is a necessary condition. In fact, Num is always a positive integer by definition. Let see us that $Num \mid L$ and that $Num \mid k$. According to the well-known Fermat Theorem [23]

If p is prime and $\nmid a$ then $a^{p-1} \equiv 1 \pmod{p} \Rightarrow a^p \equiv a \pmod{p} \Rightarrow a^p - a \equiv 0 \pmod{p} \Rightarrow a^p - a = M \cdot p$

As Num can be written in this way

$$\binom{2^L-2}{k-1} = \frac{(2^L-2)^{(k-1)}}{(k-1)!} = \frac{(2^L-2)\cdot(2^L-3)\cdot\dots\cdot(2^L-k)}{1\cdot 2\cdot\dots\cdot(k-1)} \tag{11}$$

and remarking that the first factor in (11), (2^L-2), verifies Fermat Theorem, we can rewritten Num as follows

$$\binom{2^L-2}{k-1} = \frac{M\cdot L\cdot(2^L-3)\cdot\dots\cdot(2^L-k)}{1\cdot 2\cdot\dots\cdot(k-1)} \tag{12}$$

Therefore, $Num \mid L$.

Now, let us see that $Num \mid k$.

Let us suppose that $Num \nmid k$. This hypothesis would implicate that k divides neither of the $(k-1)$ consecutive factors of the numerator of (11). As the multiples of k are separated by a distance of k, the previous number to those factors, 2^L-1, must be a multiple of k, that is to say

$$2^L - 1 \equiv 0 \pmod{k} \tag{13}$$

But the number 2^L-1 is the *Lth Mersenne number* and it is well known [23] that any divisor of the Mersenne number $M_L = 2^L-1$ is of the form $2mL+1$ where m is a positive integer. Therefore, k $(k \leq L)$ will not be a divisor of M_L. Then, the initial hypothesis will be wrong, that is to say, $Num \mid k$. Consequently, $Num \mid L\cdot k$.

The cryptographic reasoning of this fact follows in a very natural way

- If L es prime then all the cosets have cardinal L.
- In a LFSR of length L, we have 2^L-1 possible stages to choose.
- If a kth-order filter $f = a_{n+t_1}\cdot a_{n+t_2}\cdot \dots \cdot a_{n+t_k}$ when we fixe $t_1 = 0$, we have 2^L-2 possible stages to participate in the $(k-1)$ remaining phases of the filter. Therefore, the number of possible filters (excluding the "shifted" filters) will be
$$\binom{2^L-2}{k-1}$$
- The 2^j-distant filters and the shifted filters are included in the above expression. To eliminate this cryptographic redundancy is necessary to divide by $L\cdot k$.

5 On the Equivalence Classes of Nonlinear Filters of m-Sequences

In this section we are going to see how these classes are. For example, let k be equal to two, this is, we are studying the second order filters. Remarking that it is verified

$$\frac{\binom{2^L-2}{k-1}}{L\cdot k} = \frac{\binom{L}{1}+\binom{L}{2}+\ldots+\binom{L}{\lfloor L/k\rfloor}}{L} \tag{14}$$

It can be easily derived that the way of the classes of nonlinear filters, remarking that the second part of (14) is equal to the number of cosets until binary weight $\leq \lfloor L/k\rfloor$. Let see us several examples.

Example 1. The classes for $L=5,\ k=2$ are 3 since

$$N = \frac{\binom{5}{1}+\binom{5}{2}}{5} = 3$$

Moreover, they are the next

$$a_n \cdot a_{n+1}$$
$$a_n \cdot a_{n+3}$$
$$a_n \cdot a_{n+5}$$

since the cosets of weight $\leq \lfloor L/k\rfloor = \lfloor 5/2\rfloor = 2$ are 1 (weight 1) and 3, 5 (weight 2).

Example 2. The classes for $L=7,\ k=2$ are 9 since

$$N = \frac{\binom{7}{1}+\binom{7}{2}+\binom{7}{3}}{7} = 9$$

Moreover, they are the next

$$a_n \cdot a_{n+1}$$
$$a_n \cdot a_{n+3}$$
$$a_n \cdot a_{n+5}$$
$$a_n \cdot a_{n+9}$$
$$a_n \cdot a_{n+7}$$
$$a_n \cdot a_{n+11}$$
$$a_n \cdot a_{n+13}$$
$$a_n \cdot a_{n+19}$$
$$a_n \cdot a_{n+21}$$

since the cosets of weight $\leq \lfloor L/k\rfloor = \lfloor 7/2\rfloor = 3$ are 1 (weight 1), 3, 5, 9 (weight 2) and 7, 11, 13, 19, 21 (weight 3).

Example 3. The classes for $L = 11,\ k = 2$ are 93 since

$$N = \frac{\binom{11}{1} + \binom{11}{2} + \binom{11}{3} + \binom{11}{4} + \binom{11}{5}}{11} = 93$$

since the cosets of weight $\leq \lfloor L/k \rfloor = \lfloor 11/2 \rfloor = 5$ are 1 (weight 1), 3, 5, 9, 17, 33 (weight 2), 7, 11, 13, 19, 21, 25, 35, 37, 41, 49, 67, 69, 73, 81, 137 (weight 3), 15, 23, 27, 29, 39, 43, 45, 51, 53, 57, 71, 75, 77, 83, 85, 89, 99, 101, 105, 113, 139, 141, 147, 149, 153, 163, 165, 169, 201, 293 (weight 4) and 31, 47, 55, 59, 61, 79, 87, 91, 93, 103, 107, 109, 115, 117, 121, 143, 151, 155, 157, 167, 171, 173, 179, 181, 185, 199, 203, 205, 211, 213, 217, 229, 233, 295, 299, 301, 307, 309, 331, 333, 339, 341 (weight 5).

6 Conclusions

This paper discusses the relationship between the so-called Sequential Decomposition in Cosets representation and the non-linear filter of m-sequence representation. The aim is to get some knowledge about the different nonlinear filter equivalence classes, where these filters are partitioned according to cyclic shift and conjugacy. This equivalence also indicates an identical Linear Complexity for all sequences in the same equivalence class. The motivation of this paper is the desire to design stream ciphers using nonlinear filters which ensure high linear complexity. The paper ends giving a suggestion to construct nonlinear filters from different equivalence classes. Emphasis is on the cosets associated with a nonlinear filter since they determine these classes.

7 Acknowledgments

This work is supported by the Spanish Ministry of Science and Technology (MCYT, Spain) under Projects TIC2002-04516-C03-03 and TIC2000-0735. The author would like to express his appreciation to the Programa Complutense del Amo for providing him a grant to stay at IBM Research Division. During this work he was with the Information Storage Group at the IBM Almaden Research Center, San Jose, California, USA (javiervi@almaden.ibm.com).

References

1. Chepyzhov, V., Smeets, B.: On a Fast Correlation Attack on Certain Stream Ciphers. Advances in Cryptology-EUROCRYPT'91. Lecture Notes in Computer Science Vol. 547, pages 176-185, Springer-Verlag, 1991.
2. Fúster-Sabater, A., Caballero-Gil, P.: On the Linear Complexity of Nonlinearly Filtered PN-Sequences. Advances in Cryptology-ASIACRYPT'94. Lecture Notes in Computer Science Vol. 917, Springer-Verlag, 1994.
3. Fúster-Sabater, A., García-Villalba, L. J. : Likelihood that a Pseudorandom Sequence Generator has Optimal Properties. Electronics Letters 34 (1998) 646-647.

4. Fúster-Sabater, A., García-Villalba, L. J. : An Efficient Algorithm to Generate Binary Sequences for Cryptographic Purposes. Theoretical Computer Science 259 (2001) 679-688.
5. García-Villalba, L. J., Fúster-Sabater, A.: On the General Classification of Nonlinear Filters of m-Sequences. Information Processing Letters 69 (1999) 227-232.
6. García-Villalba, L. J., Fúster-Sabater, A.: On the Linear Complexity of the Sequences Generated by Nonlinear Filterings. Information Processing Letters 76 (2000) 67-73.
7. García-Villalba, L. J., Rodríguez-Palánquex, M. C.: A Family of Keystream Generators with Large Linear Complexity. Applied Mathematics Letters 14 (2001) 545-547.
8. Golomb, S. W.: Shift Register-Sequences. Aegean Park Press, California, 1982.
9. Groth, E. J.: Generation of Binary Sequences with Controllable Complexity. IEEE Trans. Inform. Theory, Vol. IT-17, May 1971.
10. Johansson, T., Jönsson, F.: Improved Fast Correlation Attacks on Stream Ciphers via Convolutional Codes. Advances in Cryptology-EUROCRYPT'99. Lecture Notes in Computer Science Vol. 1592, pages 347-362, Springer-Verlag, 1999.
11. Johansson, T., Jönsson, F.: Fast Correlation Attacks Based on Turbo Codes Techniques. Advances in Cryptology-CRYPTO'99. Lecture Notes in Computer Science Vol. 1666, pages 181-197, Springer-Verlag, 1999.
12. Key, E. L.: An Analysis of the Structure and Complexity of Non-Linear Binary Sequence Generators. IEEE Trans. on Information Theory, Vol. IT-22, No. 6, pages 732-736, Nov. 1976.
13. Kumar, P. V., Scholtz, R. A.: Bounds on the Linear Span of Bent Sequences. IEEE Trans. on Information Theory, Vol. IT-29, pages 854-862, Nov. 1983.
14. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications. Cambridge University Press, 1986.
15. Massey, J. L., Serconek, S.: A Fourier Transform Approach to the Linear Complexity of Nonlinearly Filtered Sequences. Advances in Cryptology-CRYPTO'94. Lecture Notes in Computer Science Vol. 839, pages 332-340, Springer-Verlag, 1994.
16. Meier, W., Staffelbach, O.: Fast Correlation Attacks on Stream Ciphers. Advances in Cryptology-EUROCRYPT'88. Lecture Notes in Computer Science Vol. 330, pages 301-314, Springer-Verlag, 1988.
17. Meier, W., Staffelbach, O.: Fast Correlation Attacks on Stream Ciphers. J. Cryptology, pages 159-176, 1989.
18. Meier, W., Staffelbach, O.: Nonlinearity Criteria for Cryptographic Functions. Advances in Cryptology-EUROCRYPT'89. Lecture Notes in Computer Science Vol. 434, pages 549-562, Springer-Verlag, 1989.
19. Mihaljević, M. J., Golić, J. D.: A Fast Iterative Algorithm for a Shift Register Initial State Reconstruction Given the Noisy Output Sequence. Advances in Cryptology-AUSCRYPT'90. Lecture Notes in Computer Science Vol. 453, pages 165-175, Springer-Verlag, 1990.
20. Mihaljević, M. J., Golić, J. D.: A Comparison of Cryptanalytic Principles Based on Iterative Error-Correction. Advances in Cryptology-EUROCRYPT'91. Lecture Notes in Computer Science Vol. 547, pages 527-531, Springer-Verlag, 1991.
21. Penzhorn, W. T., Kühn, G. J.: Computation of Low-Weight Parity Checks for Correlation Attacks on Stream Ciphers. Cryptography and Coding - 5th IMA Conference. Lecture Notes in Computer Science Vol. 1025, pages 74-83, Springer-Verlag, 1995.

22. Penzhorn, W. T.: Correlation Attacks on Stream Ciphers: Computing Low-Weight Parity Checks Based on Error-Correcting Codes. Fast Software Encryption'96. Lecture Notes in Computer Science Vol. 1039, pages 159-172, Springer-Verlag, 1996.
23. Rosen, K. R.: Elementary Number Theory and its Applications. Addison-Wesley, 1993.
24. Rueppel, R. A.: Analysis and Design of Stream Ciphers. Springer-Verlag, New York, 1986.
25. Simmons, G. J.: Contemporary Cryptology: The Science of Information Integrity. IEEE Press, 1991.
26. Siegenthaler, T.: Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Trans. on Computers, Vol. C-34 (1), pages 81-84, 1985.

Parallelization and Vectorization of Simulation Based Option Pricing Methods

Jürgen Schumacher[1], Uwe Jaekel[2], and Achim Basermann[2]

[1] Dept. of Computer Science III, University of Bonn,
Römerstrasse 164, D-53117 Bonn, Germany,
schumac1@cs.uni-bonn.de
[2] NEC Europe Ltd., C&C Research Laboratories,
Rathausallee 10, D-53757 Sankt Augustin, Germany,
{basermann,jaekel}@ccrl-nece.de

Abstract. Simulation based pricing methods are used for a broad range of derivative valuation problems for which no closed form solution is known. They are easily adaptable to new products, and they show a superior performance for multidimensional pricing problems compared to other pricing techniques. In this paper we show how pricing methods based on Monte Carlo simulation and the stochastic mesh method of Broadie and Glasserman can be sped up by means of parallelization and vectorization. Computational results are given for multidimensional American and European pricing problems and on two different execution platforms; an MPI based NEC PC cluster and an NEC SX-6i vector computer.

1 Introduction

Both the ever-increasing complexity of financial derivatives and the need to adapt quickly to changing market conditions makes high demands on the flexibility and the performance of pricing and hedging tools. For instance, a client may need an offer for a specially tailored product from a bank. The bank is interested in making the offer as soon as possible before a competitor can do. Furthermore, hedging erros can be reduced by re-computation of price estimates since the situation for which the parameters have been computed and the situation in which the results can be applied have changed. In this paper we present how simulation based pricing methods can be parallelized and vectorized in order to deliver price estimates and their derivatives in time.

Since Phelim Boyle [1] showed how Monte Carlo Simulation can be used to value financial derivatives it has become a commonly used tool in the financial industry. Advantages are the flexibility with respect to the model of future evolution of the underlying assets and the different types of securities that can be priced with this model. With the developement of the stochastic mesh method [3,4] and the least squares approach [5] simulation based pricing methods have become generally applicable to American style options. Their main disadvantage is the small convergence speed. However, these methods are faster than finite

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 139–147, 2003.
© Springer-Verlag Berlin Heidelberg 2003

difference alternatives for options depending on many underlyings since their asymptotic convergence rate is independent of the number of the underlyings.

Our experimental results with realizations of these methods show that the parallelized and vectorized software achieves high performance on an MPI based NEC PC cluster and on an SX-6i vector processor.

2 Simulation Based Option Pricing

2.1 European Options

All simulation based pricing methods for European style financial derivatives are working with the representation of the option price $P_t(x_t)$ at time t for an underlying vector x_t of dimension d as an expected value of the payoff values $P_T(x_T)$ under the risk neutral propabilty measure for the underlying outcomes at maturity T given by the density function $f(x_T)$:

$$P_t(x_t) = \int e^{-r(T-t)} P_T(x_T) f(x_T) dx_T.$$

Given this representation, the price can be approximated by

$$P_t(x_t) \approx \frac{e^{-r(T-t)}}{N} \sum_{i=1}^{N} P_T(F^{-1}(y_i))$$

where $y_i \in [0,1)^d$ are either uniformly independently distributed vectors in case of a Monte Carlo Simulation or elements of a low-discrepancy sequence in case of a Quasi-Monte Carlo Simulation and F^{-1} is the inverse cummulative distribution function of the risk neutral meassure. The computation of this approximation is normally performed by repeating the following steps N times:

1. Generate the next uniform vector/next low-discrepancy element y_i.
2. Transform y_i into an underlying sample $x_i = F^{-1}(y_i)$.
3. Evaluate the payoff function $z_i = P_T(x_i)$ for the generated sample.
4. Add z_i to the sum of payoff values.

2.2 American Options

Pricing American style contingent claims is much more complicated since these products offer the option buyer the right to exercise the option before maturity. The fair price for these products is determined by the expected payoff under the risk neutral measure and the optimal exercise strategy

$$P_{t_0}(x_{t_0}) = \sup_{\tau} E[e^{-rt} P_\tau(x_t)]$$

where $E[]$ denotes the expected value of the term in brackets, x_t is the sample path of the underlyings, P_τ the payoff function with respect to the exercise

strategy τ. If an optimal exercise strategy is known it can be used to determine the exercise date for each simulated underlying path which gives the appropriate payoff value of the path. The optimal exercise time is characterized as the first exercise date where the direct payoff is greater or equal to the expected value.

The stochastic mesh method of Broadie and Glasserman [3] approximates the expected value of the non-exercised option for a set of grid points. In a first phase it generates n underlying paths according to the risk neutral model and estimates the payoffs at maturity for each path. In the second phase the algorithm steps backward from the last but one exercise date to the actual time. For each path value x_{ij} at the processed exercise date i, it approximates the discounted expected value of the not exercised option $\overline{C}_{ij}$ by

$$\overline{C}_{ij} \approx \frac{e^{-r(t_{i+1}-t_i)}}{N} \sum_{k=1}^{N} \hat{C}_{x_{i+1,k}} \frac{f_{i,x_{ij}}(x_{i+1,k})}{g_{x_0}(x_{i+1,k})} \tag{1}$$

with $\hat{C}_{x_{i+1,k}}$ the estimated American option price in state $x_{i+1,k}$, $f_{i,x_{ij}}(x_{i+1,k})$ the risk neutral transition density between x_{ij} and $x_{i+1,k}$, and $g_{x_0}(x_{i+1,k})$ the sampling density of node $x_{i+1,k}$. This estimation is based on Monte Carlo estimation with importance sampling. With the approximation for the continuation value the estimate $\hat{C}_{x_{i,j}}$ for the node x_{ij} becomes

$$\hat{C}_{x_{i,j}} = \max(\overline{C}_{ij}, P_{t_i}(x_{i,j})).$$

The estimates $\hat{C}_{x_{ij}}$ are all high biased estimates of the option price for their node. In the third phase the stochastic mesh method of Broadie and Glasserman generates a new sample set of risk neutrally distributed underlying paths with underlying values y_{ij}. The exercise time for these paths is the first exercise date where the estimate of the continuation value computed with the help of the $\hat{C}_{x_{i,j}}$ from phase 2 in (1) is smaller than the exercise payoff. The discounted payoff of the j-th path is given by

$$\hat{P}_j = e^{-r(t_k-t_0)} P_{t_k}(y_{ij}) \quad \text{with}$$
$$k = \min_i \left(P_{t_i}(y_{ij}) \geq \frac{e^{-r(t_{i+1}-t_i)}}{N} \sum_{k=1}^{N} \hat{C}_{x_{i+1,k}} \frac{f_{i,y_{ij}}(x_{i+1,k})}{g_{x_0}(x_{i+1,k})} \right).$$

The average P_j is a low biased estimate of the option price. The high biased estimate for the root node and the low biased path based estimate computed in phase 3 enclose the option price with high probability.

3 The Software Interface

The simulation based option pricing methods described in this paper are part of a pricing server. This server should provide existing banking clients with enhanced pricing methods especially for multidimensional options. In order to decrease the

communication between banking client and pricing server, we found it useful to divide the problem descriptions into three parts: an underlying model, a deal description, and a parameterization. All three description types can be inserted in the database of the server, modified, or deleted. A query for the option price or the price derivatives contains the reference to the three different parts of the problem description that are combined in the calculation. This functionality of the server permits the client to demand prices for different product alternatives without re-sending the underlying model description each time, or it allows the client the computation of the current price derivatives needed for hedging purposes by sending only the information about changes in the underlying model. In addition to the potential reduction in the message size, this design allows to decrease the time needed to perform initialization steps inside the server. The interface of the software is visualized in Fig. 1.

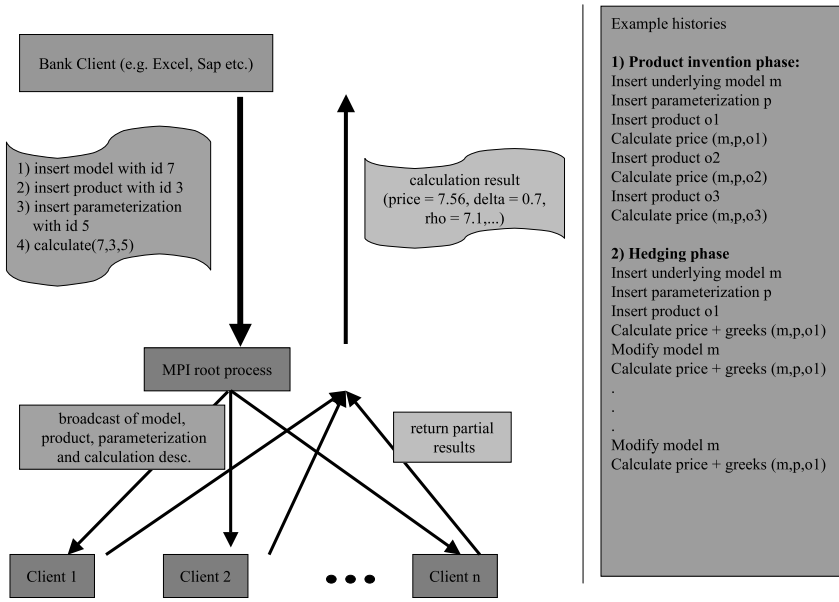


Fig. 1. Structure of the interface to bank clients.

4 Parallelization and Vectorization of the Pricing Methods

4.1 Vectorization of Pricing and Hedging Methods

Vector computers like NEC's SX-6i provide a set of different arithmetic and non-arithmetic units for vector operations. In addition, each unit exploits pipelining.

A very large register set and an extremely high bandwith between registers and main memory guarantees very fast data access. The result is that many floating point or integer operations can be executed per clock cycle. The SX-6i, for instance, achieves a peak performance of 8 Gflop/s with a clock rate of 500 MHz.

Algorithms on vector computers are efficient if they spend most of their time in loops of decent size according to the single instruction, multiple data principle. This situation usually is given in simulation codes since the basic operations mostly are or can be formulated as vector or matrix operations. Vector machines do not have a cache. Other than for cache processors, data reuse is not necessary in order to get a large fraction of peak performance. This simplifies efficient programming on vector machines distincly.

Compilers on vector machines translate unconditional innermost loops of operations in high level languages into vector instructions. The operations inside the loop should be basic operations (no function calls) which do not change the number of loop iterations.

In order to support the vectorization of the Monte Carlo simulation based price estimation and to retain the flexibility of the method we have chosen to work with vectors of samples for the different steps of the calculation. This means that we generate a vector of random numbers with the help of a random generator that supports vectorization [2]. The generated vector of independent N(0,1) distributed values is then transformed into a vector of independent underlying value paths distributed according to the risk neutral model. The resulting vector of underlying paths is transformed into a vector of payoff values according to the payoff function. In the final step the sum of generated payoff values is computed. The vector based design of the software leads to exchangeable methods for the generation of risk neutral underlying values and the evaluation of the payoff function with long innermost loops.

4.2 Parallelization for European Options

We have parallelized the Monte Carlo based estimation of option prices and price derivatives by partitioning the set of independent samples into p (p: number of processors) subsets of nearly equal size. The root process broadcasts the problem description to all client processes together with a process specific seed. After receiving the problem description, each client starts with the simulation of its part of samples. Afterwards the estimated prices and their standard deviation are sent back to the root process which computes the aggregate results.

4.3 Parallelization of the Stochastic Mesh Method

The parallelization of the stochastic mesh method is more complicated. Here we can distinguish three different phases: the generation of the underlying grid values, the estimation of high and low biased grid estimators, and the estimation of the low biased independent path estimates.

In the first phase the root process computes the set of all random underlying paths constituting the grid and evaluates the payoff function for all exercise dates. The underlying paths and the corresponding payoff values are then broadcasted to the client processes which start with the evaluation of the high and low-biased grid estimators. This step has not been parallelized so far since it is of complexity of $O(n)$ with n the number of paths simulated, which is much smaller than the complexity $O(n^2)$ of the other steps.

The second phase consists of a loop over the exercise dates of the option starting with the last exercise date. Each iteration starts with the root process broadcasting the price estimates for the current exercise date. For the first iteration this is done by sending the underlying paths and their corresponding payoff values. Each client process receives these estimates and uses them to compute new estimates for its subset of underlying path values at the previous exercise date. The computation of each estimate is costly since it requires to compute a weighted average over all estimates at the exercise date processed in the last iteration. The weighting factor used in this estimation is dependent on the underlying value for which the new estimate is computed and on the underlying value of the estimate processed at the last iteration. Fig. 2 visualizes a sample computation with the set of points divided into two parts. The connections between the points from both time layers show the data dependency of single point estimation steps. The subsets of estimates are sent back to the root process that again sends the complete set in the next iteration.

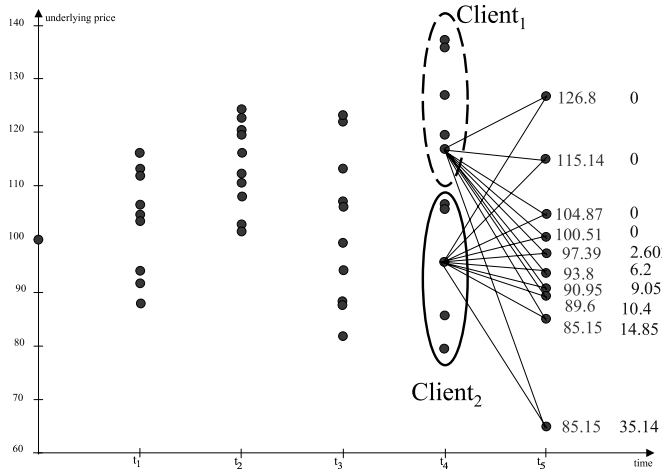


Fig. 2. Calculation of grid estimates.

In the third phase a new set of underlying paths is sampled by the root process, split into as much equally sized parts as client processes are available,

and distributed to the clients. Each client evaluates the path based low biased estimator with the help of the estimates computed in phase 2 for its set of paths. Subsequently, the clients send their average path estimates to the root process. After receiving all path based estimates, the root process computes the aggregate path estimate and sends the combined results to the client processes.

5 Experimental Results

In order to evaluate the speedups realized by the software we have performed pricing calculations for a European first to default swap on 6 correlated underlying bonds in a rating migration model, a European call option on the maximum of 5 underlyings stocks and an American call option on the maximum of 5 underlyings. All three sample problems have been priced in an identical configuration on an NEC PC cluster with 2-way AMD Athlon MP 1900+ SMP nodes and on an NEC's SX-6i vector processor. Table 1 shows the running times for the different problems and execution platforms.

Table 1. Experimental Results: Execution times in seconds.

Pricing problem	MPI 1 Client	MPI 2 Clients	MPI 4 Clients	MPI 8 Clients	SX-6i
European Max. Basket Call	20.90	10.50	5.25	2.626	0.531
First to default swap (Rating Migration Model based on Moodys Ratings)	130.27	65.57	33.754	17.311	15.977
American Max. Basket Call (Black Scholes Model)	260.71	163.24	83.53	44.875	8.159

The experimental results show that the parallelization speedup reached by using an increasing number of processors is near the theoretical optimum for the European pricing problems. For the 8 MPI clients case the speedup is 7.96 in case of the European max basket call and 7.53 for the First to Default Swap which is close to the upper bound of 8. For the American pricing problem, however, we are observing much smaller speedups. For this problem we have a speedup of 1.6 if we compare the 1 client withthe 2 clients execution times, but the ratio between the times of 4 and 8 clients is 1.86. These differences are probably caused by cache misses since the ratio between the times for 1 and 2 clients increases to 1.95 for less accurate computations on smaller grids. The results for the NEC SX-6i vector processor system show that vectorization delivers high performance for these problems. Compared to the single client PC results the speedup ratios are 39.36, 8.15, and 32.03 for the first, second and

third problem. The comparison shows that high performance can be achieved on the vector processor without the effort needed for parallelization via message passing. The relatively small speedup achieved for the rating migration model is due to the fact that these models require intensive integer operations for function table look-ups. PC processors are optimized for integer operations whereas vector processors are usually optimized for floating point performance.

The vector based design of the simulation software permits the parameterization of all simulation tasks for European options in order to achieve the highest speed. The vector size used in the algorithms can be optimized according to the architecture of the hardware platforms. On a vector processor, vector lengths as long as possible are preferable. On a PC cluster, vector operations allow efficient data reuse as well as hardware and software pipelining. There is an optimal vector size on cache processors: Too long vectors result in cache misses, too short vectors do not exploit data reuse and pipelines efficiently. The time behavior for different vector sizes on the PC cluster is shown in Fig. 3 for a call option in the Black Scholes Modell and for the first to default swap in the rating migration model. The execution diagram shows that the optimal vector size for the Black Scholes Model is around 9000 whereas the optimal vector size for the rating migration model is smaller than the minimal vector size of the random generator which is arround 2700.

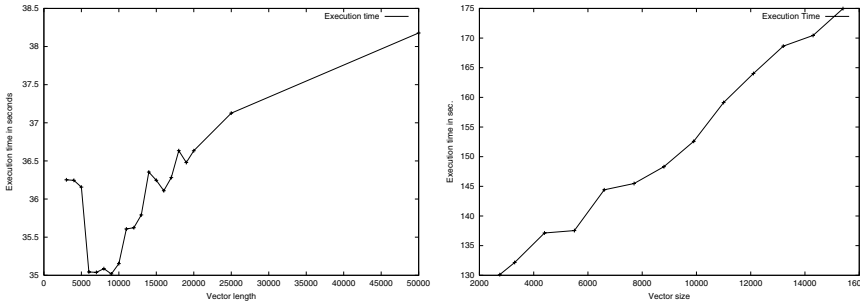


Fig. 3. Execution time versus vector size for a one dimensional Black Scholes and the rating migration model on the PC cluster.

6 Conclusions

We have presented the parallelization and vectorization of the standard Monte Carlo based option pricing methods for European options and of the Stochastic Mesh Method for American options. The experimental results for the MPI based NEC PC cluster and the NEC SX-6i show that both approaches can reduce the computation time for price estimates dramatically. The performance gain

on the vector machine is encouraging since the code adaption required only small changes to the software compared with the effort for parallelization on the basis of message passing. Our experimental results indicate that the vector based design is advantageous even for standard PCs. Future PCs will probably benefit distinctly by vectorization principles since more and more design elements of vector machines are included into standard processors and the support for pipelining within compilers will be enhanced.

References

1. Boyle, P.: Options: A Monte Carlo Approach. J. Financial Econ. **4** No. 3 (1977) 323-338
2. Brent, Richard P.: Fast Normal Random Number Generators for Vector Processors. Technical Report TR-CS-93-04 Australian National University
3. Broadie, M., Glasserman, P.: A Stochastic Mesh Method for Pricing High-Dimensional American Options. Working Paper Columbia University
4. Broadie, M., Glasserman, P.: Estimating Security Price Derivatives Using Simulation. Management Science **42** No. 2 (1996) 269-285
5. Longstafff, Francis A., Schwartz, Eduardo S.: Valuing American Options by Simulation: A Simple Least-Squares Approach Review of Financial Studies **14** No. 1 (2001) 113-147

An Algorithm for Determining the Controllers of Supervised Entities at the First and Second Levels: A Case Study with the Brazilian Central Bank

Vinícius Guilherme Fracari Branco[1], Li Weigang[2], Maria Pilar Estrela Abad[3], Jörg Denzinger[4]

[1] Deorf/Copec of the Brazilian Central Bank, Brasilia – DF, Brazil
Vinicius.fracari@bcb.gov.br
[2] Department of Computer Science, University of Brasilia, C.P. 4466, CEP: 70919-970, Brasilia – DF, Brazil
Weigang@unb.br
[3] Politec Informática Ltda., Brasilia – DF, Brazil
Pilar@bsb.politec.com.br
[4] Department of Computer Science, University of Calgary, Canada
Denzinge@cpsc.ucalgary.ca

Abstract. An algorithm was developed and implemented to find controllers from the stock shares for some financial institutions for the Brazilian Central Bank (BCB). The original problem is similar to a typical *Sum of Subset* problem that is suggested to be solved by a backtracking algorithm and the problem complexity is NP-complete. Usually BCB solves this problem manually which is time consuming and prone to errors. The heuristical approximation algorithm presented in this paper has polynomial complexity $O(n^3)$ and is based on subroutines for determining controllers at the first two levels. The paper describes the basic concepts and business rules currently employed in BCB, our algorithm and its major subroutines, it gives a brief complexity analysis and an example illustration at level 2. Our experimental results indicate the feasibility of an automation of the process of finding controllers. Though developed for BCB, our algorithm works equally well for other financial institutions.

1 Introduction

Although computer hardware development has increased the processing speed enormously over the last years, there are still a lot of real problems which cannot be solved optimally in reasonable time using modern computers, as computation theory has proven. To find controllers of an institution or company based on information about stock shares involving intermediate entities is one example of such a problem.

According to the rules of the Brazilian Central Bank (BCB), all financial institutions in Brazil, such as banks, investment and financial companies, credit societies, etc. should inform BCB about each ownership change. These institutions are called Supervised Entities (SE). After each stock movement, BCB needs to recalculate the potential controllers of an SE, which are defined as possible combinations of shareholders who have the privilege to vote directly and hold together 51% or above stock

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 148-157, 2003.
© Springer-Verlag Berlin Heidelberg 2003

shares of the SE. The importance of determining the potential controllers is significant. For example, if a SE bankrupted unreasonably, BCB needs to know its main shareholders in order to congeal immediately their properties for further juridical activities. In 2001 alone, there were 1100 financial institutions in whole Brazil and at least 500 SEs informed BCB about changes to their ownerships. This meant that BCB had to find out the controllers of all these institutions 500 times during 2001.

Usually the ownership of an SE is complicated, because of the transitive nature of the control property due to having stock shareholders. SE *A* may hold stocks of SE *B*, SE *B* may be the main controller of SE *C*. BCB divides the distribution of stock shares into levels. In this case, SE *C* is at level 1, SE *B* at level 2 and SE *A* at level 3. On the other hand, to determine the controllers is a difficult problem. In terms of mathematics, the problem is known as *Sum of Subset*, and the best known existing method to solve it optimally is by a backtracking algorithm with the complexity of $O(p(n)\, 2^n)$ or $O(p(n)\, n!)$, where $p(n)$ is a n^{th} order polynomial [2, 3, 4]. Here n corresponds to the number of individual shareholders. This means it is close to being impossible to determine the controllers from a huge number of stock shares for a big SE. To solve the problem of *Sum of Subset*, there have been some theoretical and empirical studies [5, 6, 7]. In the BCB case, the possible solution space is reduced quickly because of the great quantity of stock holders with a small amount of shares. This makes it possible to develop an approximation algorithm with polynomial complexity.

Even though all other management tasks of BCB are automatically processed, the determination of controllers is still manually checked. The main disadvantages of manually searching for controllers are: time consuming and human errors. For example, to find the controllers of a financial institute involved in a three-level SEs it can take a full day for two bank officers and sometimes, they may end up with a wrong result.

Recently, a project to automatically determine the controllers has been launched by BCB. The intention of it was the development of an algorithm and its implementation in a new system to establish the feasibility of an automation of the process of finding controllers. Though developed for BCB, the algorithm naturally works equally well for other financial institutions which have the same kinds of the applications.

This paper describes the basic concepts and business rules currently employed in BCB that have to be upheld by the algorithm, the subroutines for determining controllers at the first two levels (levels 1 and 2), the main algorithm, a brief complexity analysis of it and its application at the second level (for applications at first level, see [8]).

2 Concepts, Business Rules, and Example Description

The basic concepts and business rules for dealing with controllers have been established by BCB. In the following, through some simple examples, we describe the problem of controller calculation at two levels. It is useful to understand them to develop an efficient model to resolve the problem [1].

2.1 Basic Concepts

A Supervised Entity (SE) is a company which is registered at BCB. At any moment, BCB would like to know who the controllers of that SE are based on ownership of the stock shares. To effect the calculation of the controllers, only the stocks or quotas held by shareholders with the right to vote are considered.

- *Types of shareholder controllers*
  a) Natural Person (PF) - legal person represented by a single shareholder;
  b) Artificial Person (AP) - legal person represented by a group of shareholders;
  c) Individual Artificial Person (IAP, in Portuguese PJDD) - legal entities whose specific owners are not clearly defined.

The IAPs can be: cooperatives, associations and foundations, investment funds, entities of private providence, public companies headquartered in the exterior, financial institutions headquartered in the exterior, excepting solicitations of BCB.

An AP needs to be divided into PFs and/or IAPs who are the only entities to be considered as controllers. If an AP is found at a certain level, it surely indicates that there are upper levels of controllers at this SE. The society control of a SE is considered as definite when its final control is executed by one, and only one natural person (PF) or Individual Artificial Person (IAP) and indefinite when its final control is executed by more than one legal person.

- *Levels of the participation of the controllers in a Supervised Entity (SE)*

a) The first level: a SE is considered as on the first level when it contains only PFs and IAPs in the total composition of the stocks or quotas held by those with the right to vote.

b) The second, third and other levels: if a SE possesses some APs in their composition of voting shares, and for any AP it is possible to divide it into PFs and/or IAPs. Then his entity has more than one level of participation of the controllers. In our research, the algorithm was developed with the highest level being 3. This paper concentrates on the first two levels.

2.2 Business Rules

The business rules were established by BCB for a variety of situations. The following describes the rules for stock shares at levels 1 and 2.

- *At the level 1:*

The shareholders with the right of voting are considered to be the potential controllers in this work. If there is a large shareholder of a SE who holds more than fifty percent of the stock shares, he is already considered the unique controller of the company. If this condition does not exist, the calculation to find who the controllers are can use the following rules:

a) assessing all the possible combinations of shareholders: if the total of the stocks from such a combination is 50% plus one of the voting shares they are considered the effective controllers of the SE;

b) considering these combinations chosen in item "a", if we remove a shareholder with a determined percentage of stocks, and the total of the rest of the stocks remains larger than 50%, this removed shareholder will not be considered a controller of the SE.

- *At the level 2*

a) if a shareholder in level 2 is a unique controller of an AP, he is immediately considered as a unique shareholder PF of the AP at level 1 and, if he still possesses other shares in level 1, all his shares will be aggregated for calculating the controllers of the SE.
b) if there is no unique controller at level 2, all the controllers of this level are possible to be controllers. Therefore, no PF in an AP can sums his share within other APs and his share at level 1.
c) at level 1, if two or more APs are controlled by the same group of PFs, the shares of these APs will be summed together at level 1.
d) at level 1, if the APs are controlled by different groups of PFs at level 2, no stock shares of the same PF that constitutes within some APs and level 1 can be summed together.

2.3 Example Description

Suppose a Supervised Entity (SE), called BM Bank, with four stockholders: BM Automovies (AP1), Nelson da Silva (PF1), BM Insurance (AP2) and Andre de Barros (PF3), see Fig. 1. For BM Automovies (AP1), there are three stockholders: Nelson da Silva (PF1), Anderson Ferrnando (PF2), Marcelo Dantas (PF4). For GM Insurance (AP2), there are also three stockholders: Nelson da Silva (PF1), Anderson Ferrnando (PF2), Marcelo Dantas (PF4). All PFs in APs are considered at level 2. The first step of the analyses is to transfer APs to PFs and combine them into level 1. And then we have to find the controllers of the SE at level 1 (with only PF stockholders now). In section 4, we will present the solution for the example of BM Bank in Figure 1 using our algorithm.

3 Algorithm

To determine the controllers of a SE, three main subroutines are needed (resp. have been developed by us): a subroutine for determining controllers at level 1 (*sdcl1*), a subroutine for determining controllers at level 2 (*sdcl2*) and a subroutine for determining the possible combinations of the stock shares to form groups of controllers (*sdmlc*). *Sdmlc* arranges the most possible combinations of the stock shares for the group of controllers, but is still not proved that it can find all combinations. For every new combination, *sdcl1* is called to determine the controllers within *sdmlc*. When there is any AP to share the stock, *sdcl2* is used to analyze the stock shares of that AP and combine them to the level 1 as PFs. Figure 2 shows the general procedure to determine the controllers using these subroutines.

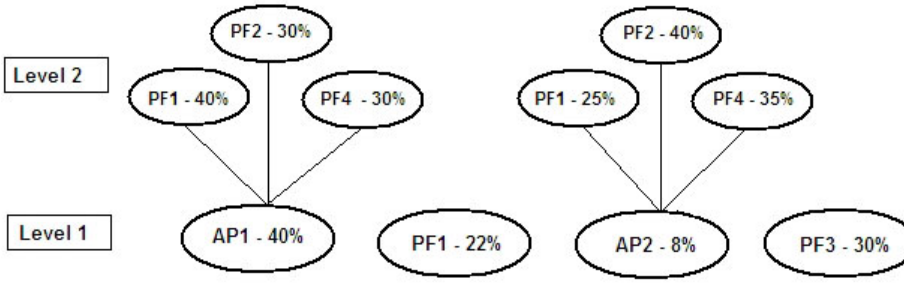


Fig. 1 The distribution of BM Bank stock shares

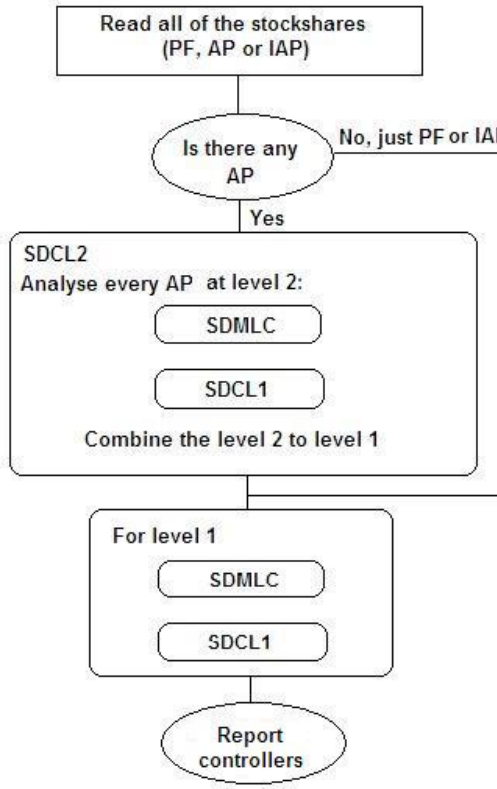


Fig. 2 The general procedure to determine the controllers

3.1 The Subroutine for Determining Controllers at Level 1 (*sdcl1*)

The task of this subroutine is to determine the controllers from a group of stock shares from a SE, if there are no APs. The stock shares of every shareholder are rep-

resented as percentages. The total of these percentages is 100%. The process of the algorithm is described in the following.

1) Rearrange the stock share percentages of the shareholders in decreasing order: A(1) > A(2) > … > A(k) > … > A(n), where k = 1, 2, …, n.
2) Test the first shareholder. If his/her shares equal 50%, all shareholders of the SE are controllers and this finishes the process. If his/her shares are larger than 50%, the shareholder is the only controller of the SE and this finishes the process.
3) Create a vector *Sum*(k) to add one by one the stock shares from A(1), A(2), … until the A(k) with which *Sum*(k) arrives at 50% or higher. If *Sum*(k) equals 50%, all of the shareholders are controllers, and we can stop the program. If it is higher than 50%, all the shareholders whose stock is in *Sum*(k) are controllers, k = 1, 2, …, s.
4) Calculate the key variable *Difference* = 50% - *Sum*(s). Test the stock shares from k=s+1: if the stock share is larger than *Difference*, the shareholder is a controller. Suppose this test stop at A(d), s< d < n.
5) Create a vector *Remainder_Sum* (k) which equals the sum of all remaining stock shares from k to n, k = 1, 2, …, n. For k = d+1, do the following tests. If the *Remainder_Sum* (k) is smaller than *Difference*, stop the program; if it equals *Difference*, all the remaining shareholders are controllers; if it is larger than *Difference*, the shareholder of A(k) is the controller. At the same time, a vector is generated i. e. V1(p) = V1(p) + A(k), k from d+1, V1(0) = 0, if V1(p) > *Difference*, V1(p) = V1(p) – A(k). Suppose this test stops at A(h), d < h < n, so p = 0, 1, 2, …, h - d.
6) There are still some additional possible controllers. The test now begins from k = h + 1. if (*Remainder_Sum* (k) + V1(p) = *Difference*) then all the remaining shareholders are controllers and we stop the program. If *Remainder_Sum* (k) + V1(p) > *Difference*, p = 0, 1, …, h - d, the shareholder of A(k) is a controller. Another vector is also generated i. e. V2(q) = V2(q) + A(k), k from h+1, V2(0) = 0, if V2(q) > *Difference*, V2(q) = V2(q) – A(k). Suppose this test stops at A(z), h < z < n, so q = 0, 1, 2, …, z - h.
7) If *Remainder_Sum* (k) + V1(p) < *Difference*, p = 0, 1, …, h, if *Remainder_Sum* (k) + V1(p) + V2(q) > *Difference* and V1(p) + V2(q) < *Difference*, then, the shareholder of A(k) is a controller. Suppose this test stops at A(z), h < z < n, so for V2(q), q = 0, 1, 2, …, z-h.
Remainder_Sum (k) + V1(p) + V2(q) < *Difference*, there is no more controller in this Supervised Entity. Stop. If z = n, it means all of the shareholders are controller, then stop the program.

3.2 The Subroutine for Determining the Possible Combinations (*sdmlc*)

The subroutine for determining the controllers at level 1 (*sdcl1*) was just developed to find controllers from a combination of the shareholders. To find more controllers, almost all of the combinations should be considered. *Sdmlc* is designed to combine almost all possible shareholders to find the possible groups of the controllers.

Suppose *I* is an upper level loop to alternate the combinations of stock shares. *J* is a lower level loop to alternate the combinations of stock shares. *I* x *J* are the total number of loops to generate the most combinations of shareholders A(k) to find possible controllers when using *sdcl1*.

1) Rearrange the stock share percentages of the shareholders in decreasing order: A(1) > A(2) > … > A(k) > … > A(n), where k = 1, 2, …, n.
2) Test the first shareholder. If his/her shares equal 50%, all shareholders of the SE are controllers and this finishes the process. If his/her shares are larger than 50%, the shareholder is the only controller of the SE and this finishes the process. In both cases, *I* equals 1 and *J* equals 1 too.
3) Define the value of *I*. To create a variable of *Sum* and add one by one of the stock share percentages from A(1) to *Sum* until it arrives 50%. If *Sum* equals 50%, all of the sharehold-

ers are controllers, and this finishes the process. If it is immediately menus 50%. All these shareholders are controllers, A(k), *k* = 1, 2, …, *s*. *l* = *k* - 1.

4) Create a vector *Remainder_Sum* (k) which equals the summary of all the rest of the stock share percentages from k = n-1, n-2, …, to 1.
5) For every i = 1, 2, …, *l* (*l* > 1) to define *J* (i). If i = 1, j = 1, call directly the subroutine for determining controller in level 1 - *sdcl1*() to find controllers.
   - if i = 1 then call *sdcl1*().
   - If i = 2, put the initial value j = 0, while (*Remainder_Sum* ((i-1) + (j+1)) – A((i-1) + (j+1)) ≥ 50%), do j = j + 1, temporarily eliminate A(i-1),…,A(j+1) to form a new group of stock shares and call *sdcl1*().
   - If i = 3, put the initial value j = 0, while (A(1) + *Remainder_Sum* ((i-1) + (j+1)) – A((i-1) + (j+1)) ≥ 50%), do j = j + 1, temporarily eliminate A(i-1),…,A(j+1) to form a new group of stock shares and call *sdcl1*().
   - If i > 3, put the initial value j = 0, while (A(1) + … + A((i-1)-1) + *Remainder_Sum* ((i-1) + (j+1)) – A((i-1) + (j+1)) ≥ 50%), do j = j + 1, temporarily eliminate A(i-1),…,A(j+1) to form a new group of stock shares and call *sdcl1*().

3.3 The Subroutine for Determining Controllers at Level 2 (*sdcl2*)

This subroutine was developed to extend the function of *sdcl1* if there are some APs in a SE. If *n* elements of the stock shares of the SE, *A(k)*, *k*=1,.., *n*, are considered as at the level 1, *m* elements of an AP of the SE, *Ap(k,i)*, i = 1,□ , *m*, are at the level 2. In this paper, all *Ap(.,i)* are PFs, i.e. the algorithm covers up to level 2. The main objective of *sdcl2* is to analyze the elements at level 2 and combine them into level 1.

1) Read all elements of A(k) of a SE, where k = 1, …, n: if A(k) is an AP, then go to step 2; if there is no any AP, this means that all elements are PFs and there is no level 2, then do *sdmlc/sdcl1* for A(.).
2) For every AP, create a matrix Ap(k,i), i = 1, …, m, where k is according to the k of A(k), m is the total number of PFs in A(k). All of the elements of Ap(k,.) are at level 2. Suppose, K is the total number of APs in A(.).

2.1) For all k ε K, do *sdmlc/sdcl1* on Ap(k,i), i = 1, …, m:
if for any Ap(k,i), which is a only controller for this AP, then use this PF, Ap(k,i), to substitute A(k). Read A(k), k = 1,…, n, if there is another same PFs as Ap(k,i) in A(.), sum it to A(k) and eliminate it from A(.);
if there is no unique controller at Ap(k,.), create a vector of Apc(k, j), j = 1,…,p for all k ε K, where p is the number of controllers of Ap(k,.).

2.2) Compare Apc(k,j), j = 1, …, p, with Apc(k', j), j = 1, …, p, for all k, k' ε K and k ≠ k',
if there are APs with same PFs, then sum the shares of Apc(k,.) and Apc(k',.), and eliminate Apc(k',.).

2.3) After the possible elimination, a new vector of A'(.) is formed, where, k = 1, …, n' and n' < n.
Do *sdmlc/sdcl1* on A'(.) and get a new vector A_controller(k''), k'' = 1, …, n'', n'' < n' and there are K'' of APs in A_controller(.).

3) In A_controller(.), there are still some APs, but all of them are combined into the level1. The following steps are to used to substitute the APs by their PFs.

3.1) For all k'' ε K" , using the elements of Apc(k'',.) to substitute A_controller(k''), where Apc(k'',.) is all PFs of AP of k''.

3.2) In the A_controller(.), all elements are PFs. There are still some repeated PFs. For k'' = 1, …, n'', eliminate the repeated A_controller(k''). Put the new element in vector A_controller_final(.).

4 Complexity Analysis

To determine the controllers from their stock shares is similar to the combinational problem *Sum of Subset*. A backtracking algorithm was developed with a possible search space which includes all of the possible solutions [2, 3]. Unfortunately, this algorithm, in the worst case, has the complexity $O(p(n)\ 2^n)$ or $O(p(n)\ n!)$, where $p(n)$ is a n^{th} order polynomial [2, 3]. Therefore, this algorithm is only of limited value to the practical problems of BCB.

The developed subroutines (*sdcl1*, *sdcl2* and *sdmlc)* take advantage of the features of the actual problem instance. For decreasing the number of combinations of stock shareholders, some criteria are established to reduce the dimensions of the search space significantly compared to the backtracking algorithm. For example, in *sdcl1*, if *Remainder_Sum* (k) + V1(p) + V2(q) < *Difference*, then A(k) is not a controller and there are no more controllers in this group of stock shares. In *sdmlc*, if A(1) + □ + A((i-1)-1) + *Remainder_Sum* (((i-1) + (j+1)) □A((i-1) + (j+1)) < 50%) then, there are no more potential controllers.

- *Complexity analysis of sdcl1*

  The complexity analysis of the main steps of *sdcl1* is illustrated in Table 1.

Table 1. Complexity analysis for *sdcl1*

Step	Method	Complexity	Condition
1	merge sort	O(*n log n*)	
3	while iterative	O(s)	$1 < s < d$
4	while iterative	O(*d-s*),	$s < d < h$
5	for iterative while iterative	O(*n*) O(*h-d*),	 $d < h < z$
6-8	while and 2 for iterative	O(*p* x *q* x (*z-h*))	$h < z < n$, $1 < p < (h$ □ *d)*, $1 < q < (z$ □*h)*

From the step 6 to 8, $O(p \times q \times (z-h)) < O(h \times z \times (z-h)) < O(z^3-h^3) < O((z-h)^3)$. After simplifying the results in Table 1, the complexity of *sdcl1* is *max*(O(*n log n*), $O((z-h)^3)$), where $1 < h < z < n$.

- *Complexity analysis of sdmlc*

  The complexity analysis of the main steps of *sdmlc* is shown in Table 2.

  From step 5, the complexity of *sdmlc* is max ($O(I \times J \times n \log n)$, $O(I \times J \times (z-h)^3)$)), where $1 < I < d < h < z < n$, and $1 < J < n$. *I* and *J* are usually given parameters. Due to max ($O(I \times J \times n \log n)$, $O(I \times J \times (z-h)^3)$)) < $O(n^3)$, the complexity of *sdmlc* is $O(n^3)$.

- *Complexity analysis of sdcl2*

  The main operations of *sdcl2* are to call *sdcl1* and *sdmlc* for some APs of a SE. Other operations may involve the necessary comparison, etc. Basically, the complexity of *sdcl2* is similarly as *sdmlc*.

Table 2. Complexity analysis of *sdmlc*

Step	Method	Complexity	Condition
1	merge sort	O(n *log* n)	
3	while iterative	O(I)	$1 < I < h$
4	for iterative	O(n)	
5	for iterative while iterative call *sdcl1*()	max (O(I x J x (n *log* n)), O(I x J x (z-h)^3))	$I < I < h$ $J = max(J(i))$ $h < z < n$

5 An Example

The following example of the application of *sdcl1* and *sdmlc* is taken from [8]. In this section, *sdcl2* was used to find the controllers for a SE, BM Bank, as in Fig. 1. In this example, there are two APs (BM Automovies and BM Insurance) and other two PFs that hold the stocks of BM Bank. To find the stock controllers of that bank, the process of applying *sdcl2* is described in Figure 3. The final stock controllers of BM Bank are PF1, PF2, PF3 and PF4. This example is just aimed at showing the problems of real instances of our problem.

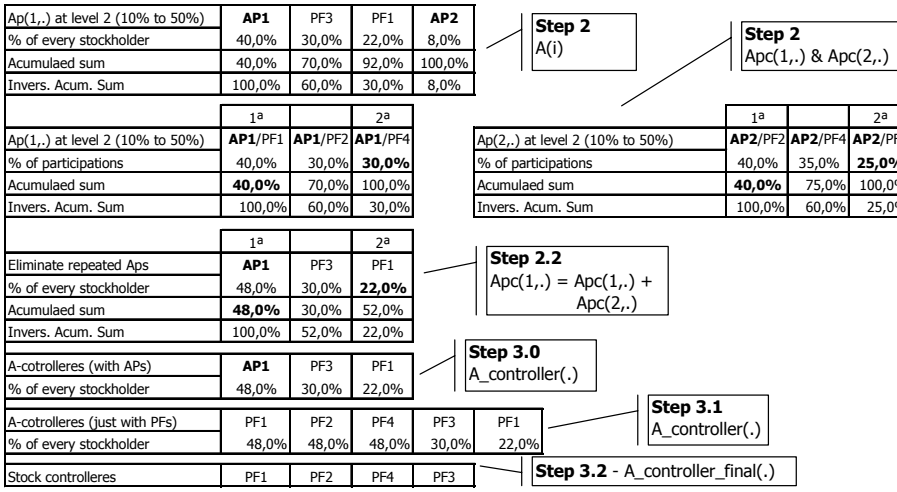


Fig. 3 The illustration of the application of algorithm: *sdcl2*

6 Conclusions

An algorithm for determining the stock controllers of some financial institutions for the Brazilian Central Bank (BCB) was developed and implemented. Initial analysis shows that the proposed algorithm has polynomial complexity $O(n^3)$. It can be used to find almost all possible controllers of a SE, though how near to a complete solution it comes has to be further evaluated. The paper described the three subroutines dealing with the first two levels, but in the real situation, to determine the controllers through stock shares for a SE, sometimes, may require up to 10 levels. The algorithm for the first two levels can be basically adapted for higher levels. But BCB asked us to just report the first two levels in this paper.

The developed algorithm was implemented by Politec Informatics Ldta, Brazil and is currently employed at Unicad/BCB. *Natural* is a basic language which was used to implementation. All the codes are adapted in the computing operation system of Unicad (National Unique Registration) of Brazil and integrated with the database of BCB. The experimental results indicate the feasibility of automating the process of finding controllers using our approximation algorithm instead of the manually operation. What took two agents a full day of work has now been reduced to just minutes of computation in BCB's computer system. Though developed for BCB, it is expected to work equally well for other financial institutions which are controlled by stockholders. The future work will involve to combine methods of Artificial Intelligent and mathematics models to improve the efficiency and reduce human involvement in the process even more.

References

1. Banco Central do Brasil: Regras para se calcular o controlador de uma entidade supervisionada pelo Banco Central do Brasil, (2001).
2. Cormen, T. H., Leiserson, C. E. and Rivest, R. L.: Introduction to Algorithms, The MIT Press, Combridge, USA, (1996).
3. Horowitz, E., Sahni, S. and Rajasekaran, S.: Computer Algorithms, Computer Science Press, New York, USA, (1998).
4. Garey, M. R., and Johnson , D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company (1979).
5. Brickell, E. F.: Solving low density knapsacks, *Advances in Cryptology, Proceedings of Crypto '83*, Plenum Press, New York (1984), 25-37.
6. Lagarias, J. C., and Odlyzko, A. M.: Solving low-density subset sum problems, *J. Assoc. Comp. Mach.* 32(1) (1985), 229-246.
7. Radziszowski, S. and Kreher, D.: Solving subset sum problems with the L^3 algorithm, *J. Combin. Math. Combin. Comput.* 3 (1988), 49-63.
8. Branco, V. G. F., Weigang, L., Ribeiro, A. F., Shibata, W., Abad, M. P. E., Torres, N. V., Dib, M. V. P. and de Andrade, V. M.: An Algorithm for Determining the Controllers of Supervised Entities at the First Level: Brazilian Central Bank Case Study, Proc. of FSKD'02 - 1st International Conference on Fuzzy Systems and Knowledge Discovery, pp. 240-244, Singapore, (2002).

Two Factor Option Pricing with Uncertain Volatility*

David M. Pooley[1], Peter A. Forsyth[1], and Ken R. Vetzal[2]

[1] School of Computer Science
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
dmpooley@elora.math.uwaterloo.ca, paforsyt@elora.math.uwaterloo.ca
[2] School of Accountancy
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
kvetzal@watarts.uwaterloo.ca

Abstract. The practical difficulty of estimating volatility values for underlying assets has lead to the uncertain volatility model for option pricing in finance. The pricing equations in this model are often cast in the form of nonlinear partial differential equations. It is well known that for one factor problems, these equations can be numerically solved by selecting volatility values according to the sign of gamma. However, with two (or more) factors, a small, constrained optimization problem must be solved for each node at every timestep. In this paper, we discuss some technical details of solving these optimization problems in the context of the overall numerical solution process. Examples are provided for max of two asset problems, as well as two asset butterfly options.

1 Introduction

Determining model parameters for option pricing problems can be a difficult task. For example, unlike prices and interest rates, volatility and correlation values between assets are not directly observable in the market. Often these values are estimated from historical data. However, deciding which data to use, and how much data to use, is rarely clear. Further, past trends may not reflect future trends, and parameter estimates can change as the historical data used in calculations changes. Another approach for obtaining unobservable model parameters is to back them out from observed option prices. While this often leads to consistent pricing models, sufficient market data may not always be available.

Instead of trying to determine a single value for a parameter, it may be easier to specify a range of possible values. In other words, let the parameter be uncertain. Unfortunately, this makes the underlying problem more complicated to specify and solve. In particular, partial differential equation (PDE) based models become nonlinear.

* This work was supported by the Natural Sciences and Engineering Research Council of Canada, the Social Sciences and Humanities Research Council of Canada, and the Royal Bank of Canada.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 158–167, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Perhaps the first studies to examine uncertain parameters in finance were [1] and [2]. Both of these studies looked at uncertain volatility for one asset option pricing models, and derived a nonlinear PDE for pricing the options. Since that time, a large number of papers have extended the basic analysis, either by adding extra features, changing the underlying process, or examining numerical behaviour. It is interesting to note that the Leland model of transaction costs is mathematically identical to the one factor uncertain volatility problem [3].

Moving beyond one factor problems, [4] examined uncertainty in multi-factor reverse convertible contracts. However, in all examples provided, the uncertainty reduced to a single parameter. Uncertain correlation for rainbow options was studied in [5], but again, this only involved uncertainty in a single parameter. To the authors' knowledge, no previous study has examined uncertainty in more than one parameter for multi-factor problems.

In this paper, we study problems in which uncertainty in one parameter has a non-trivial impact on the uncertainty of another parameter. Specifically, we examine two factor option pricing problems where the volatilities of both underlying assets are uncertain. For completeness, we also add uncertain correlation. The end result, as will be shown below, is that a small optimization problem must be solved at each node for every timestep. Obviously, this greatly increases the computational cost of solving two factor problems. Note that we could also have an uncertain interest rate, although this would be a trivial addition since the interest rate and volatilities (correlation) do not appear together in the pricing equations. As such, selecting the appropriate interest rate would be independent from the volatility selection.

The organization of this paper is as follows. In section 2, we review the basics of the uncertain volatility model when applied to two or more underlying assets. Important implementation issues for two factor problems are discussed in section 3. In particular, we discuss the discretization scheme, connections between a theoretical and numerical treatment of the uncertain pricing equations, and the steps required in selecting volatility values for each node at each timestep. Section 4 provides numerical examples involving max of two asset payoffs and butterfly payoffs. Finally, conclusions are made in section 5.

2 Basic Background

Following standard arguments, the PDE for the fair price of a contingent claim in the Black-Scholes model is determined by

$$U_\tau = -\mathbf{V} \cdot \nabla U + (\mathbf{D}\nabla) \cdot \nabla U - rU \ , \tag{1}$$

where $\mathbf{V}$ is a velocity vector, $\mathbf{D}$ is a diffusion tensor, r is the risk-free interest rate, and $\tau = T - t$ is time in the backwards direction. Details concerning the derivation of this equation can be found in [6] and the references therein.

Volatility parameters typically appear in the diffusion tensor of (1). When volatility is uncertain, the diffusion tensor is no longer unique. For example,

consider a vanilla put or call option on a single asset with value S. If we constrain the volatility, σ, to lie in the range

$$\sigma_{\min} \leq \sigma \leq \sigma_{\max} \ ,$$

then the diffusion can have values in the range

$$\frac{S^2\sigma_{\min}^2}{2} \leq [\mathbf{D}]_{1,1} \leq \frac{S^2\sigma_{\max}^2}{2} \ .$$

With uncertainty in the diffusion term, (1) does not have a unique solution. Nevertheless, the maximal and minimal solution values of the contingent claim equation are unique. Mathematically, if we let $\sigma \in \Omega$ imply that the volatility values (perhaps one for each underlying factor) are constrained to lie within values specified by the region Ω, then we can uniquely solve

$$\text{Max price:} \quad U_\tau = -\mathbf{V} \cdot \nabla U + \sup_{\sigma \in \Omega} ((\mathbf{D}\nabla) \cdot \nabla U) - rU \tag{2}$$

$$\text{Min price:} \quad U_\tau = -\mathbf{V} \cdot \nabla U + \inf_{\sigma \in \Omega} ((\mathbf{D}\nabla) \cdot \nabla U) - rU \ . \tag{3}$$

From a contract perspective, the maximal price gives the best case for the buyer of the option, or the worst case for the seller of the option. Similarly, the minimal price corresponds to the worst case for the buyer, and the best case for the seller. By charging the maximal price, the seller can hedge the option and always ensure a non-negative balance in the hedging portfolio.

For a standard one factor call or put problem, the discrete equations can be maximized or minimized by simply examining the sign of the second derivative (gamma) [1]. Thus, the discrete equations are maximized by setting σ to $\sigma_{\max}$ when gamma is positive, and to $\sigma_{\min}$ when gamma is negative. Minimizing the equations requires the opposite behaviour. In cases where gamma always has the same sign, the volatility always assumes the same value, and the problem reverts to being linear. This occurs whenever the payoff is strictly convex (or concave), such as for a regular call or put option.

With two or more factors, a small constrained optimization problem must be solved at each node and each timestep to determine the appropriate volatility values. The optimization problem will depend on the second derivative values. However, as will be discussed below, one must be careful in how the second derivative values are computed. The optimization must treat derivative values consistently with the discrete equations.

Although the maximal and minimal prices of the pricing equation are unique, the numerical solution of a nonlinear PDE is not guaranteed to be unique. When numerically solving (2) or (3), we wish to guarantee convergence to the viscosity solution. It was shown in [7] that the viscosity solution is the correct solution in a financial context. That work also showed that a consistent, stable and monotone scheme will always converge to the viscosity solution. Conditions which ensure these properties for a variety of one factor discretizations of uncertain volatility problems were derived in [8]. Unfortunately, the techniques used in that paper

do not immediately apply to the two factor problem. The main difficulty is that most discretizations for two factor problems do not lead to M-matrices, unless the cross derivative term is zero. Nevertheless, previous experience has shown that implicit methods always converge to the correct answer. While this has not been proven for multi-factor uncertain parameter problems, we do not anticipate problems.

3 Implementation Issues

3.1 Discrete Optimization Equations

To discretize (1), we use a finite volume method as described in [6]. Our computational domain is tiled by triangles, and we use the usual C^0 basis functions. For full details of this discretization, and a discussion of why the finite volume method may be preferable to simpler finite difference methods, see [6].

For concreteness, we will study the effect of uncertain volatility on two factor European options. Modifications for American or more exotic options are minor. For standard European options, volatility terms only appear in the diffusion tensor. For vanilla options on two assets, the diffusion matrix is symmetric positive definite, and has the form

$$\mathbf{D} = \frac{1}{2}\begin{bmatrix} \sigma_1^2 S_1^2 & \rho\sigma_1\sigma_2 S_1 S_2 \\ \rho\sigma_1\sigma_2 S_1 S_2 & \sigma_2^2 S_2^2 \end{bmatrix} \quad . \tag{4}$$

The velocity vector in this case is

$$\mathbf{V} = -\begin{bmatrix} rS_1 \\ rS_2 \end{bmatrix} \quad . \tag{5}$$

With uncertain volatility, we have:

$$\sigma_{1,\min} \le \sigma_1 \le \sigma_{1,\max}$$
$$\sigma_{2,\min} \le \sigma_2 \le \sigma_{2,\max} \quad .$$

In addition, we allow for uncertain correlation between the two underlying assets:

$$\rho_{\min} \le \rho \le \rho_{\max} \quad . \tag{6}$$

When maximizing or minimizing the discrete equations, we are interested in the discrete form of the diffusion term. Expanding this term is tedious and not very enlightening. Further, when constructing the discrete equations, it is common to loop over the elements of the mesh. We can then consider the contribution to equation i by triangle t containing node i. Let the remaining two nodes of triangle t be labeled as p and q. Finally, let A_t be the area of triangle t, and let N_i be the basis function for node i in triangle t used in the finite volume discretization. After some tedious algebra, we can write the contribution to the diffusion term in equation i by triangle t in matrix form as

$$\text{Contribution from triangle } t = \begin{bmatrix} \sigma_1 & \sigma_2 \end{bmatrix} \begin{bmatrix} a_t & \rho c_t \\ \rho c_t & b_t \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \tag{7}$$

where

$$a_t = \frac{A_t S_1^2}{2}\frac{\partial N_i}{\partial S_1}\left(\frac{\partial N_p}{\partial S_1}(U_p - U_i) + \frac{\partial N_q}{\partial S_1}(U_q - U_i)\right)$$
$$b_t = \frac{A_t S_2^2}{2}\frac{\partial N_i}{\partial S_2}\left(\frac{\partial N_p}{\partial S_2}(U_p - U_i) + \frac{\partial N_q}{\partial S_2}(U_q - U_i)\right)$$
$$c_t = -\frac{A_t S_1 S_2}{4}\left(\frac{\partial N_i}{\partial S_1}\frac{\partial N_p}{\partial S_2}(U_p - U_i) + \frac{\partial N_i}{\partial S_2}\frac{\partial N_p}{\partial S_1}(U_p - U_i) + \frac{\partial N_i}{\partial S_1}\frac{\partial N_q}{\partial S_2}(U_q - U_i) + \frac{\partial N_i}{\partial S_2}\frac{\partial N_q}{\partial S_1}(U_q - U_i)\right)$$

To maximize or minimize the diffusion, we need to consider contributions from all triangles attached to node i. Hence, the optimization problem for node i has the form

$$\begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}\begin{bmatrix}\Sigma a_t & \rho\Sigma c_t \\ \rho\Sigma c_t & \Sigma b_t\end{bmatrix}\begin{bmatrix}\sigma_1 \\ \sigma_2\end{bmatrix} , \tag{8}$$

where the sums are over all triangles attached to node i.

Unfortunately, it is hard to extract meaning from (8). To help understanding, we can compare (8) with an expansion of the original equation. This is done in the next section.

3.2 Connection Between Analytic and Discrete Forms

Equation (8) in the previous section provides little insight into the optimization problem. To gain further understanding, consider (1) directly. To maximize or minimize the option value, it suffices to maximize or minimize the diffusion term. To do this, we employ a forward difference approximation for the time derivative, expand the diffusion operator and rearrange terms to obtain

$$U^{n+1} = U^n + \Delta t\left(\frac{\sigma_1^2 S_1^2}{2}\frac{\partial^2 U}{\partial S_1^2} + \frac{\sigma_2^2 S_2^2}{2}\frac{\partial^2 U}{\partial S_2^2} + \rho S_1 S_2 \sigma_1 \sigma_2 \frac{\partial^2 U}{\partial S_1 \partial S_2}\right) + \text{other terms.} \tag{9}$$

Thus, we are looking to optimize the expression

$$\frac{\sigma_1^2 S_1^2}{2}\frac{\partial^2 U}{\partial S_1^2} + \frac{\sigma_2^2 S_2^2}{2}\frac{\partial^2 U}{\partial S_2^2} + \rho\sigma_1\sigma_2 S_1 S_2 \frac{\partial^2 U}{\partial S_1 \partial S_2} . \tag{10}$$

By setting the derivative with respect to ρ equal to zero, we see that ρ must attain one of its extreme values. Which value depends on the sign of $\frac{\partial^2 U}{\partial S_1 \partial S_2}$ (assuming that S_1, S_2, σ_1 and σ_2 are all non-negative). For example, if this term is positive, we should set ρ to $\rho_{\max}$ when maximizing the option price. With the value of ρ decided, we are left with a constrained optimization problem of the following quadratic form

$$\begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}\begin{bmatrix}a & \rho c \\ \rho c & b\end{bmatrix}\begin{bmatrix}\sigma_1 \\ \sigma_2\end{bmatrix} , \tag{11}$$

where $a = \frac{S_1^2}{2}\frac{\partial^2 U}{\partial S_1^2}$, $b = \frac{S_2^2}{2}\frac{\partial^2 U}{\partial S_2^2}$ and $c = \frac{S_1 S_2}{2}\frac{\partial^2 U}{\partial S_1 \partial S_2}$.

It appears that we need to compute second derivatives to optimize the volatility values. However, a naive implementation of this approach will lead to incorrect results. In the above, we have shown the steps required to optimize the analytic pricing equation. In practice, we should be numerically optimizing the discrete equations. Thus, if we were to select volatility values by numerically approximating the derivatives values in (11), then the numerical derivative approximations must be consistent with the discretization.

In our case, the derivative approximation scheme outlined in [9] is consistent with our discretization method. Specifically, the terms in the matrix of (8) are these derivative estimates (up to a constant factor related to the areas of surrounding triangles, proof omitted). Hence, we could use (11) to select the volatility values if we approximate the derivative values using the method of [9]. In general, any other derivative approximation method is inconsistent with our discretization. In practice, using another method would mean that the discrete equations are not being maximized (minimized), and iterative solution methods of the nonlinear equations might oscillate between states. For this reason, it makes sense to optimize the actual discrete equations ((8) in our case), instead of worrying about inconsistent derivative estimation techniques.

3.3 Optimization Details

In both the theoretical and numerical cases, we need to optimize a quadratic expression with linear constraints. Formally, we have

$$\mathbf{x}^T\mathbf{M}\mathbf{x} = \begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}\begin{bmatrix}a & c\\ c & b\end{bmatrix}\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} \tag{12}$$

subject to

$$\begin{aligned}\sigma_{1,\min} \le \sigma_1 \le \sigma_{1,\max}\\ \sigma_{2,\min} \le \sigma_2 \le \sigma_{2,\max}\ .\end{aligned}$$

Note that optimizing for ρ is assumed to have already occurred (and is embedded in c). For now we assume a minimization problem. Let $\mathbf{e}_1 = (1,0)'$ and $\mathbf{e}_2 = (0,1)'$. The Kuhn-Tucker conditions ([10]) then imply

$$-2\mathbf{M}\mathbf{x} - \mathbf{e}_1\lambda_1 + \mathbf{e}_1\lambda_2 - \mathbf{e}_2\lambda_3 + \mathbf{e}_2\lambda_4 = \emptyset \tag{13}$$

where

$$\begin{aligned}&\lambda_1 \ge 0 \text{ and } \lambda_1 = 0 \text{ if } \sigma_1 > \sigma_{1,\min}\\ &\lambda_2 \ge 0 \text{ and } \lambda_2 = 0 \text{ if } \sigma_1 < \sigma_{1,\max}\\ &\lambda_3 \ge 0 \text{ and } \lambda_3 = 0 \text{ if } \sigma_2 > \sigma_{2,\min}\\ &\lambda_4 \ge 0 \text{ and } \lambda_4 = 0 \text{ if } \sigma_2 < \sigma_{2,\max}.\end{aligned}$$

If the optimal values occur in the interior of the region defined by the extreme volatility values, then $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$. We then have

$$\mathbf{M}\mathbf{x} = \emptyset.$$

Table 1. Model parameters used for testing.

Time to expiry	0.5 years
Type	Call - worst case (seller)
Payoff	$\max(\max(S_1, S_2) - K, 0)$
r	0.05
strike (K)	40
$\sigma_{1,\min}, \sigma_{2,\min}, \rho_{\min}$	0.3
$\sigma_{1,\max}, \sigma_{2,\max}, \rho_{\max}$	0.5

If $\mathbf{M}$ is non-singular, then $\mathbf{x} = (0, 0)$ is a solution. This will almost surely be an infeasible solution for option pricing problems. Hence, the optimal solution will be on the boundary. If $\mathbf{M}$ is singular, then the columns are linearly dependent. We must then check if the optimal solution is on the line defined by $a\sigma_1 + c\sigma_2 = 0$.

In any case, we must also check the four corners of the region, as well as the four edges. For example, if we assume that $\sigma_1 = \sigma_{1,\max}$, then we compute

$$\frac{\partial}{\partial \sigma_2}(a\sigma_{1,\max}^2 + b\sigma_2^2 + 2c\sigma_{1,\max}\sigma_2) = 0$$
$$\rightarrow \sigma_2 = \frac{-c\sigma_{1,\max}}{b} \quad .$$

Of course, we need to check if the computed σ_2 value is feasible, and compare to all previously computed optimal values. The remaining three edges are handled similarly.

4 Results

Our first test is for a simple European call option. All model parameters are given in Table 1. A frozen coefficient iterative scheme [11] with tolerance of 10^{-6} was used to solve the nonlinear equations at each timestep. Convergence results for at-the-money values ($S_1 = S_2 = 40$) are given in Table 2.

From the table, we see that both implicit and Rannacher timestepping are converging to the same solution. By Rannacher timestepping, we mean Crank-Nicolson timestepping preceded by 2 fully implicit steps [12]. From previous work ([8], [13]), we expect this solution to be the correct viscosity solution, although we have no theoretical proof. Rannacher timestepping converges at a quadratic rate, while implicit timestepping appears to be dropping to a linear rate. Crank-Nicolson timestepping is either converging to a non-viscosity solution, or has a slightly growing instability.

Extrapolating the results from Table 2 gives a worst-case value of 9.937. Similarly, we can obtain a best-case value of 5.831. Clearly, the spread between the best-case and worst-case prices is relatively large. Even though the volatility and correlation values can vary by a maximum of 20 percent, the price varies by over 40 percent.

Table 2. Convergence results for an at-the-money ($S_1 = S_2 = 40$) call option with parameters as given in Table 1. The timestep is halved at each grid refinement. "Difference" is the absolute value of the change in the solution as the grid is refined. "Ratio" is the ratio of successive differences.

	Fully Implicit			Crank-Nicolson			Rannacher		
Nodes	Value	Difference	Ratio	Value	Difference	Ratio	Value	Difference	Ratio
41x41	9.8172			10.042			9.8571		
81x81	9.8946	0.0774		10.247	0.205		9.9163	0.0592	
161x161	9.9205	0.0259	2.98	10.369	0.123	1.67	9.9318	0.0155	3.82
321x321	9.9300	0.0095	2.73	10.445	0.076	1.61	9.9357	0.0039	3.94

We can also compute max and min prices for constant volatility and correlation values for the same model parameters. Specifically, the max price was computed with $\sigma_1 = \sigma_{1,\max}$, $\sigma_2 = \sigma_{2,\max}$ and $\rho = \rho_{\min}$, whereas the min price was computed with $\sigma_1 = \sigma_{1,\min}, \sigma_2 = \sigma_{2,\min}$ and $\rho = \rho_{\max}$. The results were 9.937 and 5.831 respectively – the same values as for the worst case and best case pricing.

Such a result is not immediately obvious, since even though U_{xx} and U_{yy} are both non-negative, U_{xy} is non-positive for a regular call option on the max of two assets (which is why the correlation is set to the smallest value to obtain the largest option price, and the largest value to get the smallest option price). Hence, maximizing or minimizing (10) is not necessarily trivial. Further, it is important to note that the numerical scheme for the uncertain case did not always set the volatility and correlation values to the same levels as for the max and min pricing values at all nodes for each timestep. Due to the discrete form of the equations, the optimization parameters given by (8) often had different signs than would be expected from (10). In other words, the numerical second derivative values sometimes had a different sign than what theory would predict. However, by optimizing the discrete equations, the problem converges to the correct answer. Had we approximated the derivative values by some other method (potentially more accurate), the discrete equations would not be properly optimized. In fact, doing so often caused the iterative method used to solve the nonlinear discrete equations to oscillate indefinitely between two (or more) states.

Our next test keeps the same parameters as in Table 1, except that the payoff has been changed to a butterfly. In particular, the payoff is

$$\begin{aligned} S_{max} &= \max(S_1, S_2) \\ U(S_1, S_2, \tau = 0) &= \max(S_{max} - K_1, 0) + \max(S_{max} - K_2, 0) \\ &\quad -2\max(S_{max} - (K_1 + K_2)/2, 0) \ , \end{aligned} \tag{14}$$

where we have set $K_1 = 35$ and $K_2 = 45$. In some sense, this test is more challenging, since the signs of U_{xx} and U_{yy} change over the solution domain. Convergence results for a worst case pricing are given in Table 3.

Table 3. Convergence results for an at-the-money ($S_1 = S_2 = 40$) butterfly spread call option with parameters as given in Table 1 and payoff specified by (14). The timestep is halved at each grid refinement. "Difference" is the absolute value of the change in the solution as the grid is refined. "Ratio" is the ratio of successive differences.

	Fully Implicit			Crank-Nicolson			Rannacher		
Nodes	Value	Difference	Ratio	Value	Difference	Ratio	Value	Difference	Ratio
41x41	1.64661			2.354			1.63913		
81x81	1.65254	0.00592		2.612	0.258		1.64997	0.01083	
161x161	1.65544	0.00291	2.03	2.790	0.179	1.44	1.65507	0.00510	2.12
321x321	1.65709	0.00165	1.76	2.908	0.118	1.52	1.65744	0.00237	2.16

Table 4. Option values at $(S_1, S_2) = (40, 40)$ for a butterfly payoff.

Test	Value
Uncertain worst case	1.660
$\sigma_1 = 0.3, \sigma_2 = 0.3, \rho = 0.3$	1.102
$\sigma_1 = 0.3, \sigma_2 = 0.3, \rho = 0.5$	1.098
$\sigma_1 = 0.4, \sigma_2 = 0.4, \rho = 0.4$	0.864
$\sigma_1 = 0.5, \sigma_2 = 0.5, \rho = 0.3$	0.716
$\sigma_1 = 0.5, \sigma_2 = 0.5, \rho = 0.5$	0.708
Uncertain best case	0.336

From the table, we see that implicit timestepping is converging to the (presumably correct) answer at a linear rate. Rannacher timestepping is converging to the same answer at what appears to be a linear rate as well. Crank-Nicolson timestepping is either converging to a non-viscosity solution, or has a slightly growing instability. These results are consistent with those seen for one factor uncertain volatility [8] and for passport options [13].

As indicated by (14), a butterfly payoff can usually be considered as the sum of 4 separate call options: long one call at strike K_1, short two calls at strike $(K_1+K_2)/2$ and long one call at strike K_2. However, with uncertain parameters, the equations are nonlinear. Hence, the sum of these 4 call options is not the same as an option with the payoff given by (14). For comparison, Table 4 gives prices of butterfly options using fixed volatility and correlation values. We see that the uncertain values form an envelope around the fixed parameter prices.

5 Conclusions

In this paper we have described the steps required to price two factor option pricing problems with uncertain volatility and correlation. The techniques described would also work for other uncertain parameters.

When solving the nonlinear equations numerically, it is important to optimize the discrete form of the equations. We showed how the discrete form of the

equations are consistent with a specific numerical approximation of the second derivative values. Using a different approximation for the derivative values during the optimization can cause the numerical scheme to fail, in the sense that an iterative solver may loop indefinitely between two or more states.

Examples were provided for a simple call on the max of two assets, as well as a butterfly type payoff. In both cases, implicit and Rannacher timestepping converged to the same (presumably correct) solution, while Crank-Nicolson timestepping either converged to a non-viscosity solution, or had a slowly growing instability. For the call payoff, Rannacher timestepping showed quadratic convergence. Unfortunately, convergence was only linear for the butterfly payoff. In general, we can only expect linear convergence for Rannacher timestepping when solving nonlinear equations.

References

1. Lyons, T.: Uncertain volatility and the risk free synthesis of derivatives. Applied Mathematical Finance **2** (1995) 117–133
2. M. Avellaneda, A.L., Parás, A.: Pricing and hedging derivative securities in markets with uncertain volatilities. Applied Mathematical Finance **2** (1995) 73–88
3. Forsyth, P.A., Vetzal, K.R.: Implicit solution of uncertain volatility/transaction cost option pricing models with discretely observed barriers. Applied Numerical Mathematics **36** (2001) 427–445
4. Topper, J.: Uncertain parameters and reverse convertibles. RISK (2001) 4–7
5. Topper, J.: Worst case pricing of rainbow options. Discussion paper No. 217, version October 12 (2001)
6. Zvan, R., Forsyth, P.A., Vetzal, K.R.: A finite volume approach for contingent claims valuation. IMA Journal of Numerical Analysis **21** (2001) 703–731
7. Barles, G.: Convergence of numerical schemes for degenerate parabolic equations arising in finance. In Rogers, L.C.G., Talay, D., eds.: Numerical Methods in Finance. Cambridge University Press, Cambridge (1997) 1–21
8. Pooley, D.M., Forsyth, P.A., Vetzal, K.R.: Numerical convergence properties of option pricing PDEs with uncertain volatility. University of Waterloo, In press (2003)
9. Zienkiewicz, O., Zhu, J.: Automatic directional refinement in adaptive analysis of compressible flow. International Journal for Numerical Methods in Engineering **37** (1994) 2189–2210
10. Taha, H.A.: Operations Research: An Introduction. Fourth edn. Macmillan Inc., 866 Third Avenue, New York, New York 10022 (1982)
11. Clift, S., Forsyth, P.: Linear and non-linear iterative methods for the incompressible Navier-Stokes equations. International Journal for Numerical Methods in Fluids **18** (1994) 229–256
12. Rannacher, R.: Finite element solution of diffusion problems with irregular data. Numerische Mathematik **43** (1984) 309–327
13. Pooley, D.M., Forsyth, P.A., Vetzal, K.R.: Pricing passport options by PDE methods. University of Waterloo, Working paper (2003)

Improved Monte Carlo Linear Solvers Through Non-diagonal Splitting

Ashok Srinivasan and Vikram Aggarwal

Department of Computer Science, Florida State University,
Tallahassee FL 32306, USA
{asriniva,viaggarw}@cs.fsu.edu

Abstract. Fast, but approximate, solutions to linear algebra problems have many potential applications, such as in graph partitioning, preconditioning, information retrieval, etc. Monte Carlo techniques appear attractive for such needs. While Monte Carlo linear solvers have a long history, their application has been limited due to slow convergence. Despite the development of techniques to improve their accuracy, current methods suffer from the drawback that they are stochastic realizations of inherently poor iterative methods. The reason for such choices is the need for efficient Monte Carlo implementation, which has restricted the splittings that are considered. However, in this paper we demonstrate that such restrictions are not necessarily required, and that efficient Monte Carlo implementations are possible even with splittings that do not appear amenable to it.

1 Introduction

The use of Monte Carlo (MC) in linear algebra dates back to the work of von Neumann and Ulam (described by Forsythe and Leibler [1] in 1950). However, with the development of modern deterministic numerical techniques, MC started losing its appeal in numerical linear algebra. However, there are several applications [2] where approximate solutions would suffice, such as preconditioning, graph partitioning, information retrieval, and feature extraction. These applications are becoming increasingly important, and MC is an ideal technique for determining approximate solutions. Furthermore, parallel MC is very latency tolerant, and so should be effective in a Grid-like environments. MC can also yield specific components of the solution.

The MC technique for solving linear systems consists of generating a random walk, based on the values of the elements of the matrix. The solution is then estimated as a function of the states visited by the walk. Other linear algebra kernels, such as matrix-vector multiplication and eigenvalue determination, can be computed via similar procedures. MC techniques can be used, either as substitutes for conventional deterministic techniques, or to accelerate the convergence of deterministic iterative techniques, for example, through preconditioning.

A major problem with current MC techniques is that they are fundamentally based on stationary iterative methods with poor convergence properties. We

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 168–177, 2003.
© Springer-Verlag Berlin Heidelberg 2003

shall summarize current schemes in Sect. 2 and point out the reason for poor convergence.

We will then explain the basis for our procedure in Sect. 3. Next, we will explain its implementation for dense matrices in Sect. 4, and give experimental results illustrating its convergence properties in Sect. 5. We shall then explain efficient implementation with sparse matrices, in Sect. 6, and then summarize our conclusions in Sect. 7.

2 Current Techniques

2.1 Matrix Vector Multiplication

The basis for MC linear algebra techniques is the ability to perform stochastic matrix-vector multiplication. Based on this ability, stochastic versions of other algorithms can be developed, such as stationary iterative techniques for linear solvers, power method for eigenvalues, etc. We first outline the idea behind stochastic matrix-vector multiplication.

Matrix-Vector Products and Sums of Products Consider the matrix-vector product, $C \cdot h$, $C \in \Re^{n \times n}$ and $h \in \Re^n$. We construct "transition-probability" and "weight" matrices P and W satisfying the following constraint

$$C_{ij} = P_{ij} \times W_{ij}, 1 \leq i, j \leq n \ , \tag{1}$$

where

$$\sum_{i=1}^{n} P_{ij} = 1, 1 \leq j \leq n \ . \tag{2}$$

(The constraint on P, introduced by (2), can be relaxed by introducing absorbing states; however we shall not discuss that.) We similarly define "initial-probability" and "initial-weight" vectors p and w satisfying the following constraint

$$h_i = p_i \times w_i, 1 \leq i \leq n \ , \tag{3}$$

with

$$\sum_{i=1}^{n} p_i = 1 \ . \tag{4}$$

MC techniques estimate $C^j h, \ j \geq 0$ by constructing a Markov chain of length j, with initial probabilities given by the vector p, and transition probabilities by P. The random walk visits a set of states in $\{1, \ldots, n\}$, and we denote the state visited in the i th step by k_i, $i \in [0, j]$. The probability of the initial state being α is given by

$$Prob(k_0 = \alpha) = p_\alpha \ , \tag{5}$$

and the transition probability is given by

$$Prob(k_i = \alpha | k_{i-1} = \beta) = P_{\alpha\beta} \ . \tag{6}$$

Consider random variables X_i defined as follows

$$X_0 = w_{k_0}, \ X_i = X_{i-1} \times W_{k_i k_{i-1}} \ . \tag{7}$$

If we let δ denote the Kronecker delta function ($\delta_{ij} = 1$ if $i = j$, and 0 otherwise), then it can be shown [3] that

$$E(X_j \delta_{ik_j}) = (C^j h)_i, \ 1 \leq i \leq n \ . \tag{8}$$

Therefore, for each random walk, $X_j \delta_{ik_j}$ can be used to estimate the i th component of $C^j h$. That is, in any random walk, the k_j th component is estimated as X_j, and all the other components are estimated as 0. We perform many random walks, maintaining a running sum for estimates of each component, and finally average over the number of walks. Note that in any single walk, the running sum for only one component needs to be updated.

We will later find it useful to estimate $\sum_{j=0}^{m} C^j h$ through

$$E(\sum_{j=0}^{m} W_j \delta_{ik_j}) = (\sum_{j=0}^{m} C^j h)_i \ . \tag{9}$$

Transition Probability Choice: Popular choices to satisfy (1) and (3) are to either (i) make all probabilities in a given column proportional to the magnitude of the corresponding element, or (ii) make all probabilities in each column equal.

Multiple Matrices We can easily extend the above to multiplying by more than one matrix. For example, we will find it useful to estimate $\sum_{j=0}^{m} (BC)^j Bh$, where $B, C \in \Re^{n \times n}$ and $h \in \Re^n$. Transition probabilities and weights for C and h are chosen with the constraints given earlier by (1) and (3). Similarly, transition probabilities and weights for B are chosen such that

$$B_{ij} = \hat{P}_{ij} \times \hat{W}_{ij}, 1 \leq i, j \leq n \ , \tag{10}$$

where $\hat{P}$ is the transition probability matrix and $\hat{W}$ is the weight matrix, with

$$\sum_{i=1}^{n} \hat{P}_{ij} = 1, 1 \leq j \leq n \ . \tag{11}$$

We construct a Markov chain of length $2m + 1$ to estimate $\sum_{j=0}^{m} (BC)^j Bh$, with initial probabilities given by the vector p, and subsequent transition probabilities by $\hat{P}$ and P. The random walk visits a set of states in $\{1, \ldots, n\}$, and we denote the state visited in the i th step by k_i, $i \in [0, 2m+1]$. The probability of the initial state being some α is given by

$$Prob(k_0 = \alpha) = p_\alpha \ , \tag{12}$$

and the transition probabilities by

$$Prob(k_{2i+1} = \alpha | k_{2i} = \beta) = \hat{P}_{\alpha\beta}, 0 \leq i \leq m \ , \tag{13}$$

and

$$Prob(k_{2i} = \alpha | k_{2i-1} = \beta) = P_{\alpha\beta}, \ 1 \leq i \leq m \ . \tag{14}$$

Consider random variables X_i defined as follows

$$X_0 = w_{k_0}, \ X_{2i+1} = X_{2i}\hat{W}_{k_i k_{i-1}}, \ 0 \leq i \leq m, \ X_{2i} = X_{2i-1}W_{k_i k_{i-1}}, \ 1 \leq i \leq m \ . \tag{15}$$

It can be shown, along the same lines as with $\sum_{j=0}^{m} C^j b$, that

$$E(\sum_{j=0}^{m} X_{2j+1}\delta_{ik_{2j+1}}) = (\sum_{j=0}^{m}(BC)^j Bh)_i \ . \tag{16}$$

This enables us to estimate $\sum_{j=0}^{m}(BC)^j Bh$.

2.2 Linear Solvers

In order to solve $Ax = b$, $A \in \Re^{n\times n}$ and $x, b \in \Re^n$, the starting point of MC techniques is to split A as

$$A = N - M \ , \tag{17}$$

and write the fixed-point iteration [4]

$$x^{(m+1)} = N^{-1}Mx^{(m)} + N^{-1}b = Cx^{(m)} + h \ , \tag{18}$$

where $C = N^{-1}M$ and $h = N^{-1}b$. Then we get

$$x^{(m)} = C^m x^{(0)} + \sum_{i=0}^{m-1} C^i h \ . \tag{19}$$

The initial vector $x^{(0)}$ is often taken to be h for convenience, yielding

$$x^{(m)} = \sum_{i=0}^{m} C^i h \ . \tag{20}$$

$x^{(m)}$ converges to the solution as $m \to \infty$ if $\|C\| < 1$.

MC techniques construct a Markov chain to estimate the sum in (20), with the initial probabilities determined by h, and transition probabilities by C, as explained in Sect. 2.1. We wish to note that there are many different estimators available [5,6,1,7,8] (with the one we mentioned being a popular one, and similar to that introduced by Wasow [8]). However, the fundamental idea behind the estimators are essentially the same.

For effectiveness of the MC technique, efficient determination of C is considered important. Therefore current MC techniques choose N to be a diagonal

matrix, thereby yielding $C = N^{-1}M$ efficiently. This yields, for example, the Jacobi method when N is taken to be the diagonal of A[1]. This perceived need for choosing N to be diagonal has resulted in the iterative schemes underlying current MC techniques having poor convergence properties. Though variance reduction, residual correction, and other techniques have been applied on top of this to get better accuracy, the fundamental limitation is that the MC techniques estimate a quantity that itself does not converge fast.

On the other hand, we note that MC techniques do not have to be based on the best possible iterative technique. Estimates can be obtained fast, and this fact may compensate for the underlying iterative scheme having poor convergence properties. Despite this fact, MC techniques have generally not been competitive with deterministic techniques, except for a limited number of applications. Furthermore, the convergence properties of the underlying iterative scheme restrict the systems to which the current MC techniques can be applied. Therefore there is a need for MC techniques based on better underlying iterations.

3 Non-diagonal Splitting

We wish to demonstrate that one can have efficient MC implementations of stationary methods even if we do not restrict ourselves to diagonal splittings. This can yield better convergence properties, at the potential cost of introducing inefficiencies at other points in the algorithm. However, we surmount these problems for our splitting through various means, as explained later. These ideas may also be generalized to other related splittings, which can yield efficient MC techniques for a wider variety of applications.

3.1 The Splitting

We choose N to be the diagonal and first subdiagonal of A. This yields N of the form

$$N = \begin{pmatrix} d_1 & & & & \\ s_2 & d_2 & & & \\ & s_3 & d_3 & & \\ & & & \ddots & \\ & & & & s_n \; d_n \end{pmatrix} . \tag{21}$$

(We assume $d_i \neq 0,\ 1 \leq i \leq n$, as with the Jacobi method. Otherwise N^{-1} does not exist.)

N^{-1} is a lower triangular matrix, and just computing $C = N^{-1}M$ would make this splitting noncompetitive. So we do not explicitly compute C, but rather, estimate the result of the recurrence

$$x^{(m+1)} = N^{-1}Mx^{(m)} + N^{-1}b \ , \tag{22}$$

[1] It is also popular to take N to be the identity matrix.

which yields the following when we take $x^{(0)} = b$.

$$x^{(m)} = \sum_{j=0}^{m} (N^{-1}M)^j N^{-1} b \ . \tag{23}$$

$x^{(m)}$ can be estimated as shown in Sect. 2.1. The number of steps in each random walk will be twice the number in the current techniques. However, if the method converges faster, then we will easily compensate for this.

Before discussing our technique further, we will determine N^{-1}.

3.2 Inverse of N

Each column of N^{-1} can be obtained by solving $Nx = e_i,\ 1 \leq i \leq n$, where e_i is the i th column of the identity matrix. We can then easily verify that N^{-1} is given by

$$N^{-1}_{ij} = \begin{cases} 0 & \text{if } i < j \\ \frac{1}{d_i} & \text{if i = j} \\ \frac{(-1)^{i-j}}{d_j} \Pi^i_{k=j+1} \frac{s_k}{d_k} & \text{otherwise} \end{cases} \tag{24}$$

We note the following two points:

- Any element of N^{-1} can be computed in constant time, using $O(n)$ storage and $O(n)$ pre-computation time as follows. Compute and store $T(i)$[2], $1 \leq i \leq n$, defined as follows

$$T(i) = \begin{cases} 1 & i = 1 \\ T(i-1)\frac{s_i}{d_i} & \text{otherwise} \end{cases} \tag{25}$$

  This takes $O(n)$ space and time.
  Any element of N^{-1} can be computed in constant time as follows

$$N^{-1}_{ij} = \begin{cases} 0 & \text{if } i < j \\ \frac{1}{d_i} & \text{if i = j} \\ \frac{(-1)^{i-j}}{d_j} \frac{T(i)}{T(j)} & \text{otherwise} \end{cases} \tag{26}$$

- The entire matrix N^{-1} can be computed in $O(n^2)$ time, for example, using (25) and (26). This contrasts with the $O(n^3)$ time required for general dense matrices.

We have not characterized the convergence properties of this splitting through a formal analysis, but since it appears to be more general than the Jacobi method, we surmise that it will converge at least for diagonally dominant systems, and empirical results seem to suggest this too. However, further work needs to be done to characterize the convergence properties, at least for matrices arising in specific applications.

[2] If some $s_i = 0$, then we need to make some modifications, as shown in [9].

One may attempt to generalize the current technique, by not choosing the sub-diagonal of A, along with the diagonal, as M, but rather, by choosing large elements of the lower triangle (or upper triangle) from each row of A. This may be advantageous in some applications where a few large entries occur at locations other than the diagonal or sub-diagonal.

Note that the MC technique has two factors that contribute to the error: (i) error from the underlying iterative technique, and (ii) stochastic error in estimating the iterative solution. Much effort has been placed by the MC community in reducing the second factor. These techniques, and also the Sequential MC idea, can be applied to our splitting too. We wish to motivate the search for MC implementations of better iterative processes by demonstrating that one need not be restricted to N being diagonal. Clearly, the splitting we have employed is not in any way as sophisticated as in modern deterministic stationary methods, not to mention non-stationary methods. The important contribution of this work is that we will show how this non-diagonal splitting can be efficiently implemented using MC techniques. These ideas will enable further improvement to MC techniques by permitting more general stationary methods, such as the simple extension mentioned in the previous paragraph.

4 Dense Matrix Implementation

The dense MC implementation of the above method is simple. Since A contains $\theta(n^2)$ elements, we will use $\theta(n^2)$ additional storage and $\theta(n^2)$ pre-computation time to enable efficient simulation. (For sparse systems, we cannot afford $\theta(n^2)$ storage or pre-computation time, and therefore the implementation is more complicated.)

Since M and b are explicitly available, and N^{-1} can be computed and stored in $O(n^2)$ space and time, as shown in Sect. 3.2, we can then estimate the solution using the procedure outlined in Sect. 2.1, taking $C = M$, $B = N^{-1}$ and $h = b$. We chose the probabilities to be proportional to the magnitude of the corresponding element. The alias method [10] is used for sampling, in view of its efficiency – the cost of determining the next state is essentially the cost of producing one random number, independent of the matrix size. This method requires some "information" on the probabilities to be pre-computed. We pre-compute this in an efficient manner, and accomplish it in time proportional to the number of elements. (Once this information is pre-computed, the explicit probabilities are no longer needed.) This requires $O(n^2)$ space and time for dense matrices. The weights are easily computed in $O(n^2)$ space and time [9].

5 Experimental Results

We next report some experimental results. We choose a matrix where the conventional technique will converge fast, in order to demonstrate different issues that affect the relative accuracy of the two techniques. The matrix A is chosen

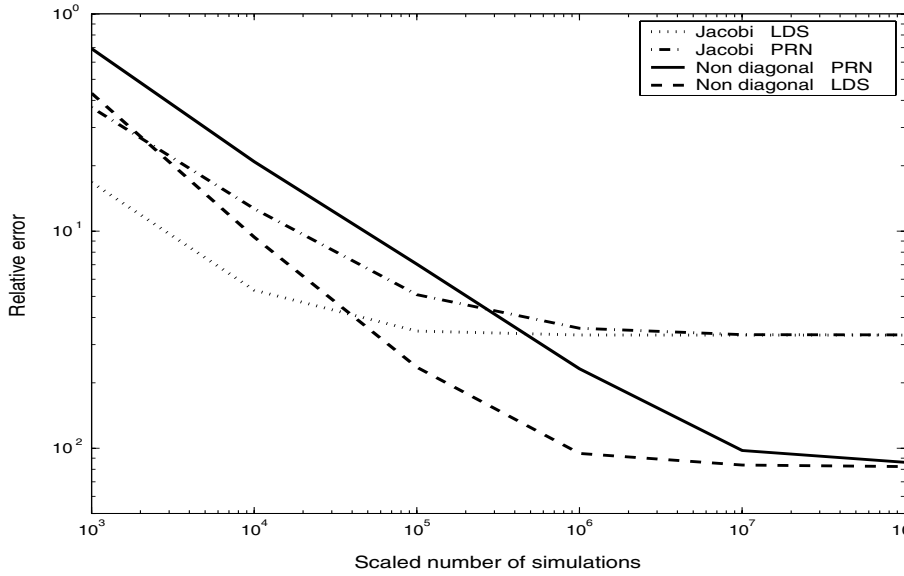


Fig. 1. Plot relative error versus scaled number of simulations. *PRN* refers to a pseudo-random number sequence, while *LDS* refers to a low discrepancy sequence.

to be a 100×100 matrix with $A_{ij} = i - j$ except for the diagonal and sub-diagonal elements. The subdiagonal is taken to be $(2/3)\times$ sum of the absolute values of all other elements of the column, excluding the diagonal. The diagonal is taken to be 1+ twice the sum of the absolute values of all the elements in the column. The vector b was taken to be $(1, 2, \cdots, n)^T$. The matrix A is diagonally dominant, ensuring that the conventional Jacobi-based technique converges. The parameter m of (20) and (23) was taken to be 2, since the iterations converge fast for this matrix. The sub-diagonal elements are moderately large, so that we can see the difference in performance between the two techniques. (At the other extreme, if the sub-diagonal were 0, then the two methods would estimate identical quantities.)

Figure 1 plots relative error ($\|\text{Exact}-\text{Computed solution}\|/\|\text{Exact solution}\|$) versus the "scaled" number of simulations. The scaling refers to the fact that, since our technique takes twice the number of steps that the conventional technique does, we multiply the number of simulations with our technique by a scaling factor of two, in order to make a fair comparison with the conventional one, in terms of time taken. For example, if the simulation number is reported as 1000, it implies 1000 simulations with the conventional technique, but only 500 with ours. The plot has two regions. When the number of simulations is small, the error is dominated by the contribution of the stochastic factor. In this region, the conventional method does better, since the actual number of

simulations is greater for it. When the convergence of the underlying iteration is the dominant factor, then the new technique does better, as expected. Note that techniques that reduce the variance should be added in a real computation, and this will tend to reduce the stochastic error, and so our method would be become better at an earlier point (that is, with fewer simulations). For example, we demonstrate the use of a low discrepancy sequence (Scrambled Halton, with the scrambling due to Faure [11]), which makes our method better at a much earlier point.

6 Sparse Matrix Implementation

We now consider the case where A is sparse. The solution is still estimated using the procedure outlined in Sect. 2.1. Since A is sparse, so is M, and the space required for the weight matrix and the transition probability (implicitly stored as data for the alias method) is proportional to the number of non-zero elements in M. Weight and probability computation for the b vector is as for the dense case, since b is, in general, dense. However, N^{-1} is dense, and we can neither afford $O(n^2)$ computation time to determine it, nor $O(n^2)$ space to store it. We therefore use a sparse representation that enables us to store certain information in $O(n)$ space, with $O(n)$ pre-computation time. Using this, we will be able to generate the next state in constant time, and also compute the weight associated with that transition in constant time.

We now show how we can accomplish the above goals. We assign transition probabilities[3] as follows

$$\hat{P}_{ij} = \begin{cases} 0 & \text{if } i < j \\ \frac{1}{n-j+1} & \text{otherwise} \end{cases} \tag{27}$$

Thus the probabilities for each element above the diagonal is 0, and all elements on and below the diagonal of each column have equal probabilities. Sampling from the uniform distribution can easily be accomplished in constant time, and does not require any additional memory.

The weights need to be chosen to satisfy $N_{ij}^{-1} = P_{ij}\hat{W}_{ij}$ (from (10)). Therefore, from (10) (with $B = N^{-1}$) and (27),

$$\hat{W}_{ij} = \begin{cases} 0 & \text{if } i < j \\ N_{ij}^{-1}(n-j+1) & \text{otherwise} \end{cases} \tag{28}$$

Any element N_{ij}^{-1} can be computed in constant time, as shown in Sect. 3.2, using $O(n)$ space and pre-computation time. Subsequently, $\hat{W}_{ij}$ is computed in constant time each time it is required, as defined by (28).

[3] If some $s_i = 0$, then we need to make some modifications, as shown in [9].

7 Conclusions

We have proposed efficient MC implementations for non-diagonal splitting, and also experimentally demonstrated its effectiveness on dense matrices. We have also proposed an effective sparse implementation, even though the explicit iteration matrix is dense. In fact, we believe that this is the major contribution of this work – demonstrating that efficient sparse implementations are possible even for splittings that lead to a dense C matrix. We expect further studies along these lines to permit stochastic solutions to a wider variety of applications, and have outlined ideas on generalization of the current work.

The sparse and dense implementations are based on the same underlying iterative scheme, and will therefore have identical convergence properties for the portion of the error arising from the iterative process. While the error from the stochastic simulations will converge to 0 with increase in the number of simulations, for both implementations, the actual magnitude of the error could differ significantly between the two implementations, since probabilities and weights are chosen in different manners. This needs to be studied empirically, once the software for the sparse technique is developed.

References

1. Forsythe, G.E., Leibler, R.A.: Matrix inversion by a Monte Carlo method. Mathematical Tables and Other Aids to Computation **4** (1950) 127–127
2. Srinivasan, A., Mascagni, M.: Monte Carlo techniques for estimating the Fiedler vector in graph applications. In: Lecture Notes in Computer Science – 2330, Springer-Verlag (2002) 635–645
3. Dimov, I.T., Karaivanova, A.N.: A power method with Monte Carlo iterations. In Iliev, Kaschiev, Margenov, Sendov, Vassilevski, eds.: Recent Advances in Numerical Methods and Appl. II. World Scientific (1999) 239–247
4. Heath, M.T.: Scientific Computing: An introductory survey. McGraw-Hill, New York (1997)
5. Curtiss, J.H.: Monte Carlo methods for the iteration of linear operators. Journal of Mathematical Physics **32** (1954) 209–232
6. Curtiss, J.H.: A theoretical comparison of the efficiencies of two classical methods and a Monte Carlo method for computing one component of the solution of a set of linear algebraic equations. In: Symposium on Monte Carlo methods, University of Florida, 1954, 191–233, New York, John Wiley and Sons (1956)
7. Halton, J.H.: Sequential Monte Carlo. Proceedings of the Cambridge Philosophical Society **58 part 1** (1962) 57–78
8. Wasow, W.: A note on the inversion of matrices by random walks. Mathematical Tables and Other Aids to Computation **6** (1952) 78–78
9. Srinivasan, A.: Improved Monte Carlo linear solvers through non-diagonal splitting. Technical Report TR-030203, Department of Computer Science, Florida State University (2003) http://www.cs.fsu.edu/research/reports/TR-030203.ps.
10. Knuth, D.E.: The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, Third edition. Addison-Wesley, Reading, Massachusetts (1998)
11. Faure, H.: Good permutations for extreme discrepancy. Journal of Number Theory **42** (1992) 47–56

Valuation of American Options Using Direct, Linear Complementarity-Based Methods

Minas D. Koulisianis and Theodore S. Papatheodorou

High Performance Information Systems Laboratory,
Computer Engineering & Informatics Department,
University of Patras, 26500, Patras, Greece
{mik,tsp}@hpclab.ceid.upatras.gr

Abstract. The Linear Complementarity Formulation of the American Option Valuation problem, which is based on the Black-Scholes partial differential operator, is often used to assist implicit solution methods. Taking advantage of numerical methods, we can obtain approximate solutions, where a "moving index" determines the approximate position of the moving boundary which corresponds to the optimal exercise boundary of the option. Both Direct Inverse Multiplication (DIM) and Stable DIM (SDIM) methods, which are presented in this paper, use the inverse of the coefficient matrix to locate the moving index and then solve a fixed boundary problem. DIM proves to be more than 100 times faster than very popular iterative competitors like Projected Successive OverRelaxation (PSOR). Due to stability problems from which DIM suffers in some cases, a stable version of it, SDIM, has been proposed. Although SDIM is slightly slower than DIM, it is still much faster compared to PSOR.

1 Introduction - Problem Definition

An option is a financial agreement that gives its owner the right to sell (put) or buy (call) an underlying asset S for a prescribed price E (exercise price) at a prescribed date T_{exp} (expiration date) [3,16]. American options have the additional feature that they can be exercised at any time before the expiration date. The calculation of the fair price for which an American option should be offered in the financial markets is called American Option Valuation (AOV) problem and constitutes one of the most interesting problems in computational finance. A mathematical formulation commonly used for this problem, based on the Black-Scholes Partial Differential Equation [1], is the Linear Complementarity Formulation (LCF) [4,5,8,10,12,15]. For this formulation and after applying a set of transformations [15,16] to the original problem [11,15,16], we come to the following transformed implicit Linear Complementarity Problem (LCP):

$$\begin{gathered} u_\tau(x,\tau) - u_{xx}(x,\tau) \geq 0, \qquad u(x,\tau) - g(x,\tau) \geq 0, \\ (u(x,\tau) - g(x,\tau)(u_\tau(x,\tau) - u_{xx}(x,\tau)) = 0 \\ x \times \tau \in (-\infty, +\infty) \times [0,T] \end{gathered} \tag{1}$$

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 178–188, 2003.
© Springer-Verlag Berlin Heidelberg 2003

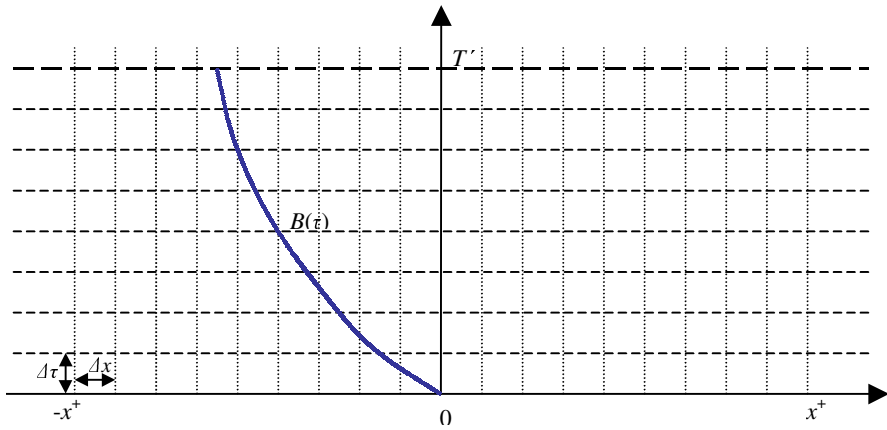


Fig. 1. The Discretization Process for the Transformed AOV Problem.

with

$$g(x,\tau) = e^{\frac{1}{4}((k_2-1)^2+4k_1)\tau} \max(e^{\frac{1}{2}(k_2-1)x} - e^{\frac{1}{2}(k_2+1)x}, 0), \tag{2}$$

$$\tau = 0.5\sigma^2(T-t), \qquad T = 0.5\sigma^2 T_{exp}, \qquad x = \ln\frac{S}{E}, \tag{3}$$

$$k_1 = \frac{r}{0.5\sigma^2}, \qquad k_2 = \frac{r-D}{0.5\sigma^2}, \tag{4}$$

$$P(S,t) = Eu(x,\tau)e^{-\frac{1}{2}(k_2-1)x-\frac{1}{4}((k_2-1)^2+4k_1)\tau}, \tag{5}$$

r being the risk-free interest rate, σ the volatility of the underlying asset, t the current time and P the unknown value of the option. Initial and boundary conditions are:

$$u(x,0) = g(x,0), \qquad x \in (-\infty,\infty), \tag{6}$$

$$\lim_{x\to-\infty} u(x,\tau) = \lim_{x\to-\infty} g(x,\tau) \approx 0, \tag{7}$$

$$\lim_{x\to\infty} u(x,\tau) = 0. \tag{8}$$

In order to solve this problem numerically, we take $x \in [-x^+, x^+]$ for large-enough $x^+ > 0$ and approximate initial and boundary conditions by

$$u(x,0) = g(x,0), \qquad u(-x^+,\tau) = g(-x^+,\tau), \qquad u(x^+,\tau) = 0. \tag{9}$$

For discretization purposes, let $\Delta\tau > 0$, $\Delta x > 0$, $n = N+1 = \frac{2x^+}{\Delta x}$, $TS = \frac{T}{\Delta\tau}$, $x_i = -x^+ + i(\Delta x)$, $i = 0,\ldots,n$, $\tau_j = j(\Delta\tau)$, $j = 0,\ldots,TS$, as shown in Figure 1. We denote by U^j the N-vector whose components $U^j(i)$ approximate $u(x_i,\tau_j)$ and, similarly, let G^j be the N-vector with $G^j(i) = g(x_i,\tau_j)$. Throughout this paper, we have used as a representative discretization method, a Crank-Nicolson

finite difference scheme, which leads, in a straightforward manner, to a sequence of implicit Discrete LCPs (DLCPs) [16], for $j = 1, \ldots, TS$:

$$(AU^j - \beta^j) \geq 0, \;\; (U^j - G^j) \geq 0, \;\; (U^j - G^j)^T (AU^j - \beta^j) = 0, \qquad (10)$$

with

$$\beta^j(i) := U^{j-1}(i+1) + (2(1-\rho)/\rho)U^{j-1}(i) + U^{j-1}(i-1), \qquad U^0 = G^0, \quad (11)$$

A being a tridiagonal matrix:

$$A = trid(-1, 2(1+\rho)/\rho, -1), \qquad \rho = \Delta\tau/(\Delta x)^2. \qquad (12)$$

2 Existence of the Moving Index

In this section we discuss the existence of a moving index in the DLCP defined by Eqs. 10-12. We study this issue for the case of put options (for call options similar results can be obtained). For that case, the range of the underlying asset values $(0, S^\tau_{max})$ for which it is optimal to exercise an option immediately, reduces with time, i.e. $S^{\tau+\Delta\tau}_{max} < S^\tau_{max}$ [15,16]. This happens because the remaining, potential interest on the underlying asset, decreases as we get closer to the expiration date [3,16]. In terms of the transformed discretized problem, this means that as we move upwards in the grid, the moving boundary moves to the left and always separates two regions (see Figure 1); the right region, where it is optimal to hold the option and the left region, where the option should be immediately exercised.

To handle DLCPs more conveniently, we define index sets that contain a series of indices, which correspond to x-nodes of the grid (see Figure 1). Moreover, we define as dimension of an index set C, and denote it as $\|C\|$, the number of elements (indices) that it contains. For every time step j, let index set C^j, consist of a number of consecutive indices that correspond to the unknown coordinates of U^j for the right region (i.e. for $x > B(\tau)$), and denote as $U^j_{C^j}$ the coordinates that correspond to those indices. It follows then that $U^j_{C^j}$ should satisfy the equations of a linear system $A_{C^j} U^j_{C^j} = \beta^j_{C^j}$. This happens because the linear system comes from discretization of $u_\tau(x,\tau) - u_{xx}(x,\tau) = 0$, which is satisfied to the right of the moving boundary. However, trying to verify the above equation to the left of the moving boundary (i.e. for $x < B(\tau)$), would produce coordinate values $U^j(i) < G^j(i)$ for the unknowns that correspond to the left region, since the solution of $u_\tau(x,\tau) - u_{xx}(x,\tau) = 0$ satisfies $u(x,\tau) < g(x,\tau)$ there. On the contrary, setting $U^j_{\overline{C^j}} = G^j_{\overline{C^j}}$ ($\overline{C^j}$ being the complement of C^j with respect to $\{1, 2, \ldots, N\}$), produces coordinate values U^j that satisfy $u_\tau(x,\tau) - u_{xx}(x,\tau) > 0$ in the left region. Using algebraic notation, we can state the following Corollary:

Corollary 1. *Let, k^j be an index that corresponds to the moving boundary at time step j. Then there exist index sets $C^j = \{k^j + 1, \ldots, N\}$ and $\overline{C^j} =$*

$\{1,\ldots,k^j\}$ *such that the solution of the transformed AOV problem can be obtained as a synthesis of two parts, which reside in two neighboring regions:*

$$U^j_{\overline{C^j}} = G^j_{\overline{C^j}} \text{ in the Left Region (LR), and} \tag{13}$$

$$A_{C^j} U^j_{C^j} = \beta^j_{C^j} \text{ in the Right Region (RR).} \tag{14}$$

Although, in the LCF, there is not explicit reference to the moving boundary, the techniques presented in this paper "simulate" its behavior. The proposed methods are based on the fact that whatever happens with the moving boundary, has to be reflected on the corresponding discrete problem. Index k^j, which separates LR from RR within each time step, represents the moving boundary and is thus called Moving Index.[1] Positioning of k^j is the main objective of the methods presented in this paper. The following Lemma holds:

Lemma 1. *At time step j, let $C^j_{ar} = \{k^j_{ar}+1,\ldots,N\}$, $\overline{C^j_{ar}} = \{1,\ldots,k^j_{ar}\}$ be two index sets for an arbitrarily chosen k^j_{ar}. Then the following possibilities exist:*
Case 1: If $k^j_{ar} < k^j$, then $C^j_{ar} \supset C^j$ and $\overline{C^j_{ar}} \subset \overline{C^j}$. Setting $U^j_{\overline{C^j_{ar}}} = G^j_{\overline{C^j_{ar}}}$ and solving the subsystem $A_{C^j_{ar}} U^j_{C^j_{ar}} = \beta^j_{C^j_{ar}}$, leads to violations of the type $U^j(i) < G^j(i)$ for a number of consecutive equations, i.e. $i = k^j_{ar}+1,\ldots,k^j$.
Case 2: If $k^j_{ar} > k^j$, then $C^j_{ar} \subset C^j$ and $\overline{C^j_{ar}} \supset \overline{C^j}$. Setting $U^j_{\overline{C^j_{ar}}} = G^j_{\overline{C^j_{ar}}}$ and solving the subsystem $A_{C^j_{ar}} U^j_{C^j_{ar}} = \beta^j_{C^j_{ar}}$, leads to violation of $A_{\overline{C^j_{ar}}} U^j_{C^j_{ar}} \geq \beta^j_{\overline{C^j_{ar}}}$.
Case 3: If $k^j_{ar} = k^j$, then $C^j_{ar} = C^j$ and $\overline{C^j_{ar}} = \overline{C^j}$. Setting $U^j_{\overline{C^j_{ar}}} = G^j_{\overline{C^j_{ar}}}$ and solving the subsystem $A_{C^j_{ar}} U^j_{C^j_{ar}} = \beta^j_{C^j_{ar}}$, leads to the solution of the DLCP.

Proof. If $k^j_{ar} < k^j$ holds (Case 1), then we attempt to satisfy equation $u_\tau(x,\tau) - u_{xx}(x,\tau) = 0$ in LR. This, as explained earlier, produces coordinates $U^j(i) < G^j(i)$ and thus leads to a number of constraint violations. If $k^j_{ar} > k^j$ holds (Case 2) then, for some values of x, we accept that $u_\tau(x,\tau) - u_{xx}(x,\tau) > 0$ holds, instead of $u_\tau(x,\tau) - u_{xx}(x,\tau) = 0$. As all conditions of 10 must hold, this definitely leads to a violation of inequality $A_{\overline{C^j_{ar}}} U^j_{\overline{C^j_{ar}}} \geq \beta^j_{\overline{C^j_{ar}}}$. Due to existence and uniqueness of solution [4,6,9,11,16] there is only one k^j, that satisfies Eqs. 13-14. Thus, the solution of the problem can be calculated only if Case 3 holds.

3 Direct Inverse Multiplication (DIM) and Stable DIM

Both DIM and SDIM can be considered direct methods as they provide the exact solution of the algebraic problem without executing successive iterations. For this kind of methods the determination of the moving index is necessary. Both of them first track the moving index and then solve a linear system. Moreover, they exploit information obtained in a time step to determine the moving index

[1] More accurately, the moving boundary resides between grid points k^j and k^j+1.

of the next time step. In order to locate the moving index, the inverse of the coefficient matrix is used. Although calculating an inverse is generally inefficient and expensive, there are cases where such a calculation is direct, easy and worthwhile. The discretization method plays significant role. More specifically, for the case of Crank-Nicolson, finite difference discretization, the resulting coefficient matrix is very simple (see Eq.12), and presents a number of desirable properties (tridiagonal, toeplitz, symmetric, diagonally dominant). For such a matrix, calculation of the inverse is straightforward and can be done very fast [14].

DIM and SDIM methods make use of Lemma 1. At each time step, j, they try to calculate k^j by solving successively subsystems of equations for successive values of k^j_{ar} (starting with $k^j_{ar} = k^{j-1}$). An initial guess, k^0_{ar}, that falls into Case 2 or Case 3 of Lemma 1 is necessary as a starting point. The exact location of the moving boundary at time 0 resides on $x = 0$, in case of absence of dividends, or on $x = \min(0, \ln(r/D)) \leq 0$ if the option pays dividends [15]. Thus, a selection $k^0_{ar} = N/2$, makes sure that at time step 1 we start with $k^1_{ar} (= k^0_{ar}) \geq k^1$, and *Case 2* or *Case 3* of Lemma 1 holds. After every calculation of $U^j_{C^j_{ar}}$, if there are not violations of the type $U^j(i) < G^j(i)$, for some i, then C^j_{ar} resides completely in RR of Corollary 1 (*Case 2* or *3* of Lemma 1) and thus we have to augment C^j_{ar}, adding index $k^j_{ar} - 1$, and calculate another $U^j_{C^j_{ar}}$. Expanding C^j_{ar} and calculating $U^j_{C^j_{ar}}$ successively, for some k^j_{ar}, a violation of the type $U^j(k^j_{ar}) < G^j(k^j_{ar})$ will take place. When this happens, we have just passed from *Case 3* to *Case 1* of Lemma 1 and thus $C^j_{ar} - \{k^j_{ar}\} = C^j$ and $k^j_{ar} = k^j$.

We should emphasize that, successive values $U^j(k^j_{ar})$ can be calculated by multiplying the first row of the inverse of the appropriate dimension with the appropriate right hand side subvector. Both DIM (Algorithm 3) and SDIM (Algorithm 4) calculate the first line of $A^{-1}_{H^j_{ar}}$ using a method described in [14] (H^j_{ar} being an index set defined as $H^j_{ar} = \{k^j_{ar}\} + C^j_{ar}$). The whole process is presented in Algorithm 1:

Algorithm 1

Do $j = 1, M$
Step 0: Initialization. Set $k^j_{ar} = k^{j-1}_{ar} (k^0_{ar} = \frac{N}{2})$
Step 1: Set $C^j_{ar} = \{k^j_{ar} + 1, \ldots, N\}$
Step 2: Calculate $U^j(k^j_{ar})$, using Algorithm 3 or 4 (see Sections 3.1, 3.2)
Step 3: *If* $U^j(k^j_{ar}) \geq G^j(k^j_{ar})$ *then* $C^j = C^j_{ar}, k^j = k^j_{ar}, U^j_{C^j} = G^j_{C^j}$
Solve the subsystem $A_{C^j} U^j_{C^j} = \beta^j_{C^j}$
else $C^j_{ar} = \{C^j_{ar}\} + \{k^j_{ar}\}$, $k^j_{ar} = k^j_{ar} - 1$, return to *Step 1*
End If
End Do

3.1 DIM

According to [14] the inverse of a matrix $A_N = trid(-1, \alpha, -1)$(equivalent to the coefficient matrix of Eq.12) can be calculated as follows:

$$A_N^{-1}(p,i) = \frac{1}{R(1)R(N+1)} R(p)R(N+1-i), \qquad i,p = 1,\ldots,N \tag{15}$$

where

$$R(m) = r_+^m - r_-^m, \qquad m = 1,\ldots,N+1 \tag{16}$$

and

$$r_\pm = \frac{1}{2}(a \pm \sqrt{a^2-4}). \tag{17}$$

Consequently, the first row of $A_{H_{ar}^j}^{-1}$ is given by:

$$A_{H_{ar}^j}^{-1}(1,i) = \frac{1}{R(\|H_{ar}^j\|+1)} R(\|H_{ar}^j\|+1-i), \qquad i = 1,\ldots,\|H_{ar}^j\| \tag{18}$$

Algorithm 2

Step 0: Initialization. $r_\pm = \frac{1}{2}(a \pm \sqrt{a^2-4})$
Do $p = 1, \|H_{ar}^j\|$
$R(p) = r_+^p - r_-^p$
End Do
Step 1: Do $i = 1, \|H_{ar}^j\|$
$A_{H_{ar}^j}^{-1}(1,i) = \frac{1}{R(\|H_{ar}^j\|+1)} R(\|H_{ar}^j\|+1-i)$
End Do
Step 2: Calculate $U^j(k_{ar}^j) = A_{H_{ar}^j}^{-1}(1,.)\beta_{H_{ar}^j}^j$

Algorithm 2 calculates $U^j(k_{ar}^j)$ within each time step. Given that it is applied successively until k^j has been determined, a basic deficiency of this calculation is that it does not take advantage of previous computational effort spent, in order to accelerate subsequent computational steps. The reason is that each time we expand H_{ar}^j, we conclude to an inverse of larger dimension, every element of which is now different. Only calculations done in the initialization step (*Step 0*) remain the same in the new problem. The following technique optimizes Algorithm 2 recomposing calculations in such a way that reuse of previous work done becomes possible: let $A_N^{-1}(1,.)$ be the first line of the inverse with the maximum possible dimension N (calculated by substituting $p = 1$ to Eq.15):

$$A_N^{-1}(1,i) = \frac{R(N+1-i)}{R(N+1)}, \qquad i = 1,\ldots,N, \tag{19}$$

and construct a vector θ such that

$$\theta(i) = \frac{R(N+1)}{R(N+1-i)} = \frac{1}{A_N^{-1}(1,i)}, \qquad i = 1,\ldots,N. \tag{20}$$

It is obvious now that every $U^j(k^j_{ar})$ can be constructed, taking advantage of calculations 19 and 20 by multiplying $\lambda = A_N^{-1}(1,.)\beta^j_{H^j_{ar}}$ by $\theta(k^j_{ar})$. Whenever $\|H^j_{ar}\|$ is increased by one, we add $A_N^{-1}(1,k^j_{ar}-1)\beta^j(k^j_{ar}-1)$ to λ and multiply the result by $\theta(k^j_{ar}-1)$. Of course, special care must be taken for the appropriate corrections that should be applied due to boundary conditions, to the first element of the right hand side at each stage of the process. Algorithm 3 presents the process just stated.

Algorithm 3

Step 0: Initialization. $r_{\pm} = \frac{1}{2}(a \pm \sqrt{a^2-4})$
Do $p = 1, N$
$R(p) = r_+^p - r_-^p$
End Do
Step 1: $A_N^{-1}(1,i) = \frac{R(N+1-i)}{R(N+1)}, \quad i = 1,\ldots,N, \qquad \theta(i) = \frac{R(N+1)}{R(N+1-i)}, \quad i = 0,\ldots,N-1$
Step 2: $\lambda = A_N^{-1}(1,k^j_{ar}:N)\beta^T_{H^j_{ar}}$, $U^j(k^j_{ar}) = \theta(k^j_{ar})\lambda$
Do until $k^j_{ar} = k^j$
$\lambda = \lambda + A_N^{-1}(1,k^j_{ar})\beta^j(k^j_{ar})$
$U^j(k^j_{ar}) = \theta(k^j_{ar})\lambda$
End Do

3.2 SDIM

Although $A^{-1}_{H^j_{ar}}(1,.)$ may contain very "reasonable" numbers, calculation of $R(m)$ in Eq.16 is possible to cause overflows or underflows for very large values of m. Fortunately, we can handle overflows and underflows, by applying a technique that leads to a variation of DIM, SDIM. To check if a quantity q^l underflows, we must take into account the smallest positive number that can be represented under the precision of the arithmetic used, which is equal to 2^{-1022} (2.2251×10^{-308}) in the current work (double precision). It is then sufficient to check if

$$-\frac{1022}{\log_2(q)} \geq l. \tag{21}$$

Calculation 21 is easy and safe, and in case that its result is true, an underflow would undoubtedly take place if attempting calculation q^l.

Setting $\gamma = \frac{r_-}{r_+}$, it is obvious that $\gamma \in (0,1)$ and we can rewrite (18) as

$$A^{-1}_{H^j_{ar}}(1,i) = r_+^{-i}\frac{1-\gamma^{\|H^j_{ar}\|+1-i}}{1-\gamma^{\|H^j_{ar}\|+1}}, \qquad i = 1,\ldots,\|H^j_{ar}\|. \tag{22}$$

Calculation 22, is not possible to overflow but, depending on the value of γ and its exponent, it is possible to underflow. However, for very large values of $\|H^j_{ar}\|$

$$\gamma^{\|H^j_{ar}\|+1} \to 1. \tag{23}$$

Thus, we may set $A^{-1}_{H^j_{ar}}(1,i) = r_+^{-i}(1-\gamma^{\|H^j_{ar}\|+1-i})$, in case that $\gamma^{\|H^j_{ar}\|+1}$ would produce an underflow, provided that calculation of $r_+^{-i}(1-\gamma^{\|H^j_{ar}\|+1-i})$ itself would not produce an underflow. In case that $r_+^{-i}(1-\gamma^{\|H^j_{ar}\|+1-i})$ would underflow for some values of i, we set $A^{-1}_{H^j_{ar}} = 0$ for those values. The technique presented above, can not be embedded in Algorithm 3 because quantities $\theta(i)$, may overflow and we cannot handle overflows. The Algorithm follows:

Algorithm 4

Step 0: Initialization. $r_\pm = \frac{1}{2}(a \pm \sqrt{a^2-4}), \gamma = \frac{r_-}{r_+}$

Step 1: *Do* $i = 1, \|H^j_{ar}\|$

If calculation of $\gamma^{\|H^j_{ar}\|+1}$ is stable *then* $A^{-1}_{H^j_{ar}}(1,i) = r_+^{-i}\frac{1-\gamma^{\|H^j_{ar}\|+1-i}}{1-\gamma^{\|H^j_{ar}\|+1}}$

else if calculation of $r_+^{-i}(1-\gamma^{\|H^j_{ar}\|+1-i})$ is stable *then* $A^{-1}_{H^j_{ar}}(1,i) = r_+^{-i}(1-\gamma^{\|H^j_{ar}\|+1-i})$

else $A^{-1}_{H^j_{ar}}(1,i) = 0$

End If

End Do

Step 2: Calculate $U^j(k^j_{ar}) = A^{-1}_{H^j_{ar}}(1,.)\beta^j_{H^j_{ar}}$

4 Evaluation

In order to make the evaluation as objective as possible, we have used realistic data, withdrawn from ranges of possible values given in [2]. A representative test problem has been used with $E = 10$, $\sigma = 15\%$, $r = 10\%$, $D = 0$ and $T_{exp} = 4/12$. Based on the transformed LCP defined by Eqs. 8-10, we calculate a solution vector U, from which put option values for any $S \in (l, L)$ ($l \to 0$ and $L \to \infty$) can be obtained. The current evaluation is based on accuracy. For direct methods, accuracy depends only on the grid density. Thus, for a given grid size, DIM and SDIM present identical accuracy, i.e. the maximum possible accuracy determined by the precision of the arithmetic used. For our experiments, we have used double precision for which the maximum countable error in arithmetic calculations is 2^{-52} (2.22×10^{-16}). For the Projected Successive OverRelaxation (PSOR) method [7,16], setting as termination error $\epsilon = 2.22 \times 10^{-16}$, equivalent precision to that of DIM and SDIM is achieved. Thus, direct comparison of execution times for all three methods is objective and meaningful. All methods have been implemented using Fortran 90, optimized and executed on a PIII/500 P/C with 256MB RAM. Moreover, they are available through a Problem Solving Environment developed under the world wide web [13].

In Tables 1, 2 and 3, we quote execution times achieved by PSOR, DIM and SDIM respectively for various values of N (number of x-nodes) and TS (number of time steps). The first thing to notice about DIM and SDIM is that, doubling N or TS roughly doubles execution times. This can be easily explained: doubling N leads to the solution of problems of double size, while doubling TS doubles the

Table 1. Execution times (in seconds) for the PSOR method, with $\epsilon = 2.22 \times 10^{-16}$. (*: more than 1000 seconds)

TS	**N**							
	128	**256**	**512**	**1024**	**2048**	**4096**	**8192**	**16384**
128	0.128	0.195	0.500	1.844	7.265	31.616	166.704	*
256	0.163	0.336	0.821	2.632	10.312	44.171	192.938	*
512	0.289	0.601	1.399	4.024	13.241	58.428	260.357	*
1024	0.539	1.109	2.421	6.617	19.327	71.106	332.299	*
2048	1.016	2.055	4.422	11.133	30.163	98.183	394.540	*
4096	2.015	4.094	8.414	19.804	49.365	153.885	546.151	*
8192	3.820	7.984	16.249	35.710	83.606	373.775	786.501	*
16384	7.570	15.554	31.913	68.435	149.550	775.846	*	*

Table 2. Execution times (in seconds) for the DIM method. (**: calculation fails)

TS	**N**							
	128	**256**	**512**	**1024**	**2048**	**4096**	**8192**	**16384**
128	0.078	0.107	0.187	0.329	0.648	1.266	2.547	5.249
256	0.102	0.172	0.322	0.625	1.250	2.492	4.964	10.398
512	0.179	0.329	0.625	1.238	2.473	4.937	10.001	20.324
1024	0.336	0.626	1.246	2.439	**	**	**	**
2048	0.648	1.242	**	**	**	**	**	**
4096	1.227	2.443	**	**	**	**	**	**
8192	2.436	**	**	**	**	**	**	**
16384	4.835	**	**	**	**	**	**	**

number of problems. For small values of N (e.g. $N = 128$), DIM performs slightly better than PSOR (less than 2 times faster). However, for large values of N, DIM becomes extremely faster than PSOR, e.g. about 65 times faster for $N = 8192$, $TS = 128$ and more than 190 times faster for $N = 16384$, $TS = 128$. In general, the extent of improvement of DIM over PSOR decreases as TS increases. The reason is that PSOR converges faster for larger values of TS. However, DIM presents instability problems and cannot be used with all values of N and TS (** is entered for these cases in Table 2). Although SDIM is slightly slower than DIM, it still consists an extremely efficient alternative of iterative methods. For example, for $N = 8192$, $TS = 128$, SDIM is about 2 times slower that DIM, but it is about 35 times faster than PSOR. Moreover, there are cases where DIM can not be applied, while SDIM works properly and clearly outperforms PSOR.

5 Conclusions

In this paper we have presented two direct, linear complementarity-based methods for the solution of the American Option Valuation problem. A "moving index" determines the approximate position of the optimal exercise boundary of the option and gives rise to techniques that try to locate it. Calculation of

Table 3. Execution times (in seconds) for the SDIM method.

TS	**N**							
	128	**256**	**512**	**1024**	**2048**	**4096**	**8192**	**16384**
128	0.099	0.120	0.216	0.430	0.922	2.016	4.836	12.836
256	0.108	0.196	0.406	0.828	1.711	3.633	8.039	19.468
512	0.203	0.398	0.790	1.602	3.250	6.742	14.257	32.139
1024	0.399	0.782	1.539	3.062	6.195	12.664	26.132	56.412
2048	0.805	1.547	3.016	5.938	11.844	24.007	49.092	104.059
4096	1.563	3.047	5.867	11.452	22.781	46.217	93.590	195.337
8192	3.094	6.000	11.461	22.359	44.311	89.536	179.876	377.845
16384	6.188	11.804	22.562	44.038	87.114	177.509	351.112	734.589

this index plays significant role for both DIM and SDIM methods. Both of them use the inverse of the coefficient matrix, which is the outcome of a discretization process, to locate the moving index, and then solve a fixed boundary problem. DIM proves to be up to more than 100 times faster than very popular iterative competitors like PSOR. However, due to stability problems from which DIM suffers in some cases, a stable version of it, SDIM, has been proposed. Although SDIM is slightly slower than DIM, it is still much faster compared to PSOR. Moreover, SDIM works properly even in cases that DIM fails.

References

1. Black, F., Scholes, M.: The Pricing of Options and Corporate Liabilities. Journal of Political Economy **81** (1973) 637–654
2. Broadie, M., Detemple, J.: Recent Advances in Numerical Methods for Pricing Derivative Securities. Numerical Methods in Finance, Cambridge Univ. Press (1997)
3. Clewlow, L., Strickland, C.: Implementing Derivatives Models. Wiley (1998)
4. Cottle, R., Pang, J., Stone, R.: The Linear Complementarity Problem. Academic Press (1993)
5. Cottle, R., Sacher, R.: On the solution of large, structured linear complementarity problems: the tridiagonal case. Appl. Math. Optimization **3(4)** (1976-7) 321-341
6. Crank, J.: Free and moving boundary problems. Clarendon Press, (1984)
7. Cryer, C.: The solution of a quadratic programming problem using systematic overrelaxation. SIAM J. Control **9** (1971) 385–392
8. Cryer, C.: The efficient solution of linear complementarity problems for tridiagonal Minkowski matrices. ACM Trans. Math. Software **9(2)** (1983)
9. Elliot, C., Ockendon, J.: Weak and Variational Methods for Moving Boundary Problems, Pitman (1982)
10. Huang J., Pang, J.: Option Pricing and Linear Complementarity. Journal of Finance **2(3)** (1998)
11. Koulisianis, M., Papatheodorou, T.: A 'Moving Index' method for the solution of the American options valuation problem. Mathematics and Computers in Simulation, Elsevier Science **54(4-5)** (2000)
12. Koulisianis, M., Papatheodorou, T.: Improving Projected Successive OverRelaxation Method for Linear Complementarity Problems. 5th IMACS Conference on Iterative Methods in Scientific Computing, Greece (2001) (to appear in APNUM)

13. Koulisianis, M., Tsolis, G., Papatheodorou, T.: A Web-Based Problem Solving Environment for Solution of Option Pricing Problems and Comparison of Methods. ICCS 2002, LNCS 2329 (2002) 673-682
14. Meraunt, G.: A Review on the Inverse of Symmetric Tridiagonal and Block Tridiagonal Matrices. SIAM J. Matrix Anal. Appl., **13(3)** (1992) 707-728
15. Pantazopoulos, K.: Numerical Methods and Software for the Pricing of American Financial Derivatives. phD Thesis, Purdue University (1998)
16. Willmott, P., Dewynne, J., Howison, S.: Option Pricing, Mathematical Models and Computation. Oxford Financial Press (1993)

Testing the Expectations Hypothesis for Interest Rate Term Structure: Some Australian Evidence

Victor Fang [1] and Vincent C.S. Lee [2]

[1] Department of Accounting and Finance, Faculty of Business and Economics, Monash University,
victor.fang@buseco.monash.edu.au

[2] School of Business Systems, Faculty of Information Technology, Monash University,
vincent.lee@infotech.monash.edu.au

Corresponding author: Dr Vincent C S Lee, http://www.bsys.Monash.edu.au

Abstract: Many test results are found inconsistent with the expectations hypothesis of the term structure. The aim of this paper is to re-examine the expectations hypothesis of the term structure using the Australian interest rate data from 1969(7) to 1995(7). We start with the cointegration test on R_t, r_t, and S_t followed by the Granger causality test from S_t to ∇r_t. Finally we carry out the VAR model of cross-equation restrictions test. Our findings show that there is no conclusive rejection of the expectations hypothesis of the term structure.

1 Introduction

A large number of empirical tests on the expectations hypothesis of interest rate term structure have been carried out jointly with the hypothesis of rational expectations for real wage/prices in the context of macroeconomic context during the recent times. But most of the joint hypothesis test results are found inconsistent with the expectations hypothesis of the interest rate term structure. For example, Shiller (1979), Sargent (1979) and Campbell and Shiller(1987), in all their papers, they were all 'overwhelmingly' rejected the expectations hypothesis of the term structure. In contrary to their results, MacDonald and Speight (1988) using the techniques developed by Campbell and Shiller (1987) to examine UK quarterly data from 1963 to 1987 found evidence consistent with the expectations hypothesis. Mills (1991) also uses the UK interest rates data from 1870 to 1988 to examine the expectations hypothesis of term structure. His results are somewhat different. He found that the hypothesis is rejected for data up to the Pre-world War. However, there is little support for the rejection of the hypothesis for the data from the 1980's.

The aim of this paper is to examine the expectations hypothesis of the term structure using the Australian interest rate data from 1969(7) to 1995(7).

The structure of the paper is as follows: Section II briefly sets out the expectations model and develops a testable implication. Section III discusses the data

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 189-198, 2003.
© Springer-Verlag Berlin Heidelberg 2003

used and Section IV reports the empirical results and the paper closes with a concluding section.

2 The Expectations Hypothesis of Interest Rate Term Structure

The expectations hypothesis states that the slope of the yield curve may be explained by investors' expectations of future short-term interest rates. An upward sloping yield curve implies that investors expect future spot rates to be higher than the current spot rates; a downward sloping yield curve implies that investors expect future spot rates to fall below current levels.

From Shiller (1979), this expectations hypothesis can be expressed mathematically as follows:

$$R_t^n = \frac{1-\gamma}{1-\gamma^n}\sum_{i=0}^{n-1}\gamma^i E_t(r_{t+i}) + \theta^n \qquad (1)$$

where $0 \prec \gamma \prec 1$ and θ^n is a constant liquidity premium. R_t^n denotes the yield to maturity on an n-period bond, E_t is the mathematical expectations operator conditional on publicly available information at time t, r_{t+i} denotes the current expectation of the one period rate i periods ahead.

From equation (1), Campbell and Shiller (1987,1988) then further showed that when a bond is a perpetuity, the spread i.e. the difference between the long rate and the short rate can be expressed as:

$$S_t^\infty = \sum_{i=1}^{\infty}\gamma^i E_t(\nabla r_{t+i}) + \theta^\infty \qquad (2)$$

They argued that if R_t^∞ and r_t are both integrated of order one and their first differences (∇R_t^∞, ∇r_t) and S_t^∞ are I(0) series i.e. they are stationary. It follows that R_t^∞ and r_t must be co-integrated with the co-integration vector (1,-1).

Mills (1991) pointed out that since S_t^∞ and ∇r_t are stationary, i.e. I(0) series, then there should exist a bi-variate *Wold representation*[*] which can be approximated by the vector autoregression (VAR) of appropriate lags:

$$\begin{aligned} S_t^\infty &= \sum_{i=1}^{j} a_i S_{t-i}^\infty + \sum_{i=1}^{j} b_i \nabla r_{t-i} + \varepsilon_t \\ \nabla r_t &= \sum_{i=1}^{j} c_i S_{t-i}^\infty + \sum_{i=1}^{j} e_i \nabla r_{t-i} + \eta_t \end{aligned} \qquad (3)$$

where a_i, b_i, c_i, e_i are restrictions.

We can then transform equation (3) into matrix form as:

*Wold decomposition implies that the bivariate process has a unique, infinite order moving average representation.

$$\begin{bmatrix} S_t^{\infty} \\ \\ \\ S_{t-j+1}^{\infty} \\ \nabla r_t \\ \\ \\ \nabla r_{t-j+1} \end{bmatrix} = \begin{bmatrix} a_1 \cdots\cdots a_j & b_1 \cdots\cdots b_j \\ 1 \cdots\cdots\cdots 0 & 0 \cdots\cdots\cdots 0 \\ \\ 0 \cdots\cdots\cdots 1 & 0 \cdots\cdots\cdots 0 \\ c_1 \cdots\cdots c_j & e_1 \cdots\cdots e_j \\ 0 \cdots\cdots\cdots 0 & 1 \cdots\cdots\cdots 0 \\ \\ 0 \cdots\cdots\cdots 0 & 0 \cdots\cdots\cdots 1 \end{bmatrix} \begin{bmatrix} S_{t-1}^{\infty} \\ \\ \\ S_{t-j}^{\infty} \\ \nabla r_{t-1} \\ \\ \\ \nabla r_{t-j} \end{bmatrix} + \begin{bmatrix} \varepsilon_t \\ 0 \\ \\ 0 \\ \eta_t \\ 0 \\ \\ 0 \end{bmatrix} \tag{4}$$

which can be written more compactly as:

$$Z_t = AZ_{t-1} + V_t \tag{5}$$

From this, we can develop a set of cross equation restrictions on the VAR:

$$\begin{aligned} & b_i + e_i = 0 \ldots\ldots; where \ldots\ldots\ldots i = 1, \ldots\ldots\ldots j \\ & a_1 + c_1 = \gamma^{-1} \\ & a_i + c_i = 0 \ldots\ldots\ldots; where \ldots\ldots\ldots i = 2, \ldots\ldots\ldots j \end{aligned} \tag{6}$$

From equation (4), we have two equations,

$$\begin{aligned} S_t^{\infty} &= a_1 S_{t-1}^{\infty} + \cdots\cdots a_j S_{t-j}^{\infty} + b_1 \nabla r_{t-1} + \cdots\cdots b_j \nabla r_{t-j} + \varepsilon_t \\ \nabla r_t &= c_1 S_{t-1}^{\infty} + \cdots\cdots c_j S_{t-j}^{\infty} + e_1 \nabla r_{t-j} + e_j \nabla r_{t-j} + \eta_t \end{aligned}$$

By adding together the two equations, it becomes

$$S_t^{\infty} + \nabla r_t = \sum_{i=1}^{j} (a_i + c_i) S_{t-i}^{\infty} + \sum_{i=1}^{j} (b_i + e_i) \nabla r_{t-i} + \varepsilon_t + \eta_t$$

and imposing the restrictions equation(6), the following equation is obtained:

$$\nabla r_{t+1} + S_{t+1}^{\infty} - \gamma^{-1} S_t^{\infty} = \varepsilon_t + \eta_t = \mu_t \tag{7}$$

This equation shows that ($\nabla r_{t+1} + S_{t+1}^{\infty} - \gamma^{-1} S_t^{\infty}$) is unpredictable given the past values of ∇r_t and S_t^{∞}. Therefore, by assuming $\nabla R_{t_t}^n$, ∇r_t and S_t^{∞} are stationary i.e. *I(0)* series, we can use Wald test statistic to test the restrictions in equation (6) implied by the expectations hypothesis of the term structure.

From the VAR model, Campbell and Shiller (1987) show that the expectations hypothesis implies that the spread S_t^{∞} must linearly Granger-cause the changes in the short rate ∇r_t .The intuition of this implication is that if agents have information other than simply the past values of r , then S_t^{∞} will have additional explanatory power for forecasting future ∇r_t given that S_t^{∞} is an optimal forecast of future values of ∇r_t . Therefore, we can test for Granger causality from S_t^{∞} to ∇r_t for appropriate lag length implied by the expectations hypothesis.

3 Data

The data used in this study are end of month observations on 90-day bank accepted bill rate and the 10-year Treasury bond rate over the period 1969(7) to 1995(7) We need to consider the data set before the period of deregulation i.e. before mid 1982's and after the period of deregulation. The Reserve Bank of Australia (RBA) controlled the Australian Interest Rates prior to the deregulation of Australian currency. The RBA set the ceiling rates for the deposit rates and lending rates. No banks were allowed to take in deposits or giving out loans higher than the ceiling rates set by the RBA. The yields on the Treasury notes and Treasury bonds were also fixed by the RBA. Since the liberalisation of the financial markets in mid 1982's, the RBA lifted the ceiling rates and allowed the market to determine the interest rates. The RBA also introduced the tender system for issuing the T-bond and T-notes, which would reflect the genera liquidity conditions in the market. Because of the interest rate restrictions during the Pre-deregulated period and the free market conditions after the deregulations, it is necessary to sub-divide the data set into Pre-deregulated period i.e. from 1969(7) to 1982(6) and Post-deregulated period i.e. from 1982(7) to 1995(7).

Plots of 10 year T-bonds (denoted by '*R*') and 90 day Bank bill rates (denoted by '*r*') and the associated spreads (*R* - *r*) for various periods are presented as Figure 3.1.

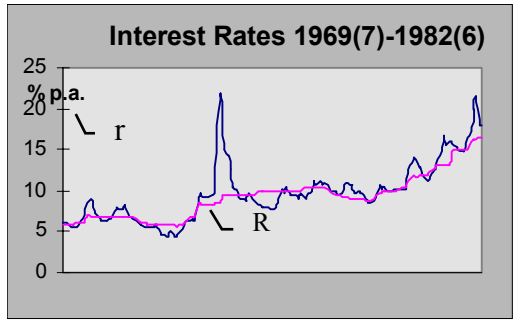


Figure 3.1 (a) Australian Interest Rates -1969(7)-1982(6)

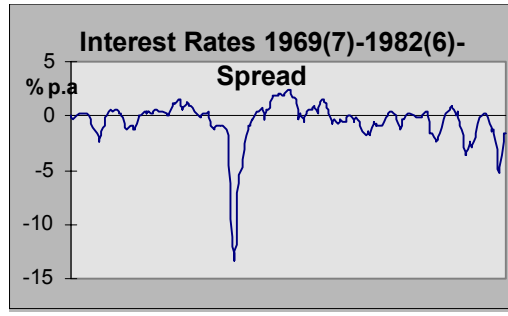


Figure 3.1 (b) Australian Interest Rates Spread -1969(7)-1982(6)

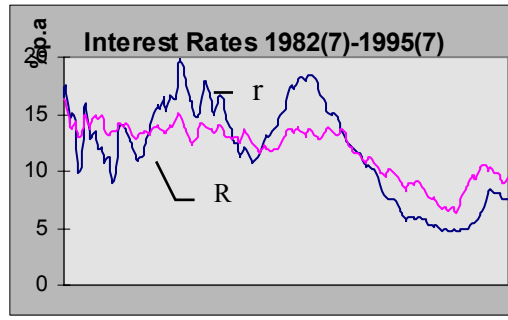


Figure 3.1 (c) Australian Interest Rates -1982(7)-1995(7)

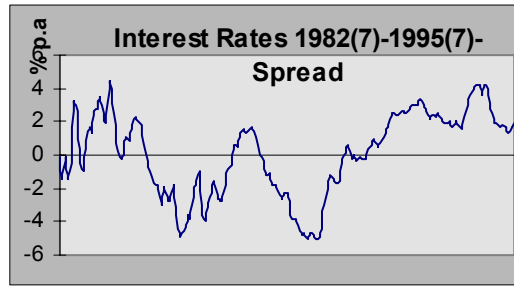


Figure 3.1(b) Australian Interest Rates Spread -1982(7)-1995(7)

4 Empirical Results

Three different empirical tests (Co-integration, Granger Causality, and VAR) were implemented in this paper. The respective test results are reported as follows.

4.1 Co-integration Results

As we have noted that the model of the term structure showed in equation (2) implies that long rate R_t^{∞} and short rate r_t should be co-integrated. This requires that the levels of R_t and r_t are *I(1)*, and the spread S_t is *I(0)* and does not contain a unit root. In Table 4.1.1 we report Augmented Dickey-Fuller (ADF) t ratios which are a test of the null hypothesis that our null hypothesis is non-stationary; in particular that they follow random walks.

Table 4.1.1 -Test for a unit root

Interest Rate (1969(7)-1982(6))	ADF t-ratios	Interest Rates (1982(7)-1995(7))	ADF t-ratios
r_t	0.16	r_t	-0.21
R_t	0.65	R_t	-0.04
S_t	-4.3	S_t	-3.46

The critical value for t at 5 percent level of significance is -2.87.

As is clear, the interest rate variables R_t and r_t are non-stationary in level forms, and the spread is stationary i.e. *I(0)*. Therefore we have satisfied a necessary condition for VAR approach to test the expectations hypothesis of the term structure. We have also carried out Dickey and Pantula (1987) procedures to test the presence of 'rational bubbles'.Table 4.1.2 presents the results of Dickey and Pantula test for rational bubbles. The results clearly show that r_t and R_t have no bubbles i.e. they are not *I(2)* series.

Table 4.1.2 -Test for Rational Bubbles

Interest Rate (1969(7)-1982(6))	DF t-ratios	Interest Rates (1982(7)-1995(7))	DF t-ratios
∇r_t	-13.99	∇r_t	-13.21
r_t	-2.09	r_t	-2.11
∇R_t	-10.30	∇R_t	-13.76
R_t	1.99	R_t	-1.83

The critical value for t at 5 percent level of significance is -2.87.

4.2 Granger Causality Test

Table 4.2.1 contains results of tests of Granger causality from S_t to ∇r_t. In order to ensure that the tests were not biased or lacked power because of an inappropriate choice of lag length,, all lag lengths from two*(t-6)* to eight*(t-24)* were considered. In order to test that S_t Granger-causes ∇r_t, we need to test the equation:

$$\nabla r_t = \sum_{i=1}^{j} c_i S_{t-i} + \sum_{i=1}^{j} e_i \nabla r_{t-i} + \eta_t$$

such that $\sum_{i=1} c_i = 0$

The reported *F* statistics illustrate this implication of the expectations hypothesis of term structure. These statistics are supportive of the model, in that spreads (S_t) Granger-cause the changes in the short rates (∇r_t) in all lag lengths considered

Table 4.2.1- Granger Causality Test from S_t to ∇r_t .

$S_t \rightarrow \nabla r_t$	1969(7)-1982(6)	1982(7)-1995(7)
Lags		
t-6	14.90	5.14
	(3.07)	(3.07)
t-12	21.03	8.25
	(2.45)	(2.45)
t-18	14.36	3.66
	(2.17)	(2.17)
t-24	33.53	(5.04)
	(2.02)	(2.02)

Note: values in parenthesis are the critical values of F tests with degrees of freedom (lags) and n=130

4.3 VAR Test of the Expectations Hypothesis

In setting up the VAR model, the initial task is to determine the order of lag length. This is a delicate task since under-parameterisation will tend to bias results and over-parameterisation will diminish the power of the tests. We have identified the overlapping problem in our data and it has a MA (*2*) structure in the series and the

error term may exhibit an ARCH effect. Therefore, our procedure is to use a GMM method to test the cross-equation restriction conditions by taking into account of the presence of ARCH effect and also the MA (*2*) structure. To test the cross-equation restrictions, we need to test the equation:

$$S_t + \nabla r_t - \gamma^{-1} S_{t-3} = (a_1 + c_1 - \gamma^{-1}) S_{t-3} + \sum_{i=2}^{j} (a_i + c_i) S_{t-3i} + \sum_{i=1}^{j} (b_i + e_i) \nabla r_{t-3i} + \varepsilon_t + \eta_t$$

such that the slopes $(a_1 + c_1 - \gamma^{-1}) = 0$, $\sum_{i=2}^{j} (a_i + c_i) = 0$ and $\sum_{i=1}^{j} (b_i + e_i) = 0$

Our null hypothesis is that all the coefficients on the right hand side are equal zero. This means that the equation has no predictive power.

Table 4.3.1 - Test of the Rational Expectations Model of the Term Structure.

Data Series	**Wald Test**		
	~ χ^2 (6)	**~ χ^2 (8) ~ χ^2 (10)**	
	lag (t-9)	**lag (t-12)**	**lag (t-15)**
1969(7)-1982(6)	16.04	30.61	31.38
	(12.59)	(15.5)	(18.30)
1982(7)-1995(7)	9.33	18.64	38.40
	(12.59)	(15.5)	(18.30)

Wald Statistics are central χ^2 variates with ( ) degrees of freedom under the null hypothesis. Figures in parenthesis are the critical values of χ^2 at 5% level of significance.

Table 4.3.1 shows test of the rational expectation hypothesis of the term structure which is carried out by calculating the Wald statistic for the restrictions on the VAR representation of the S_t and ∇r_t. These statistics show that for the 1969(7)-1982(6) period, there is strong evidence of rejecting the hypothesis. However, for the 1982(7)-1995(7) period there is evidence of accepting the hypothesis with lag (t-9) but there is no support for accepting the model with lags (*t-12*) and (*t-15*) at 5 % level of significance.

5 Conclusion

We have investigated the expectations hypothesis of the term structure of interest rate using the 90-day Australian bank bill rate and the 10-year Treasury bond rate. We have tested the expectations hypothesis of the term structure in a VAR form using the spread (long rate minus short rate) and the change in the 90-day bank bill rate. This was carried out after we have ensured that the10-year Treasury bond rate and the bank bill rate were co-integrated which is the necessary condition for VAR model to test the expectations hypothesis of the term structure. It has shown that the restrictions implied by the expectations hypothesis of the term structure on the VAR model is not supportive by the Australian data at 5 % level of significance except for the second series of data with lag (*t-9*). However, when we are conducting a Granger causality test, the results showed that the spread Granger-caused the changes in the bank bill rates, which have an implication of the expectations hypothesis of the term structure. It can therefore infer that the validity of the expectations hypothesis of the term structure of interest rates may not be conclusively rejected.

Reference

1. Attfield, C. and Duck, N. (1982) " *Tests of rational expectations model of the term structure of UK interest rates* " Economic Letters.
2. Campbell J.Y and Shiller, R.J. (1987) '*Cointegration and tests of present value models*'- Journal of Political Economy.
3. Chow, G. C. (1989) "*Rational versus adaptative expectations in present value models*" Review of Economics and Statistics
4. Dickey, D. A. and Pantula, S. G. (1987) " *Determination of order of differencing in autoregressive processes* " Journal of Business and Economic Statistics
5. Jones, D. S. and Roley, V. (1983) " *Rational expectations and expectations model of the term structure: a test using weekly data* " Journal of Monetary Economics.
6. Leroy, S. F. and Porter, R. D. " *The present value relation: tests based on implied variance bounds generalised method of moments* " Econometrica.
7. Mark P Taylor '*Modelling the yield curve'*
8. MacDonald R. and A. Speight(1988) '*The term structure of interest rates in the UK'*- Bulletin of economic research.
9. Robert J. Shiller(1979) '*The volatility of long term interest rates and expectations models of the term structure'*.- Journal of political economy
10. Sargent, T.J(1979)*'A note on the maximum likelihood estimation of rational expectations model of the term structure*,' -Journal of Monetary Economics.
11. Shiller, R. J. (1979) " *The volatility of long term interest rates and expectations models of the term structure* " Journal of Political Economy
12. Terence C. Mills (1991) "*The term structure of UK interest rates: tests of the expectations hypothesis",* Applied Economics.

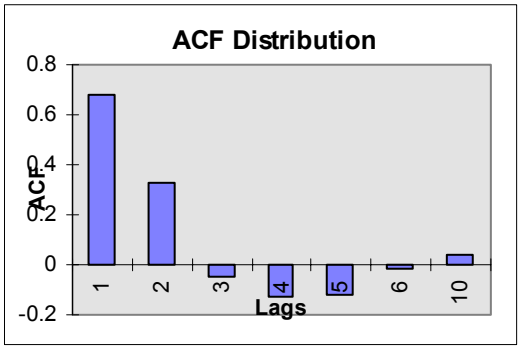
Appendix

Overlapping Data Problem

Since the data used are end of month on 90-day bank bill rates, there could poise an overlapping data problem, therefore, we need to test for overlapping data using the Auto Correlation Function (ACF). In theory, if there is data overlapping ACF would exhibit a number of outstanding spikes depending on the MA *(I)* structure. From our data set we would expect ACF has two outstanding spikes since it is a MA *(2)* structure.

Table 1A shows the lag length and the respective ACF results.

Lags	1	2	3	4	5	6	10	CI @ 95% confidence level
ACF	0.68	0.33	-0.05	-0.13	-0.12	-0.02	0.04	-0.16 +0.16



Statistically, it shows that the results do not follow the theoretical ACF distribution.

A Fourth Order *L*-stable Method for the Black-Scholes Model with Barrier Options

D.A. Voss, A.Q.M. Khaliq, S.H.K. Kazmi, and H. He

Department of Mathematics
Western Illinois University
Macomb, IL 61455, USA

Abstract. In this paper we develop a fourth order L–stable method for solving the Black–Scholes equation with barrier options. The method is based on a rational approximation to the matrix exponential function possessing real and distinct poles which allows the implementation of the algorithm as Backward Euler–like solves on concurrent processors. Due to barriers or nonsmooth payoffs which cause discontinuities in the solution, standard A–stable methods, such as Crank–Nicolson, are prone to produce large and spurious oscillations in the numerical solution which would lead to poor estimates of options. The proposed higher order method does not suffer these drawbacks due to its strong stability properties at infinity.

Key words. Black–Scholes PDE, Barrier options, L–stability, parallelism

1 Introduction

Barrier options are particularly challenging for traditional numerical pricing methods which can perform poorly due to discontinuities in the solution induced by the barriers (see, for example, [12]). In particular, it is known that the Crank–Nicolson method can permit the development of spurious oscillations unless the time step size is no more than twice that required for a fully explicit method to be stable (Zvan et al. [14]). Although the fully implicit Backward Euler method may used to effectively solve the Black–Scholes PDE due to its strong stability properties at infinity (L–stability), it is only first–order accurate in time. An alternative method developed by Rannacher [6], which uses a finite number of fully implicit steps followed by the usual Crank–Nicolson method, can quell such oscillations. However, this A–stable method is only a second order timestepping method as is other alternative L–stable BDF methods. While applicable higher order timestepping methods have been developed in [6] utilizing higher order diagonal Padé approximations, they entail higher powers of the semidiscretization matrix leading to enhanced computational complexity and ill–conditioning. L–stable methods have proven useful in numerous applications including, for example, buffered calcium waves [17] and photochemical dispersion problems [16].

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 199–207, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The purpose of this paper is to develop an efficient high order L–stable method suitable for solving the Black–Scholes equation with barrier options.

Black & Scholes obtained the value of a European call option as the solution to a final value problem for a second order parabolic partial equation

$$\frac{\partial C}{\partial \zeta} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0, \tag{1}$$

with

$$C\,(0,\zeta) = 0 \quad \& \qquad C\,(S,\zeta) = S - Ee^{-r(T-\zeta)} \quad as \;\; S \to \infty, \tag{2}$$

and

$$C(S,T) = \max(S-E,0). \tag{3}$$

In (1)–(3), ζ is the current time, T is the time of expiration, C denotes the value of the call option, E is the exercise price, S is the price of the underlying asset, σ is its volatility, and r is the continuously compounded risk–free interest rate. Option pricing PDEs are typically convection–diffusion type equations.

In the next section, we consider the semidiscretization of the transformed and dimensionless form [11] of (1)

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + (k-1)\frac{\partial u}{\partial x} - ku, \quad k = \frac{r}{\frac{1}{2}\sigma^2}, \quad t_0 \le t \le t_f, \quad a \le x \le b, \tag{4}$$

using central difference approximations to the spatial derivatives. We also examine the recurrence relation satisfied by the exact solution of the resulting inhomogeneous linear system of ODEs together with typical Padé approximations to the exponential used in time–stepping algorithms. In section 3, we focus on the design of a new fourth order rational approximation to the exponential (similar to that in Voss & Khaliq [13]) that leads to a stable, oscillation–free, time–stepping algorithm which can also utilize concurrent processors. Numerical results are reported on two problems in section 4. Our first example is a European call option, and the second example demonstrates that L–stable schemes are preferable to A–stable schemes when numerically solving the Black–Scholes PDE with discrete barrier options.

2 MOL Semidiscretization

The Method Of Lines (MOL) is an approach to the numerical solution of quite general PDEs that involve a time variable and one or more space variables. The partial derivatives with respect to the space variables are discretized to result in an approximating system of ODEs. For linear PDEs, the MOL approach leads to an approximating linear inhomogeneous system of ODEs whose solution satisfies a two-term recurrence relation involving a matrix exponential function, where the matrix is determined from the form of spatial discretization employed such

as finite difference, finite element, or spectral approach. The way in which one replaces this matrix exponential function determines the efficacy of the resulting method in a parallel computing environment. For the linear homogeneous heat equation Reusch *et al.* [5] describe an efficient serial, "multiplicative" algorithm involving the "factorization" of diagonal Padé approximations to the matrix exponential function. For the same equation, Gallopoulos and Saad [2] extend this to a parallel computing environment through an "additive" algorithm involving the "fractionization " of diagonal Padé approximations to the matrix exponential function. While possessing high order for such PDEs, diagonal Padé approximations lead to algorithms which are not strongly stable at infinity [7]; consequently, unless the time-step is suitably restricted, such algorithms will incur oscillations for problems possessing high frequency components in their solution. The associated restrictions were established by Khaliq *et al.* [3] and parallel schemes corresponding to L_0 –acceptable sub–diagonal Padé approximations were developed as high–order (complex) extensions of the fully implicit Backward Euler method.

Whether diagonal or not, high order Padé approximations to the matrix exponential function entail the use of complex arithmetic or require the squaring of the matrix factor thereby reducing sparseness while increasing the condition number.

For positive integer N and $X = b - a$, we let $h = \frac{X}{N+1}$ and approximate the exact solution, $u(mh, t)$, of (4) at $x = mh$ by $u_m(t)$, $m = 0, 1, \ldots, N + 1$. The approximations are to satisfy a system of ODEs arising through the replacement of the partial derivatives in (4) with respect to x by the second order central differences

$$\frac{\partial u}{\partial x} = \frac{u(x+h,t) - u(x-h,t)}{2h} + O(h^2)$$
$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2} + O(h^2)$$

The resulting initial value system has the form

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u} + \mathbf{b}(t), \quad \mathbf{u}(0) = \mathbf{f}, \tag{5}$$

where

$$\mathbf{u} = [u_1(t), \ldots, u_N(t)]^T,$$
$$A = \frac{1}{h^2} tridag[1 + \frac{h}{2}(1-k), -2-k, 1 - \frac{h}{2}(1-k)],$$

and $\mathbf{f}$ and $\mathbf{b}(t)$ are the corresponding transformed initial and boundary conditions for (1). The exact solution of (5) satisfies the two term recurrence relation

$$\mathbf{u}(t+\tau) = e^{\tau A}\mathbf{u}(t) + \tau \int_0^1 e^{\tau(1-s)A}\mathbf{b}(t+s\tau)ds \tag{6}$$

which forms the basis of various time-stepping methods through approximating the exponential function by rational functions. The eigenvalues of A are well

known [8], and if $1 - \frac{2}{h} < k \leq 1 + \frac{2}{h}$, then they are all real and negative and have the form

$$\lambda_s = -\frac{2}{h^2} + \frac{2}{h^2}\sqrt{1 - \left(\frac{h}{2}(1-k)\right)^2}\,\cos(s\pi h), \quad s = 1, \ldots, N; \tag{7}$$

otherwise, the eigenvalues of A are complex with negative real part with form

$$\lambda_s = -\frac{2}{h^2} + \frac{2}{h^2} i\sqrt{\left(\frac{h}{2}(1-k)\right)^2 - 1}\,\cos(s\pi h), \quad s = 1, \ldots, N. \tag{8}$$

When k is large the PDE in (4) is *convection–dominated* and, consequently, the order used by one of the popular BDF codes might be restricted by stability [7]. This situation can occur if the interest rate is very high relative to the volatility [15].

Diagonal $[m/m]$Padé approximations, $R_{m,m}(z) = \frac{P_m(z)}{Q_m(z)}$, may be used to approximate the exponential function in (6). Such algorithms involve the solution of linear systems wherein the coefficient matrix is a polynomial of degree m in τA. This increases the computational complexity of higher order schemes corresponding to $m > 1$, while reducing the sparseness of the coefficient matrix. To circumvent these effects, as did Reusch *et al.* [5] for the simple heat equation, $[m/m]$Padé approximations can be factorized into the form

$$R_{m,m}(z) = \frac{P_m(z)}{Q_m(z)} = \prod_{n=1}^{m} \frac{1 + \frac{z}{c_n}}{1 - \frac{z}{c_n^*}}, \tag{9}$$

which yields the efficient multiplicative algorithm for serial computers:

$$\begin{gathered} \mathbf{u}^0 = \mathbf{u}(t) \\ \textit{Solve} \quad (I - \frac{\tau}{c_n^*}A)\mathbf{u}^n = (I + \frac{\tau}{c_n}A)\mathbf{u}^{n-1} \quad \textit{sequentially}, \quad n = 1, \ldots, m \\ \mathbf{u}(t+\tau) = \mathbf{u}^m. \end{gathered} \tag{10}$$

For parallel computers, as did Gallopoulos and Saad [2] for the simple heat equation, $[m/m]$Padé approximations can be decomposed into the form

$$R_{m,m}(z) = (-1)^m + \sum_{n=1}^{q_1} \frac{w_n}{z - c_n} + 2 \sum_{n=q_1+1}^{q_1+q_2} Re(\frac{w_n}{z - c_n}), \tag{11}$$

where $R_{m,m}(z)$ possesses q_1 real poles and $2q_2$ complex poles with $q_1 + 2q_2 = m$. On a machine with at least $q_1 + q_2$ processors this leads to the following additive algorithm which is no more complicated than a complex variant of the $[1/1]$ Padé, that is, the Crank-Nicolson method:

$$\textit{Solve} \quad (\tau A - c_n)\mathbf{v}_n = \mathbf{u}(t) \quad \textit{concurrently}, \quad n = 1, \ldots, q_1 + q_2$$

$$\mathbf{u}(t+\tau) = (-1)^m \mathbf{u}(t) + \sum_{n=1}^{q_1} w_n \mathbf{v}_n + 2 \sum_{n=q_1+1}^{q_1+q_2} Re(w_n \mathbf{v}_n).$$

3 Fourth Order Time–Stepping

We consider rational approximations to e^z of the form

$$R(z) = \frac{N(z)}{D(z)} = \frac{1 + a_1 z + a_2 z^2 + a_3 z^3}{(1 - b_1 z)(1 - b_2 z)(1 - b_3 z)(1 - b_4 z)}, \tag{12}$$

where the coefficients a_i and b_i are real and $b_i > 0$. Approximation (12) will be of order p to the exponential function if $R(z) = e^z + C_{p+1} z^{p+1} + O(z^{p+2})$, where C_{p+1} denotes the error constant. It is known (see [4]) that the maximum order attainable by $R(z)$ is four, and that the smallest error constant occurs in the case of repeated poles, $b_i = b$. A rational approximation to e^z is said to be *A–acceptable* if $|R(z)| < 1$ whenever $Re(z) < 0$ and *L–acceptable* if, in addition, $|R(z)| \to 0$ as $Re(z) \to -\infty$. We consider *L–acceptable* rational approximations to e^z as oscillations are prevalent in methods arising from the *A–acceptable* diagonal $[m/m]$Padé approximations. The *L–acceptable*, fourth order $R(z)$ with minimal error constant has $b_i = b \approx 0.572$ and $|C_5^*| \approx .02725$.

Replacing the exponential function by such rational repeated pole approximations leads to inherently serial algorithms. Instead, we have constructed the *L–acceptable* rational approximation

$$R(z) = \frac{1 - \frac{54}{35} z + \frac{39}{140} z^2 + \frac{13}{42} z^3}{(1 - z)(1 - \frac{2}{5} z)(1 - \frac{9}{14} z)(1 - \frac{1}{2} z)} \tag{13}$$

similar to that in [13] but possessing rational coefficients and an error constant, C_5, of smaller magnitude where $|C_5| = \frac{1}{28} \approx 0.0357 \approx |C_5^*|$. Since $R(z)$ possesses distinct real poles, it admits a partial fraction expansion of the form

$$R(z) = \sum_{n=1}^{4} \frac{w_n}{1 - b_n z}, \tag{14}$$

where the expansion coefficients are found in the usual way

$$w_n = \lim_{z \to \frac{1}{b_n}} (1 - b_n z) R(z),$$

and the ensuing algorithm can be implemented in a multiprocessor environment. The recurrence relation (6) is then approximated by

$$\mathbf{u}(t + \tau) = R(\tau A)\mathbf{u}(t) + \tau \sum_{i=1}^{3} Q_i(\tau A)\mathbf{b}(t + \sigma_i \tau) \tag{15}$$

where the rational functions $Q_i(z)$ are of form (12) with the same $b_n, n = 1(1)4$, as $R(z)$ in Table 1, and the quadrature nodes are selected as $\sigma_1 = 0, \sigma_2 = \frac{1}{2}$, and $\sigma_3 = 1$. Fourth order accuracy prevails if [1]

$$\sum_{i=1}^{3} \sigma_i^l Q_i(z) = \frac{l!}{z^{l+1}} (R(z) - \sum_{j=0}^{l} \frac{z^j}{j!}), \quad l = 0, 1, 2 \tag{16}$$

which completes the determination of $Q_i(z), i = 1(1)3$. Similar to (14), each of these rational functions are decomposed into the form

$$Q_i(z) = \sum_{n=1}^{4} \frac{\delta_n^{(i)}}{1 - b_n z}, \quad i = 1, 2, 3 \tag{17}$$

where the expansion coefficients,

$$\delta_n^{(i)} = \lim_{z \to \frac{1}{b_n}} (1 - b_n z) Q_i(z), \tag{18}$$

are listed in Table 2.

n	1	2	3	4
b_n	1	$\frac{2}{5}$	$\frac{9}{14}$	$\frac{1}{2}$
w_n	$\frac{19}{45}$	$\frac{-2500}{153}$	$\frac{-2401}{255}$	$\frac{79}{3}$

Table 1. Definition of $R(z)$

n / i	1	2	3	4
1	$\frac{38}{45}$	$\frac{-440}{153}$	$\frac{-1491}{340}$	$\frac{79}{12}$
2	$\frac{-76}{45}$	$\frac{-320}{153}$	$\frac{378}{85}$	0
3	$\frac{19}{15}$	$\frac{-80}{51}$	$\frac{-2079}{340}$	$\frac{79}{12}$

Table 2. Expansion coefficients $\delta_n^{(i)}$ of $Q_i(z)$

The resulting L–stable, parallel time–stepping algorithm, denoted $L4$, based on recurrence relation (15) then becomes:

$$\begin{aligned} &Solve \quad (I - b_n \tau A)\mathbf{v}_n = w_n \mathbf{y}(t) + \tau \sum_{i=1}^{3} \delta_n^{(i)} \mathbf{b}(t + \sigma_i \tau), \quad n = 1, \ldots, 4 \\ &\mathbf{y}(t + \tau) = \sum_{n=1}^{4} \mathbf{v}_n. \end{aligned} \tag{19}$$

The typically large and sparse linear systems in (19) can be solved concurrently on four processors. Of course, further opportunity for parallelism exists in space, that is, in parallel numerical linear algebra routines for solving each system in (19).

4 Numerical Results

Problem 1: We consider the Black–Scholes PDE in transformed and dimensionless form ([11], Sec. 5.4) using parameter values from ([10], Chap. 13).

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + (k-1)\frac{\partial u}{\partial x} - ku, \quad t_0 \le t \le t_f, \quad a \le x \le b$$

where $k = \frac{r}{\frac{1}{2}\sigma^2},\ r = 0.065,\ \sigma = 0.8,\ a = \ln(\frac{2}{5}),\ b = \ln(\frac{7}{5}),\ t_0 = 0,\ t_f = 5$, with initial condition

$$u(x,0) = \max(\exp(x) - 1, 0)$$

and boundary conditions

$$u(a,t) = 0, \quad u(b,t) = \frac{7 - 5\exp(-kt)}{5}.$$

Using a uniform space mesh consisting of 40 points, even with the kink that exists in the initial condition at $x = 0$, Figure 1 displays the oscillation–free surface plot produced by method L4 at 20 equally spaced time levels.

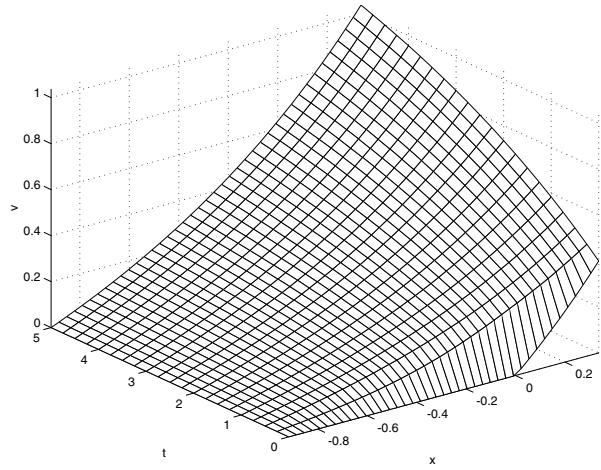


Fig. 1. European call option

Problem 2: By defining $\eta = T - \zeta$, we consider (1) in the more conventional form

$$\frac{\partial V}{\partial \eta} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV,$$

$$V(S, \eta = 0) = \begin{cases} \max(S - E, 0) \text{ for a call} \\ \max(E - S, 0) \text{ for a put} \end{cases}$$

$$V(0, \eta) \to V_\eta = -rV$$

$$V(S = \infty, \eta) \to \begin{cases} \approx S \text{ for a call} \\ \approx 0 \text{ for a put} \end{cases}.$$

With uniform space and time steps, $\Delta S = .05$ and $\Delta\eta = 0.002$, Figure 2 and Figure 3 display the European double knock–out call option at time $\eta = 0.5$ using the Crank–Nicolson and L4 methods, respectively. Here we have used the same parameter values as in [14], namely, $r = 0.10, \sigma = 0.2, E = 100$, and with barriers $H_{lower} = 95, H_{upper} = 110$. Figure 2 clearly exhibits the oscillatory behavior of the Crank–Nicolson method while the the L–stable, fourth order method $L4$ returns an accurate representation of the solution.

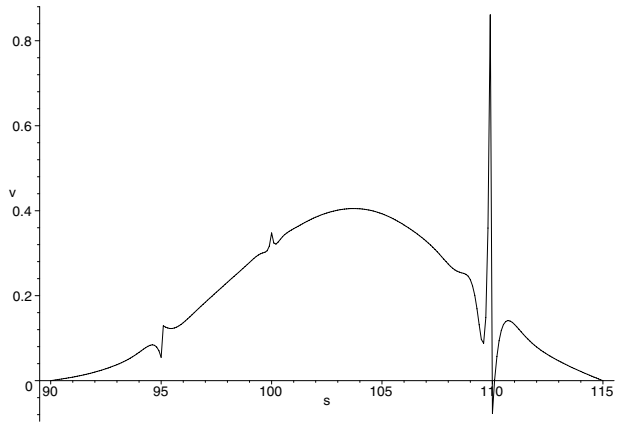


Fig. 2. Double barrier option: Crank–Nicolson method

References

1. P. Brenner, M. Crouzeix, and V. Thomée, Single step methods for inhomogeneous linear differential equations in a Banach space, RAIRO Anal. Numér., 16 (1982), 5–26.
2. E. Gallopoulos and Saad, On the parallel solution of parabolic equations, CSRD Report No. 854 (1989).
3. A. Q. M. Khaliq, E. H. Twizell, and D. A. Voss, On parallel algorithms for semidiscretized parabolic partial differential equations based on subdiagonal Padé approximations, Numer. Meth. Partial Diff. Equ., 9 (1993), 107–116.

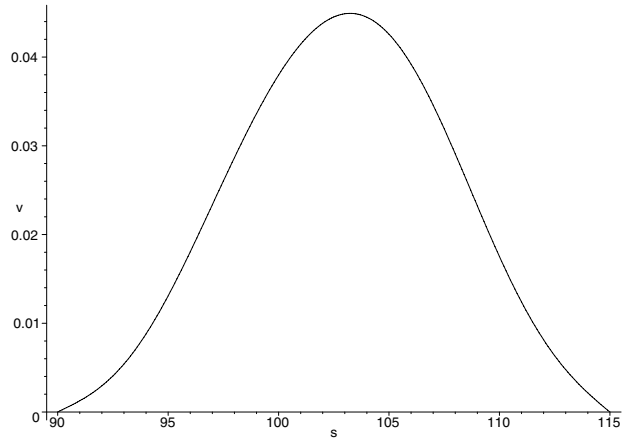


Fig. 3. Double barrier option: L4

4. S. P. Norsett and A. Wolfbrandt, Attainable order of rational approximations to the exponential function with only real poles, BIT, 17 (1977), 200–208.
5. M. F. Reusch, L. Ratzan, N. Pomphrey, and W. Park, Diagonal Padé approximations for initial value problems, SIAM J. Sci. Stat. Comput., 9 (1988), 829–838.
6. R. Rannacher, Finite element solution of diffusion problems with irregular data, Numerische Mathematik, 43 (1984), 309–327.
7. L. F. Shampine, ODE Solvers and the Method of Lines, Numer. Meth. Partial Diff. Equ., 10 (1994), 739–755.
8. G. D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, Oxford U. P., London, 1985.
9. C. Johnson, Y. Nie, and V. Thomée, An a posteriori error estimate and adaptive timestep control for a backward Euler discretization of a parabolic problem, SIAM J. Numer. Anal., 2 (1990), 277–291.
10. S. N. Neftci, An Introduction to the Mathematics of Financial Derivatives, Academic Press, 2000.
11. P. Wilmott, S. Howison, and J. Dewynne, The Mathematics of Financial Derivatives: A Student Introduction, Cambridge University Press, 1995.
12. D. Tavella and C. Randall, Pricing Financial Instruments, John Wiley & Sons, Inc., 2000.
13. D.A. Voss and A.Q.M. Khaliq, Time–stepping algorithms for parabolic PDEs based on rational approximations with real distinct poles, Advances in Computational Mathematics, vol. 6, pp. 353–363, 1996.
14. R. Zvan, K. R. Vetzal, and P. A. Forsyth, PDE methods for pricing barrier options, Journal of Economic Dynamics & Control, 24 (2000), 1563–1590.
15. R. Zvan, K. R. Vetzal, P. A. Forsyth, and K. R. Vetzal, Robust numerical methods for PDE models of Asian options, J. Computational Finance, 1 (1998), 39–78.
16. J. G. Verwer, E. J. Spee, J. G. Blom, and W. Hundsdorfer, A second–order Rosenbrock method applied to photochemical dispersion problems, SIAM J. Sci. Comput. **20**, 1456–1480 (1999).
17. B .M. Slepchenko, J. C. Schaff, and Y. S. Choi, Numerical approach to fast reactions in reaction-diffusion systems: application to buffered calcium waves in bistable model, J. Comput. Phys., 162 (2000), 186–218.

Ranking Decision Variants by Subjective Paired Comparisons in Cases with Incomplete Data

Miroslaw Kwiesielewicz [1] and Ewa van Uden [2]

[1] Technical University of Gdansk
Faculty of Electrical and Control Engineering
Narutowicza 11/12, Gdansk, POLAND, 80-952
Phone: +48 58 3472124, Fax: +48 58 3471802
mkwies@ely.pg.gda.pl

[2] Technical University of Gdansk
Faculty of Electrical and Control Engineering
Narutowicza 11/12, Gdansk, POLAND, 80-952
Phone: +48 58 3471225, Fax: +48 58 3471802
cotynato@ely.pg.gda.pl

Abstract. The pairwise comparison method is widely used to rank a finite, usually small number of decision variants especially in a case when neither a direct evaluation nor the utility theory gives satisfactory results. In this method, an expert or a group of experts is asked to provide his/their opinions concerning each pair of factors expressing a relative importance of one variant in a pair over the second one. It happens however that an expert or few experts cannot provide his/their opinions concerning a pair or pairs of factors. In such a case the resulting judgement matrices are incomplete and a problem of estimating missing data arises. The paper addresses some issues concerning lacking data. Some numerical examples are included.

1 Introduction

A concept of paired comparisons was originally introduced by Thurstone [1,2,3] and developed by Saaty [4] and based on relative preferences concerning pairs of decision variants. It is an element of the well-known AHP (Analytic Hierarchy Process) [4], widely used multiattribute decision-making method. The paired comparisons are used to elicit knowledge from an expert. In cases with many attributes rankings are aggregated by a simple additive method. The AHP is a useful tool in different areas of applications, such as e.g.: risk analysis, safety management, economical and technical problems.

In the AHP an expert or a group of experts is asked to provide his/their opinions concerning each pair of factors. Next, based on an assumed scale, a judgement matrix is created and a ranking of factors is extracted from expressed opinions. It happens however that an expert or a group of experts cannot provide his/their opinions concerning a pair or pairs of factors. In such a case the resulting judgement matrices are incomplete and a problem of estimating missing data arises.

Many authors have discussed the problem of missing data and there is a different philosophy standing behind each approach of estimating them. The aim of this paper

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 208-215, 2003.
© Springer-Verlag Berlin Heidelberg 2003

is an elicitation of the ideas of different methods of estimating missing data. It can be very helpful for the decision-maker to select the proper method of fulfilling the empty places in the judgement matrices, which will be the most suitable for his purposes. Knowledge of the intentions hidden behind the different proposals seems to be a very important matter. It is worth underlying that every method is based on different properties of reciprocal matrices and is a result of different methods of approximation of judgement matrices.

Harker's method [5] is based on the eigenvalue method [4]. Shiraishi, Obata, Daigo built their method on an observation of the coefficients of the characteristic polynomial [6]. Unfortunately, their method allows them to estimate only one missing entry. In [7] Harker proposes to reduce the complexity of the preference eliciting process. For example he notices that in order to handle a multiatribute decision problem with 9 alternatives and 5 criteria, the experts must answer 190 questions. To simplify the whole process, he proposes to start out with incomplete pairwise comparison matrices and then to complete them in the most consistent way.

Kwiesielewicz's approach [8] assumes that the missing data are in the form of ratios consistent with the result ranking. The approach presented by van Uden [9] is based on the geometric mean method [4] and extracts the missing data from the intermediate comparisons. Carmone [10] investigated incomplete pairwise comparison matrices using Monte Carlo method. Usually the value of consistency index [4] is taken as an efficiency criterion of the fulfilling the empty places in the matrices by applying different methods.

2 Paired Comparison

The matrix of judgements $\mathbf{R} = \lfloor \mathbf{r}_{ij} \rfloor_{nxn}$ that expresses the relative significance of factors F_i and F_j is called reciprocal if $r_{ij} = 1/r_{ji}$ for all $i, j = 1,2,\ldots,n$ and $r_{ij} > 0$. The matrix $\mathbf{R}$ is called consistent if $r_{ij} r_{jk} = r_{ik}$ for all $i, j, k = 1,2,\ldots,n$. When the matrix $\mathbf{R}$ is consistent then the ranking of factors is univocal and follows from any row or column [4].

In Saaty's eigenvector method a vector $\mathbf{w}$ corresponding to the maximal eigenvalue $\lambda_{\max}$ is obtained. It is a solution of the following eigenvalue problem

$$\mathbf{Rw} = \lambda_{\max} \mathbf{w} \tag{1}$$

In the logarithmic least square method (geometric mean method) a vector $\mathbf{w}$ minimising the following expression

$$\sum_{i,j=1}^{n} \left(\ln\left(r_{ij}\right) - \ln\left(w_i / w_j\right) \right)^2 \tag{2}$$

is obtained. The solution of this problem is given by [9]:

$$w_i = \left(\prod_{j=1}^{n} r_{ij} \right)^{\frac{1}{n}} \qquad i = 1,2,\ldots,n \tag{3}$$

When the matrix $\mathbf{R}$ is consistent, both these methods give the same ranking. If not, the results can differ.

Many authors have discussed the problem of incomplete matrices. There have been some proposals for estimating missing data, e.g. in [5,6] or [7]. Each of them is based on different properties of reciprocal matrices and is a result of different methods of approximation of judgement matrices. In all these methods it is important to obtain a matrix with an acceptable consistency index, which is calculated according to the formula [4]:

$$C.I. = \frac{\lambda_{\max} - n}{n-1} . \tag{4}$$

3 Chosen Methods for Cases with Incomplete Set of Judgements

Sometimes an expert is not able to provide his opinion concerning a pair or the certain pairs of factors. This situation is very typical in reliability, when there does not exist a sufficient collection of data. The reason of this fact follows from rarely appearing damages or disasters. In such cases a problem arises how to obtain the priority vector based on incomplete information. The pairwise comparison method allows to deal with incomplete matrices of judgements in some aspects, but the approach chosen depends on the intentions of the decision-maker. The aim, which the decision-maker wants to achieve suggest the proper method of filling the missing data. The exemplary possibilities are the following ones:

to obtain the most consistent set of judgements,

to allow the expert to agree with the rest of a group,

to extract information about missing data based on all intermediate comparisons,

to assume that in empty places there should appear the weights from the final ranking.

3.1 Shiraishi, Obata, Daigo Method [3]

This method is based on certain properties of the coefficients of characteristic polynomial of matrix $\mathbf{R}$. To solve the problem, an eigenvalue approach was applied. The authors consider an incomplete pairwise comparison matrix, which has one missing entry - *x*. They try to solve the following optimization problem:

$$\min_{x} \lambda_{\max} \tag{5}$$

Instead of solving (5) exactly they proposed a heuristic method in which they minimize:

$$\min_x c_3(x) \tag{6}$$

where

$$c_3(x) = \sum_{i<j<k} \{2 - \left(\frac{r_{ij} r_{jk}}{r_{ik}} + \frac{r_{ik}}{r_{ij} r_{jk}} \right). \tag{7}$$

Their method consists from the following steps:

Step 1: Find x_0 of the solution of the problem (6).

Step 2: Calculate the largest eigenvalue of $\mathbf{R}(x_0)$ and its associate eigenvector.

Step 3: Normalize the eigenvector into a priority weight vector.

Unfortunately this method can not be applied for more then one missing datum.

3.2 Harker's Method [5]

In this method also an eigenvalue technique is used. Formally, the method can be described as follows: for given incomplete matrix $\mathbf{R} = (r_{ij})$, define the corresponding derived reciprocal matrix $\widetilde{\mathbf{R}} = (\widetilde{r}_{ij})$ by:

$$\widetilde{r}_{ij} = \begin{cases} 1 + m_i, & \text{if } i = j, \\ 0, & \text{if } i \neq j \text{ and } (i, j) - \text{component is missing}, \\ r_{ij} & \text{otherwise}, \end{cases} \tag{8}$$

where m_i denotes the number of missing components in the i-th row.

Step 1: Construct a derived reciprocal matrix $\widetilde{\mathbf{R}}$ of $\mathbf{R}$.

Step 2: Calculate the largest eigenvalue $\widetilde{\lambda}_{\max}$ of $\widetilde{\mathbf{R}}$ and its associate eigenvector.

Step 3: Normalize the eigenvector into a priority weight vector.

This method can be applied to one or more missing data and assumes that in empty places there should appear the weights from the final ranking.

3.3 Kwiesielewicz's Method [8]

The method is based on logarithmic least squares approach. In the case of one decision-maker the problem leads to the geometric mean method. This approach allows solving the problem for one and more decision-makers. The minimization problem (2) leads to a set of so-called normal equations

$$x_i \sum_{i\neq j,\,j=1}^{n} d_{ij} - \sum_{i\neq j,\,j=1}^{n} d_{ij}x_j = \sum_{i\neq j,\,j=1}^{n}\sum_{k=1}^{d_{ij}} y_{ijk},\ i=1,2,\ldots,n\ , \tag{9}$$

where:

$$d_{ij} \geq 0, \forall i,j\ ,\ \sum_{i\neq j,\,j=1}^{n} d_{ij} > 0, \forall i\ ,$$

$$x_i = \ln(p_i),\ y_{ijk} = \ln(r_{ijk}) \text{ for } k = 1,\ldots,d_{ij}\ .$$

where d_{ij} is a number of judgements per the pair (*i,j*).

which can be written in a matrix form

$$\mathbf{Ax} = \mathbf{b} \tag{10}$$

The rank of the matrix $\mathbf{A}$ is less then *n*. For the solution a pseudoinverse approach was used. When matrix $\mathbf{A}$ has rank *n-1* (as is usually in practice) then solution always exists and has a form:

$$\mathbf{x} = \mathbf{A}^{+}\mathbf{b} \tag{11}$$

where the solution vector **x** is unique and geometrically normalized.

In the case of multiple decision-makers and different judgements for each pair of objects the problem can be solved in two ways: (i) assume that in place of lacking judgements the number 1 is put, what means that for a decision-maker which was not able to compare a pair of objects no preference is assumed, (ii) assume that in place of the lacking judgements or nothing is put or a number which is consistent with a final solution. In both cases we obtain different solutions depending on assumptions we made, but nevertheless we obtain a geometrically normalized final solution.

3.4 Van Uden's Method [9]

This method is based on geometric mean method and missing data are extracted from all intermediate comparisons. The formulae of calculating missing entries are dependent on the quantity of missing data. For one missing datum r_{ik} (and in consequence also r_{ki}), let us denote $X = \prod_{\substack{j \\ j\neq k}} r_{ij}$, $Y = \prod_{\substack{j \\ j\neq i}} r_{kj}$. Moreover, assume that $r_{ik} = \frac{w_i}{w_k}$, where $\mathbf{W}$ is a consistent approximation of the matrix $\mathbf{R}$. The most

consistent approximation of missing datum r_{ik} from the point of view of the geometric mean method is

$$r_{ik} = \sqrt[n-2]{\frac{X}{Y}} \,. \tag{11}$$

In the case of two missing data in any row or column (r_{ik} and r_{il}), let us introduce the following notation: $X = \prod_{\substack{j \\ j \neq k, j \neq l}} r_{ij}$, $Y = \prod_{\substack{j \\ j \neq i}} r_{kj}$ and $Z = \prod_{\substack{j \\ j \neq i}} r_{lj}$, and moreover, assume that $r_{ik} = \frac{w_i}{w_k}$, $r_{il} = \frac{w_i}{w_l}$. The most consistent approximation of missing data r_{ik} and r_{il} from the point of view of the geometric mean method is:

$$r_{ik} = \frac{X^{\frac{1}{n-3}} Y^{\frac{2-n}{(n-1)(n-3)}}}{Z^{\frac{1}{(n-1)(n-3)}}} \qquad \text{and} \qquad r_{il} = \frac{X^{\frac{1}{n-3}} Z^{\frac{2-n}{(n-1)(n-3)}}}{Y^{\frac{1}{(n-1)(n-3)}}} \,. \tag{12}$$

Procedure described above can be easily extended to cases with more lacking judgements.

4 Numerical Example

In order to illustrate the methods described in previous section, one example will be calculated applying four presented above methods. Let's assume the following judgement matrix containing expert opinions:

$$\mathbf{R} = \begin{pmatrix} 1 & 3 & - & 1/2 \\ 1 & 1/2 & 3 & 4 \\ 1/3 & 1 & 2 & 1/5 \\ 2 & 1 & - & 9 \\ - & 1/2 & 1 & 1/9 \\ 1/3 & - & 1 & 2 \\ 2 & 5 & 9 & 1 \\ 1/4 & 1/9 & 1/2 & 1 \end{pmatrix}$$

Note that the first expert did not provide his opinion concerning the pair $(1,3)$, while the second one did not provide his opinion concerning the pair $(2,3)$. Separately the judgements matrices for the experts are equal respectively:

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 3 & - & 1/2 \\ 1/3 & 1 & 2 & 1/5 \\ - & 1/2 & 1 & 1/9 \\ 2 & 5 & 9 & 1 \end{bmatrix} \text{ and } \mathbf{R}_2 = \begin{bmatrix} 1 & 1/2 & 3 & 4 \\ 2 & 1 & - & 9 \\ 1/3 & - & 1 & 2 \\ 1/4 & 1/9 & 1/2 & 1 \end{bmatrix}$$

Only Kwiesielewicz's method allows making calculations on entire matrix $\mathbf{R}$. The other methods will be applied for the matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ and the aggregation will be made according to a simple additive method, with the weights for experts equal to 1/2 and 1/2.

Kwiesielewicz's method (for entire matrix $\mathbf{R}$) gives the following ranking:

$$(0.35,0.29,0.12,0.24)$$

Kwiesielewicz's method (for the matrices $\mathbf{R}_1$ and $\mathbf{R}_2$) gives the following ranking:

$$(0.27,0.42,0.04,0.27)$$

Van Uden's method gives the ranking:

$$(0.29,0.33,0.09,0.30)$$

Harker's method gives the ranking:

$$(0.29,0.33,0.09,0.30)$$

Shiraishi method gives the ranking:

$$(0.27,0.33,0.09,0.30)$$

It's necessary to stress that when matrix size increases, the results differ more. Such phenomenon can be also observed when the matrices are less consistent.

5 Conclusions

The problem of incomplete comparison matrices in the paired comparison method used in the AHP was considered in this paper. A short review of the literature in this field was done and chosen methods were recalled and compared. Some exemplary numerical calculations were performed.

The choice of a proper method of filling missing data has a meaningful influence on the relative weights. At the beginning of a decision-making process, it is necessary

to realize what intentions should be satisfied according to a decision-maker. After this step, the choice of the most suitable method can be done.

References

1. Thurstone L.LA low of comparative judgements. *Psychological Rev*., 34, (1927) 273-286.
2. Thurstone, L.L. Psychophysical Analysis. *American Journal of Psychology,* 38, (1927b) 368-389.
3. Thurstone, L.L. The method of paired comparisons for social values. *Journal of Abnormal Social Psychology,* 21, (1927c) 384-400.
4. Saaty, T.L.: The Analytic Hierarchy Process Series I, RWS publication, 1990.
5. P.T. Harker, Alternative modes of questioning in the analytic hierarchy process. Mathematical Modelling **9: 3**, (1987) 353-360.
6. S. Shiraishi and T. Obata and M. Daigo, Properties of a positive reciprocal matrix and their application to AHP, Journal of the Operations Research Society of Japan **41: 3,** (1998) 404-414.
7. P.T. Harker, Incomplete pairwise comparisons in the analytic hierarchy process. Mathematical Modelling **9: 11**, (1987) 837-848.
8. M. Kwiesielewicz The logarithmic least squares and the generalized pseudoinverse in estimating ratios, European Journal of Operational Research, **93**, (1996) 611-619.
9. E. van Uden, Estimating missing data in pairwise comparison matrices. Operational and Systems Research in the Face to Challenge the XXI Century, Methods and Techniques in Information Analysis and Decision Making (Z. Bubnicki, O. Hryniewicz and R. Kulikowski Eds.) Academic Printing House, Warsaw, (2002) II-73-II-80.
10. F. Carmone, Kara A., Zanakis S. H. A Monte Carlo Investigation of Incomplete Pairwise Comparison Matrices in AHP, European Journal of Operational Research, **102: 3**, (1997) 538-553.

Quantum Games and Minimum Entropy

Edward Jiménez[1,2]

[1] CRM and Experimental Economics, Todo1 Services Inc, Miami Fl 33126
[2] Research and Development Department, Petroecuador, Quito-Ecuador, 5932
ejimenezecu@yahoo.com

Abstract. This paper analyze Nash's equilibrium (maximum utility MU) and its relations with the order state(minimum entropy ME). I introduce the concept of minimum entropy as a paradigm of both Nash-Hayek's equilibrium. The ME concept is related to Quantum Games. One questions arises after completing the exercise: What do the Quantum Mechanics postulates indicate about Game Theory and Economics?

1 Introduction

The quantum games and the quantum computer are closely-related. The science of Quantum Computer is one of the modern paradigms in Computer Science and Game Theory [2], [3], [4], [5], [6] [7], [8]. Quantum Computer increases processing speed to have a more effective data-base search. The quantum games incorporate Quantum Theory to Game Theory [4] algorithms. This simple extrapolation allows the Prisoner's Dilemma to be solved and demonstrates that the cooperative equilibrium is viable and stable with a probability differential of zero. In [4], [5] and [6], the analogy between Quantum Games and Game Theory is expressed as: "At the most abstract level, game theory is about numbers of entities that are efficiently acting to maximize or minimize. For a quantum physicist, it is then legitimate to ask: what happens if linear superpositions of these actions are allowed for?, that is, if games are generalized into the quantum domain. For a particular case of the Prisoner's Dilemma, we show that this game ceases to pose a dilemma if quantum strategies are implemented for."They demonstrate that classical strategies are particular quantum strategies cases. Eisert, Wilkens, and Lewenstein, 2001, not only provide a physical model of quantum strategies but also express the idea of identifying moves using quantum operations and quantum properties. This approach appears to be fruitful in at least two ways. On one hand, several recently proposed quantum information application theories can already be conceived as competitive situations, where several factors which have opposing motives interact. These parts may apply quantum operations using a bipartite quantum system. On the other hand, generalizing decision theory in the domain of quantum probabilities seems interesting, as the roots of game theory are partly rooted in probability theory. In this context, it is of interest to investigate what solutions are attainable if superpositions of strategies are allowed [10], [14], [15] and [16]. A game is also related to the transference of information. It is possible to ask: What happens if these carriers of information are applied

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 216–225, 2003.
© Springer-Verlag Berlin Heidelberg 2003

to be quantum systems, where Quantum information is a fundamental notion of information [8] . Nash's equilibria concept as related to quantum games is essentially the same as that of game theory, but the most important difference is that the strategies appear as a function of quantum properties in the physical system [14], [15], [19] and [20]. Econophysics is a new branch of scientific development that seeks to establish analogies between economics and physics in a particular case (analogies between Game Theory and Quantum Mechanic). The establishment of analogies is a creative way to apply the idea of cooperative equilibrium. The result of this cooperative equilibrium will allow synergies between these two sciences to occur. From my point of view, the power of physics is its ability to equilibrium formal treatment in stochastic dynamic systems. Therefore, the power of economics resides in its formal study of rationality, cooperation and the non-cooperative equilibrium. Econophysics is the beginning of a unification stage of the systemic approach of scientific thought. I demonstrate that it is the beginning of a unification stage, but it is also related to the remaining sciences. This paper essentially explains the relationship which exists among Quantum Mechanics, Nash's equilibria, and the Minimum Entropy Principle. This paper is organized as follows: Section 2: Quantum Games or Minimum Entropy, Section 3: Application of the Model. Section 4: Conclusion.

2 Quantum Games or Minimum Entropy

Let $\Gamma = (K, S, v)$ be a game with $n-$players, where K is the set of players $k = 1, ..., n$. The finite set S_k of cardinality $l_k \in N$ is the set of pure strategies of each player, where $k \in K$, $s_{kj_k} \in S_k$, $j_k = 1, ..., l_k$ and $S = \Pi_K S_k$ represents the set of pure strategy profiles with $s \in S$ an element of that set, $l = l_1, l_2, ..., l_n$ represents the cardinality of S . The vectorial function $\mathbf{v} : S \to R^n$ associates every profile $s \in S$, where the vector of utilities $\mathbf{v}(s) = (v^1(s), ..., v^n(s))^T$, and $v^k(s)$ designates the utility of the player k facing the profile s. In order to understand calculus easier, we write the function $v^k(s)$ in one explicit way $v^k(s) = v^k(j_1, j_2, ..., j_n)$.The matrix $\mathbf{v}_{n,l}$ represents all points of the Cartesian product $\Pi_{k\in K} S_k$, [2],[9], [16] and [18]. The vector $\mathbf{v}^k(\mathbf{s})$ is the $k-$ column of $\mathbf{v}$.

If the mixed strategies are allowed, then we have:

$$\Delta(S_k) = \left\{ \mathbf{p}^k \in R^{l_k} : \sum_{j_k=1}^{l_k} p_{j_k}^k = 1 \right\}$$

the unit simplex of the mixed strategies of player $k \in K$, and $\mathbf{p}^k = (p_{j_k}^k)$ the probability vector. The set of profiles in mixed strategies is the polyhedron Δ with $\Delta = \Pi_{k\in K}\ \Delta(S_k)$, where $\mathbf{p} = (p_{j_1}^1, p_{j_2}^2 ..., p_{j_n}^n)$, and $\mathbf{p}^k = (p_1^k, p_2^k, ..., p_{l_n}^k)^T$. Using the Kronecker product, $\otimes$ it is possible to write[1]:

$$\mathbf{p} = \mathbf{p}^1 \otimes \mathbf{p}^2 \otimes ... \otimes \mathbf{p}^{k-1} \otimes \mathbf{p}^k \otimes \mathbf{p}^{k+1} \otimes ... \otimes \mathbf{p}^n \tag{1}$$

[1] We use bolt in order to represent vector or matrix.

$$\mathbf{p}^{(-k)}=\mathbf{p}^1\otimes\mathbf{p}^2\otimes...\otimes\mathbf{p}^{k-1}\otimes\mathbf{l}^k\otimes\mathbf{p}^{k+1}\otimes...\otimes\mathbf{p}^n \tag{2}$$

where

$$\mathbf{l}^k=(1,1,...,1)^T,\left[\mathbf{l}^k\right]_{l_k,1} \tag{3}$$

$$\mathbf{o}^k=(0,0,...,0)^T,\left[\mathbf{o}^k\right]_{l_k,1} \tag{4}$$

The $n-$ dimensional function $\overline{\mathbf{u}}:\Delta\rightarrow R^n$ is associated with every profile in mixed strategies and the vector of expected utilities

$$\overline{\mathbf{u}}(\mathbf{p})=\left(\overline{u^1}\left(\mathbf{p},\mathbf{v}(s)\right),..,\overline{u^n}\left(\mathbf{p},\mathbf{v}(s)\right)\right)^T \tag{5}$$

where $\overline{u^k}(\mathbf{p},\mathbf{v}(s))$ is the expected utility of the player k. Every $\overline{u^k_{j_k}}=\overline{u^k_{j_k}}(\mathbf{p}^{(-k)},\mathbf{v}(s))$ represents the expected utility of each player's strategy and the vector $\mathbf{u}^k$ is noted $\mathbf{u}^k=(\overline{u^k_1},\overline{u^k_2},...,\overline{u^k_n})^T$.

$$\overline{u_k}=\sum_{j_k=1}^{l_k}\overline{u^k_{j_k}}(\mathbf{p}^{(-k)},v(s))p^k_{j_k} \tag{6}$$

$$\overline{\mathbf{u}}=\mathbf{v}'\mathbf{p} \tag{7}$$

$$\mathbf{u}^k=\left(\mathbf{l}^k\otimes\mathbf{v}^k\right)\mathbf{p}^{(-k)} \tag{8}$$

The triplet $(K,\Delta,\overline{\mathbf{u}}(\mathbf{p}))$ designates the extension of the game Γ with the mixed strategies. We get Nash's equilibrium (the maximization of utility [2],[9], [16] and [18]) if and only if $\forall k$, $\mathbf{p}$, the inequality $\overline{u^k}(\mathbf{p}^*)\geq\overline{u^k}(\left(\mathbf{p}^k\right)^*,\mathbf{p}^{(-k)})$ is respected .

Another way to calculate Nash's equilibrium [9,16,18], is leveling the values of the expected prospective utilities of each strategy, when possible.

$$\overline{u^k_1}\left(\mathbf{p}^{(-k)},v(s)\right)=\overline{u^k_2}\left(\mathbf{p}^{(-k)},v(s)\right)=...=\overline{u^k_{j_k}}\left(\mathbf{p}^{(-k)},v(s)\right) \tag{9}$$

$$\sum_{j_k=1}^{l_k}p^k_{j_k}=1\qquad\forall k=1,...,n \tag{10}$$

$$\sigma^2_k=\sum_{j_k=1}^{l_k}\left(\overline{u^k_{j_k}}\left(\mathbf{p}^{(-k)},v(s)\right)-\overline{u}_k\right)^2p^k_{j_k}=0 \tag{11}$$

If the resulting system of equations doesn't result in $\left(\mathbf{p}^{(-k)}\right)^*$, then we propose the *Minimum Entropy Method* [11] and [12]. This method is expressed as $Min_{\mathbf{p}}\left(\sum_k H_k(\mathbf{p})\right)$, where $\sigma^2_k(\mathbf{p}^*)$ is the standard deviation and $H_k(\mathbf{p}^*)$ is the entropy of each player k.

$$\sigma^2_k(\mathbf{p}^*)\leq\sigma^2_k\left(\left(\mathbf{p}^k\right)^*,\mathbf{p}^{(-k)}\right) \tag{12}$$

or

$$H_k(\mathbf{p}^*)\leq H_k\left(\left(\mathbf{p}^k\right)^*,\mathbf{p}^{(-k)}\right) \tag{13}$$

2.1 Minimum Entropy Method

Theorem 1. *(Minimum Entropy Theorem). The game entropy is minimum only in mixed strategy Nash's equilibrium [11] and [12]. The entropy minimization program* $Min_{\mathbf{p}}\left(\sum_k H_k(\mathbf{p})\right)$*, is equal to the standard deviation minimization program* $Min_{\mathbf{p}}\left(\Pi_k \sigma_k(\mathbf{p})\right)$*, when* $\left(\overline{u_{j_k}^k}\right)$ *has gaussian density function or multinomial logit.*

According to Hayek, equilibrium refers to the order state or minimum entropy. The order state is opposite to entropy (disorder measure). There are some intellectual influences and historical events which inspired Hayek to develop the idea of a spontaneous order. Here, we present the technical tools needed in order to study the order state [11], [12], [17] and [17].

Case 2. *If the probability density of a variable* X *is normal:* $N(\mu_k, \sigma_k)$*, then its entropy is minimum for the minimum standard deviation* $(H_k)_{\min} \Leftrightarrow (\sigma_k)_{\min}$. $\forall k = 1, ..., n$.

Proof. Let the entropy function $H_k = -\int_{-\infty}^{+\infty} p(x)\ln(p(x))dx$ and $p(x)$ the normal density function.

$$H_k = -\int_{-\infty}^{+\infty} p(x)\ln(p(x))dx \tag{14}$$

$$H_k = -\int_{-\infty}^{+\infty} p(x)\left(\ln\left(\frac{1}{\sqrt{2\pi\sigma_k^2}}\right) - \left(\frac{x-\overline{u_k}}{\sqrt{2}\sigma_k}\right)^2\right)dx \tag{15}$$

developing the integral we have

$$H_k = \ln\left(\sqrt{2\pi}\sigma_k\right)\int_{-\infty}^{+\infty} p(x)dx \tag{16}$$

$$+\frac{1}{2\sigma_k^2}\int_{-\infty}^{+\infty}(x-\overline{u_k})^2 p(x)dx \tag{17}$$

$$= \ln\left(\sqrt{2\pi}\sigma_k\right) + \frac{\sigma_k^2}{2\sigma_k^2} = \left(\frac{1}{2} + \ln\sqrt{2\pi}\right) + \ln\sigma_k \tag{18}$$

For a game to $n-$ players, the total entropy can be written as follows:

$$\sum_{k=1}^{n} H_k = n\left(\frac{1}{2} + \ln\sqrt{2\pi}\right) + \ln(\Pi_{k=1}^n \sigma_k) \tag{19}$$

$$Min_{\mathbf{p}}\left(\sum_k H_k(\mathbf{p})\right) \Leftrightarrow Min_{\mathbf{p}}\left(\Pi_k \sigma_k(\mathbf{p})\right) \tag{20}$$

after making a few more calculations, it is possible to demonstrate that

$$Min_{\mathbf{p}}\left(\sum_{k=1}^{n}\sigma_k(\mathbf{p})\right) \Rightarrow Min_{\mathbf{p}}\left(\Pi_k\sigma_k(\mathbf{p})\right) \tag{21}$$

□

The entropy or measure of the disorder is directly proportional to the standard deviation or measure of uncertainty. Clausius, who discovered the entropy idea, presents it as both an evolutionary measure and as the characterization of reversible and irreversible processes [1], [3], [7] and [13].

Case 3. *If the probability function of* $\overline{u^k_{j_k}}(\mathbf{p}^{(-k)}, v(s))$ *is a multinomial logit of parameter* λ*, then its entropy is minimum and its standard deviation is minimum for* $\lambda \to \infty$.

Proof. Let $p^k_{j_k}$ the probability for $k \in K$, $j_k = 1, ..., l_k$

$$p^k_{j_k} = \frac{e^{\lambda u^k_{j_k}(\mathbf{p}^{-k})}}{\sum_{j_k=1}^{l_k} e^{\lambda u^k_{j_k}(\mathbf{p}^{-k})}} = \frac{e^{\lambda u^k_{j_k}(\mathbf{p}^{-k})}}{Z_k} \tag{22}$$

where $Z_k\left(\lambda, \mathbf{u}^k(\mathbf{p}^{-k})\right) = \sum_{j_k=1}^{l_k} e^{\lambda u^k_{j_k}(\mathbf{p}^{-k})}$ represents the partition utility function [1], [11] and [12].

The entropy $H_k(\mathbf{p}^k)$, expected utility $E\{u^k_{j_k}(\mathbf{p}^{-k})\} = \overline{u^k}(p)$, and variance $Var\{u^k_{j_k}(\mathbf{p}^{-k})\}$ will be different for each player k.

$$H_k(\mathbf{p}^k) = -\sum_{j_k=1}^{l_k} p^k_{j_k}\ln(p^k_{j_k}) \tag{23}$$

$$\overline{u^k}(p) = \sum_{j_k=1}^{l_k} p^k_{j_k}\, u^k_{j_k} \tag{24}$$

$$Var\left\{u^k_{j_k}(\mathbf{p}^{-k})\right\} = \sum_{j_k=1}^{l_k} p^k_{j_k}(u^k_{j_k} - \overline{u^k})^2 \tag{25}$$

Using the explicit form of $p^k_{j_k}$, we obtain the entropy, the expected utility and the variance [11] and [12]:

$$H_k(\mathbf{p}^k) = \ln(Z_k) - \lambda\overline{u^k}(p) \tag{26}$$

$$\overline{u^k}(p) = \frac{\partial \ln(Z_k)}{\partial\lambda} \tag{27}$$

$$\frac{\partial\overline{u^k(p)}}{\partial\lambda} = Var\{u^k_{j_k}(\mathbf{p}^{-k})\} = \sigma_k^2 \tag{28}$$

The equation $\frac{\partial H_k(\mathbf{p}^k)}{\partial \lambda} = -\lambda\sigma_k^2$ can be obtained using the last seven equations; it explains that, when entropy diminishes, rationality increases.

The rationality increases from an initial value of zero when the entropy arrives at its maximum value, and drops to its minimum value when the rationality spreads toward the infinite value: $Lim_{\lambda\to\infty} H_k(\mathbf{p}^k(\lambda)) = \min(H_k)$

The standard deviation is minimum in Nash's equilibria [2] and [16].

If rationality increases, then Nash's equilibria can be reached when the rationality spreads to its infinite value: $Lim_{\lambda\to\infty}\sigma_k(\lambda) = 0$.

Using the logical chain that has just been demonstrated, we can conclude that the entropy diminishes when the standard deviation diminishes:

$$(H_k(\mathbf{p}) = (H_k(\mathbf{p}))_{\min}) \Leftrightarrow (\sigma_k(\lambda) = (\sigma_k)_{\min}) \tag{29}$$

$$Min_{\mathbf{p}}\left(\sum_k H_k(\mathbf{p})\right) \Leftrightarrow Min_{\mathbf{p}}\left(\Pi_k\sigma_k(\mathbf{p})\right) \tag{30}$$

after making a few more calculations, it is possible to demonstrate that

$$Min_{\mathbf{p}}\left(\sum_{k=1}^{n}\sigma_k(\mathbf{p})\right) \Rightarrow Min_{\mathbf{p}}\left(\Pi_k\sigma_k(\mathbf{p})\right) \tag{31}$$

□

Remark 1. The entropy H_k for gaussian probability density and multinomial logit is written as $\left(\frac{1}{2} + \ln\sqrt{2\pi}\right) + \ln\sigma_k$ and $\ln(\sum_{j_k=1}^{l_k} e^{\lambda\overline{u_{j_k}^k}}) - \lambda\overline{u^k}$

Case 4. *The special case of Minimum Entropy is when* $\sigma_k^2 = 0$ *and the utility function value of each strategy* $\overline{u_{j_k}^k}(\mathbf{p}^{(-k)}, v(s)) = \overline{u^k}$, *is the same* $\forall j_k, \forall k$.

3 Application of the Model

Let $\Gamma = (K, S, v)$ be a game to 3−players, with K the set of players $k = 1, 2, 3$. The finite set S_k of cardinality $l_k \in N$ is the set of pure strategies of each player where $k \in K$, $s_{kj_k} \in S_k$, $j_k = 1, 2, 3$ and $S = \Pi_k S_k$ represents the set of pure strategy profiles which have $s \in S$ as an element of that set, and $l = 3{*}3{*}3 = 27$ represents the cardinality of S. The vectorial function $\mathbf{v} : S \to R^3$ associates every profile $s \in S$ with the vector of utilities $\mathbf{v}(s) = (v^1(s), ..., v^3(s))$, where $v^k(s)$ designates the utility of the player k using the profile s. In order to use calculus easier, we write the function $v^k(s)$ in one explicit way: $v^k(s) = v^k(j_1, j_2, ..., j_n)$.The matrix $\mathbf{v}_{3,27}$ represents all points of the Cartesian product $\Pi_K S_k$ see in (Table 1). The vector $\mathbf{v}^k(\mathbf{s})$ is the k- column of $\mathbf{v}$. The graphic representation of the 3-players game (Figure 1).

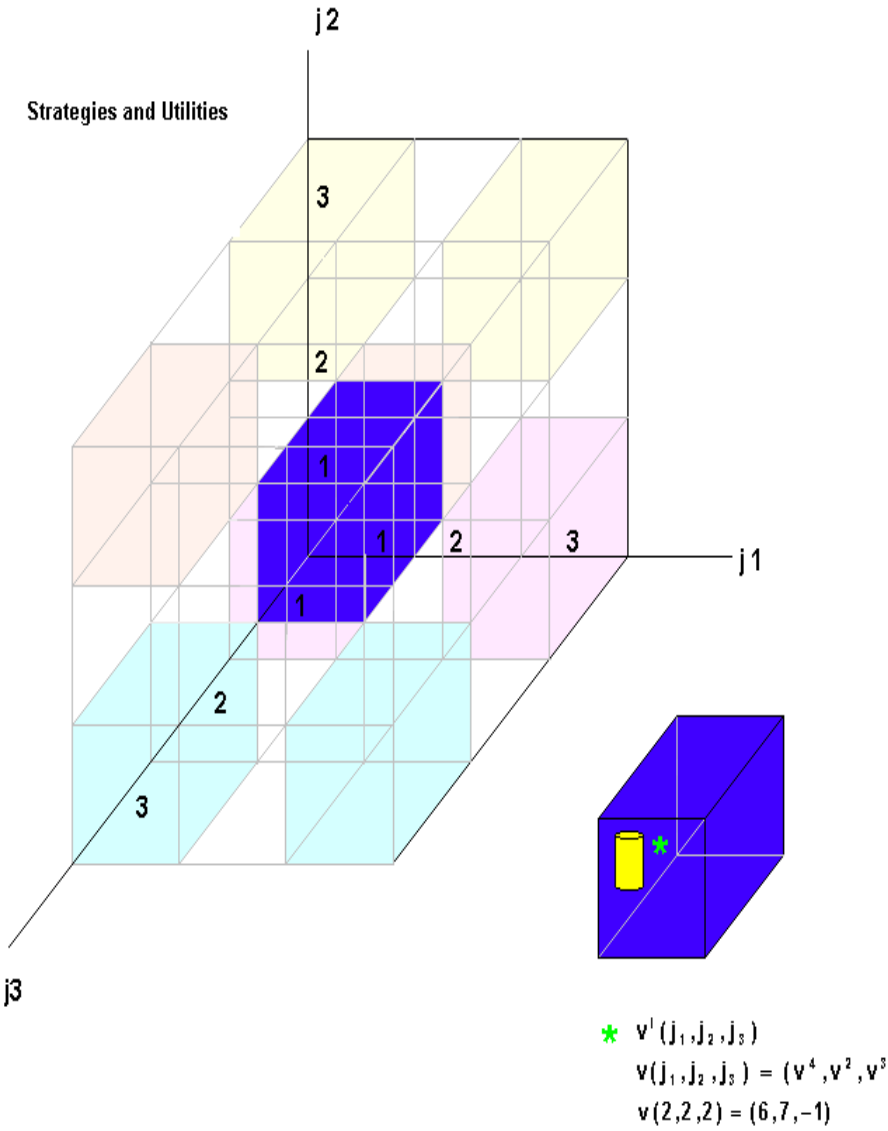


Fig. 1. 3-players game strategies

Table 1. Minimum Entropy: The Game of Stone-Paper-Scissors.
$\min_{\mathrm{p}}(H_1 + H_2 + H_2) \Leftrightarrow Min_{\mathrm{p}}\,(\sigma_1(\mathbf{p})\sigma_2(\mathbf{p})\sigma_3(\mathbf{p}))$

Nash Utilities and Standard Deviations

$\mathrm{Min_p}$	$(\sigma^1\sigma^2\sigma^3)$	=	0

u^1	u^2	u^3
0.5500	0.5500	0.5500
σ^1	σ^2	σ^3
0.0000	0.0000	0.0000
H_1	H_2	H_3
0.0353	0.0353	0.0353

Nash Equilibria: Probabilities, Utilities

Player 1			**Player 2**			**Player 3**		
p^1_1	p^1_2	p^1_3	p^2_1	p^2_2	p^2_3	p^3_1	p^3_2	p^3_3
0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3300	0.3300	0.3300
u^1_1	u^1_2	u^1_3	u^2_1	u^2_2	u^2_3	u^3_1	u^3_2	u^3_3
0.5500	0.5500	0.5500	0.5500	0.5500	0.5500	0.5556	0.5556	0.5556
$p^1_1 u^1_1$	$p^1_2 u^1_2$	$p^1_3 u^1_3$	$p^2_1 u^2_1$	$p^2_2 u^2_2$	$p^2_3 u^2_3$	$p^3_1 u^3_1$	$p^3_2 u^3_2$	$p^3_3 u^3_3$
0.1833	0.1833	0.1833	0.1833	0.1833	0.1833	0.1833	0.1833	0.1833

Kroneker Products

j_1	j_2	j_3	p^1_{j1}	p^2_{j2}	p^3_{j3}	$p^1_{j1}p^2_{j2}$	$p^1_{j1}p^3_{j3}$	$p^2_{j2}p^3_{j3}$	$u^1(j_1,j_2,j_3)$	$u^2(j_1,j_2,j_3)$	$u^3(j_1,j_2,j_3)$	$v^1(j_1,j_2,j_3)$	$v^2(j_1,j_2,j_3)$	$v^3(j_1,j_2,j_3)$
stone	stone	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.0000	0.0000	0	0	0
stone	stone	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.0000	0.1111	0	0	1
stone	stone	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.0000	1	1	0
stone	paper	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.1100	0.0000	0	1	0
stone	paper	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.1100	0.1111	0	1	1
stone	paper	scsisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.1111	1	1	1
stone	sccisor	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.0000	0.1111	1	0	1
stone	sccisor	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.1111	1	1	1
stone	sccisor	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.0000	0.0000	1	0	0
paper	stone	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.0000	0.0000	1	0	0
paper	stone	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.0000	0.1111	1	0	1
paper	stone	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.1111	1	1	1
paper	paper	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.0000	1	1	0
paper	paper	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.0000	0.0000	0	0	0
paper	paper	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.0000	0.1111	0	0	1
paper	scsisor	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.1111	1	1	1
paper	sccisor	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.1100	0.0000	0	1	0
paper	sccisor	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.1100	0.1111	0	1	1
sccisor	stone	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.1100	0.1111	0	1	1
sccisor	stone	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.1111	1	1	1
sccisor	stone	scsisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.1100	0.0000	0	1	0
sccisor	paper	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.1111	1	1	1
sccisor	paper	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.0000	0.0000	1	0	0
sccisor	paper	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.0000	0.1111	1	0	1
sccisor	scsisor	stone	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.0000	0.1111	0	0	1
sccisor	sccisor	paper	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.1100	0.1100	0.0000	1	1	0
sccisor	sccisor	sccisor	0.3333	0.3333	0.3300	0.1111	0.1100	0.1100	0.0000	0.0000	0.0000	0	0	0

4 Conclusion

1. An immediate application of quantum games in the science of economics is related to the principal-agent relationship. Specifically, we can use m to represent types of agents in the adverse selection model. In moral risk, quantum games could be used as a discrete or continuous set of efforts.
2. In this paper we have demonstrated that the Nash-Hayek equilibrium opens new doors so that entropy in game theory can be used. Remembering that the primary way to prove Nash's equilibria is through utility maximization, we can affirm that human behavior arbitrates between these two stochastic-utility (benefits) $U(p(\mathbf{x}))$ and entropy (risk or disorder) $H(p(\mathbf{x}))$ elements. Accepting that the *stochastic-utility/entropy* $\left[\frac{U(p(\mathbf{x}))}{H(p(\mathbf{x}))}\right]$ relationship is equivalent to the well-known *benefits/cost/ratio,* we present a new way to calculate equilibria: $Max_{\mathbf{x}}\left(\frac{U(p(\mathbf{x}))}{H(p(\mathbf{x}))}\right)$, where $p(x)$ probability function and $\mathbf{x} = (x_1, x_2, ..., x_n)$ exogenous variables.
3. This paper, which uses Kronecker product $\otimes$, represents an easy, new formalization of game $(K, \Delta, \overline{\mathbf{u}}(\mathbf{p}))$,which extends the game Γ to the mixed strategies.

Acknowledgment
This study and analysis was made possible through the generous financial support of TODO1 SERVICES INC and PETROECUADOR.

The author is grateful to: Jorge Moyano from Diners Club International, Fernando Pozo Banco Pichincha Ecuador, Gonzalo Pozo Todo1 Services-Miami, the 8th Annual Conference in Computing in Economics and Finance, July 2002, (the assistants), AIX France, the assistants to 10th Annual Conference of the SouthEast SAS User Group USA,(the assistants). M2002, the 5th annual data mining technology conference SAS 2002 USA (the assistants).

The opinions and errors are solely my responsability.

The author also thanks his PhD advisor, Prof. Jean Louis Rulliere (GATE director, Universite Lumiere Lyon II France).

The author can be reached at the following fax number (593) 22524766, email: ejimenezecu@yahoo.com

References

1. Bacry H. *Introduction aux concepts de la Mecanique Statistique.* Ellipsis, París,(1991) 100-300.
2. Binmore K. *Fun and Games.* D.C.Health and Company,(1992) 50-300.
3. Cohen-Tannoudji C., Diu B., Laloe F. *Quantum Mechanics.* Hermann, París,(1977) 160-200.
4. Eisert J., Scheel S., Plenio M.B. *On the Impossibility of distilling Gaussian States with Gaussian Operations.* Mimeo, Imperial College of London, (2002).
5. Eisert J. and Wilkens M. *Quantum Games.* Mimeo, University of Potsdam of Germany, (2002).

6. Eisert J., Wilkens M., Lewenstein M. *Quantum Games and Quantum Strategies*, Working Paper, University of Postdam, Germany, (2001).
7. Feynman R. *Le cours de Physique de Feynman.* DUNOT, (1998) 100-250.
8. Frieden B. *Information Sciences*, *Probability, Statistical, Optics and Data Testing*Springer-Verlag (2001).
9. Fudenberg D., Tirole J. *Game Theory*, Massachusetts, The Mit Press, (1991) 10-200.
10. Jiangfeng, Du and all . *Experimental Realizations of Quantum Games on a Quantum Computer*, Mimeo, University of Science and Technology of China, (2002).
11. Jimenez, E. (1999). Aplicacion de Redes Neuronales Artificiales en Sismica del Pozo, PETROECUADOR, Quito - Ecuador, (1999) 1-100.
12. Jimenez E. *Unified Game Theory (unpublished)*, Paper presented in 8th CONFERENCE IN COMPUTING IN ECONOMICS AND FINANCE AIX (2002).
13. Jimenez E. *Preemption and Attrition in Credit/Debit Cards Inventives: Models and Experiments (Forcoming in Proceedings of 2003 International Conference on Computational Intelligence for Financial Engineering (CIFEr2003))*, Hong Kong, (2003).
14. Landau. L (1978). *Mecanica Cuántica*, Editorial Reverté, (1978) 1-350.
15. Meyer D. *Quantum Strategies.* Phys. Rev. Lett. 82 (1999) 1052-1055, (1999).
16. Meyer D. *Quantum Games and Quantum Algorithms*, To appear in the AMS Contemporany Mathematics volume: Quantum Computation and Quantum Information Science, (2000).
17. Myerson R. *Game Theory Analysis of Conflict.*, Massachusetts, Harvard University Press, (1991) 1-200.
18. Neyman A. *Repeated Games with Bounded Entropy.*, Games and Economic Behavior, Ideal, 30,(2000) 228-247.
19. Rasmusen E. *Games and Economic Information: An Introduction to Game Theory.*, Oxfort: Basil Blackwell, (1989) 100-150.
20. SENKO Corporation. *Quantum Computer Simulation*, SENKO Corporation, Yokohama Japan, (1999).
21. Weibul J.W. *Evolutionary Game Theory.*, The MIT press, (1995) 1-120.

Distributed Solution of High-Order Compact Difference Schemes for Multidimensional Convection-Diffusion Equations

Matthew F. Dixon and Kenneth Tan

Heuchera Technologies UK,
{mdixon,cjtan}@optimanumerics.com,
http://www.optimanumerics.com

Abstract. The performance of a high-order compact (HOC) discretisation of a multidimensional time dependent convection-diffusion equation on a distributed computer cluster is demonstrated. We consider a D dimensional fourth-order compact difference scheme and concurrently solve the implicit equation with GMRES(m). The performance of GMRES(m) on distributed computers is limited by the inter-processor communication required by the matrix-vector multiplication. It is shown that the compact scheme requires approximately half the number of communications as a non-compact difference scheme of the same order of truncation error. As the dimensionality is increased, the ratio of computation that can be overlapped with communication also increases. CPU times and parallel efficiency graphs for single time step approximation of up to 7D HOC coarse approximations demonstrate improved parallel scalability over non-compact difference schemes.

1 Introduction

There are many examples of high dimensional numerical methods for which stochastic simulation provides an inadequate approximation solution. These include multifactor American Option pricing, Ab-initio Schrödringer and Boltzmann equations. In the wider sense, this includes any collective interdependent system of partial differential equations with complex geometries or singularities.

It is generally accepted that standard implicit finite difference methods can solve up to three-dimensional partial differential equations with flexibility and efficiency [2, 12]. The exponential growth of computational complexity and memory requirements with dimensionality, however, prevent their application to higher dimensional problems.

The high-order compact (HOC) implicit difference scheme is a numerically stable method for approximating D dimensional convection-diffusion problems to fourth-order accuracy [3]. The most computationally complex stage of the implicit approximation is the inversion of the coefficient matrix for the finite

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 226–235, 2003.
© Springer-Verlag Berlin Heidelberg 2003

difference operators at each time step. For fixed-boundary linear problems, the inverse matrix can most efficiently be approximated using Krylov subspace methods when the matrix is non-singular, positive definite and, in the absence of a preconditioner, its condition number is small. The requirement for both high dimensionality and distributed parallel solution presents a new perspective on high-order finite difference schemes. Specifically, it is the condition number, sparsity and bandwidth of the finite difference coefficient matrix which are most critical to satisfying this requirement using distributed Krylov subspace solvers.

This paper will specifically investigate the effect of the bandwidth and sparsity of high order finite difference coefficient matrices for a class of Convection-Diffusion equations solved on distributed computers. The effect of the condition number of a finite difference coefficient matrix for a particular equation in the Convection-Diffusion class, the Black-Scholes equation, is investigated in [3]. As a benchmark for the HOC scheme, we will consider the central difference (CD) and a simple non-compact fourth-order scheme, referred to herein as the HO1 scheme. The performance limitations of distributed GMRES(m) will be discussed in Section 4 and their comparative effect on each high-order scheme assessed. A parallel model will then be implemented using `PETSc` [8] and algorithm performance data will be presented in Section 5 for the multifactor Option pricing example on a 16 node cluster. Specifically, cpu times for the CD, HO1 and HOC are compared in serial and in parallel. Finally the parallel scalability of the GMRES(m) is compared between the non-compact and compact high-order scheme with increasing dimensionality.

2 The Transient Multidimensional Convection-Diffusion Equation

2.1 The Continuous Problem

Consider a bounded domain $\Omega \subset \mathbb{R}^D$ on which a diffusion tensor (including cross-terms) $a : \Omega \to \mathbb{R}^{D\times D}$ and a velocity function $b : \Omega \to \mathbb{R}^D$ are defined. The transient convection-diffusion problem is

$$\frac{\partial \phi}{\partial t} - \nabla \cdot (a\nabla\phi + b\phi) = f(\phi), \tag{1}$$

where the source function $f(\phi) : \Omega \to \mathbb{R}^D$ is typically continuous. The twice differentiable solution ϕ to Equation 1 also satisfies the boundary conditions on $\partial\Omega$ such that

$$\alpha(\phi) + \beta\frac{\partial \phi}{\partial n_d} + \gamma\frac{\partial^2 \phi}{\partial x_d} = g_{\partial\Omega}, \tag{2}$$

where x_d is the dependent variable in dimension d and n_d the unit vector normal to the boundary of variable x_d. To apply the theory in this paper, a log-transformed multi-factor form of the celebrated Black-Scholes equation for financial Option pricing under constant volatility is used as defined in [11, Chapter 3]. This produces a matrix for which the interior domain is spatially

invariant, simplifying the numerical analysis and potentially enabling the use of run-length encoding of the coefficient matrix to further reduce memory requirements. The variables x_d representing financial assets are typically correlated and exhibit cross diffusion, complicating the numerical scheme. The cartesian domain $\Omega \subset \mathbb{R}^D$ can be transformed so that the diffusion tensors act along the principal axes, eliminating the cross-derivatives [3].

3 High-Order Finite Difference Schemes

High-order finite difference scheme are used here, in the absence of singularities, to reduce the number of nodes for a fixed order of spatial truncation error.

Definition 1. *Let $\hat{n}_d$ and $\hat{n}_d^{HO}$ be the number of grid points along the axis in d for the Central Difference (CD) and the $O(n^\Lambda)$ High-Order (HO) scheme respectively, where $\Lambda > 2$. Let $\hat{n}_d = \hat{n}$ for all $d \in [1, D]$ on Ω.*

It can be shown that the order of reduction of the number of nodes of a $O(\Delta S^\Lambda)$ accurate scheme compared with the standard second order accurate (CD) scheme at a single time step is $O(n^{\frac{2D}{\Lambda}})$ (See [4, p. 23] for a proof). We consider two fourth-order accurate schemes; an HO1 scheme, derived by extending the number of points in each axis of the stencil, and a HOC scheme described below.

3.1 HOC Stencil

The HOC scheme is derived by directly truncating the Taylor series expansion of each discrete operator to higher orders [10]. The number of nodes in the stencil is larger than that of the CD scheme. This is less significant, however, than the increase in the order of truncation error and improved data locality in comparison with the other schemes considered. A full description of the fourth-order compact scheme for the diagonalised log-transformed two-factor Black-Scholes equation $\hat{L}(\xi, \eta, t)$ is defined in [3].

The HOC stencil contains diagonal terms originating from mixed derivates in the truncation error leading to a semi-implicit scheme

$$Ax_{t+1} = Cx_t,$$

where A and C are large, irregular and non-symmetric coefficient matrices at time $t + 1$ and t respectively, and x_{t+1} is the unknown vector best determined using GMRES(m) to invert A.

4 Distributed GMRES(m)

The GMRES algorithm [9] consists of two stages (i) construction of an orthonormal basis and (ii) projection of this basis on to a Krylov subspace $\mathbb{K}$. The method

is classically implemented using the Arnoldi process. This a modified Gram-Schmidt method specifically for Krylov subspaces, designed to orthonormalise the J bases of $\mathbb{K}$. Each new orthonormal base, referred to as a dimension or a recurrence i, is stored as a basis vector $A^i r_0$. GMRES is usually restarted after m iterations to reduce memory requirements and for this reason is referred to as GMRES(m).

4.1 Performance Determining Factors

In addition to the affect of the condition number on the convergence properties, the number of non-zero elements in the matrix determine the time taken to perform each iteration of the GMRES(m) method. Although both high-order schemes use the mesh properties to achieve the same order of accuracy, diagonal elements in the HOC stencil increase the sparse matrix density. It is shown in Section 5 that the comparative sparsity of the coefficient matrices is, however, less significant at higher dimensions than the data locality when solving in parallel on a high latency network.

The modified Gram-Schmidt orthogonalisation consists of BLAS level 1 and 2 kernels. Of these the Matrix-vector multiplication, `MatVec` and, depending on the convergence properties, the inner-vector product limit parallel scalability on distributed computers, as reported in [5, 1, 6]. The matrix-vector multiplication is considered in more detail. Data dependency analysis [4, pp. 32-34] suggests that the optimal way to parallelise this kernel is to partition the matrix $A \in \mathbb{R}^{n \times n}$ into submatrices over n_p processors as shown in figure 1.



Fig. 1. Vector element dependencies in a parallel matrix-vector multiplication, shown here with a single partition.

The granularity of `MatVec` varies at any stage of the operation depending on the (i) number of rows in the local submatrix $\lceil \frac{n}{n_p} \rceil$ (ii) the proximity of each matrix element from the processor submatrix boundary (iii) the number of processors holding vector elements (referred to as the *span*) required by the stencil for matrix-vector multiplication and (iv) the method of communicating the remote vector elements. Put succinctly, the wider the bandwidth of the stencil - the greater the number of cross domain vector element fetch operations and the smaller the number of computations that can be overlapped with communication [7].

4.2 Scalability Analysis

Definition 2 (Parallel Model). *Consider a SPMD parallel model in which each processor p stores an index of references to remote data elements in distributed memory required by the submatrix on p, A_p. All remote dependent elements are each asynchronously fetched in a single communication from each processor in turn with a network latency of γ_L^x and bandwidth $\frac{1}{\beta}$, where $0 < \beta << \gamma_L^x$.*

Using this model, the cost of communication for all processors, $\gamma_{sync}^{A\times x}$ is $2n_p(P\gamma_L^x + w\beta)$, where P is the number of submatrices spanned by the stencil $\mathbb{S}$. The speedup of the modified Gram-Schmidt algorithm is defined for a single iteration, where $\tau = n(\alpha+\gamma)(2n(w)+1+2m)$ and α is a scalar dependent on hardware performance,

$$S_p(n_p) = \frac{n_p\tau}{\tau + n_p^2(\alpha + 2\gamma_L^x(m+P) + 2\beta(m+w))},$$

and the efficiency of this method is

$$\epsilon = \frac{\tau}{\tau + n_p^2(\alpha + 2\gamma_L^x(m+P) + 2\beta(m+w))}. \tag{3}$$

The comparative performance of this model is dependent on the stencil half-bandwidth w, which determines P and thus the number of inter-processor communications.

Definition 3. *Define $\mathbb{P}^{HO1}$ and $\mathbb{P}^{HOC}$ as the data dependency spans of the stencils $\mathbb{S}_{HO1}$ and $\mathbb{S}_{HOC}$ respectively forming the Matrix $A \in \mathbb{R}^{n\times n}$ where $n = \hat{n}^D$.*

Lemma 1. *The number of dependent processor domains required for* `MatVec` *of a HOC coefficient matrix P_{HOC} is approximately half that required for the HO1 matrix.*

Proof. Considering the HO1 node with the greatest offset $\mathbb{O}_i$ for a uniform grid

$$\mathbb{O} = 2\sum_{d=1}^{D-1} \hat{n}_d = 2\hat{n}^{D-1},$$

where the width of the uniform submatrix is $w_p = \frac{\hat{n}^D}{n_p}$, leading to $\mathbb{P}_0^{HO1} = 2\lceil\frac{n_p}{\hat{n}}\rceil$. The HOC node with the greatest offset $\mathbb{O}_{ij} = \hat{n}^{D-1} + \hat{n}^{D-2}$, where the ratio of the spans

$$\frac{P_{HOC}}{P_{HO1}} = \frac{\lceil\frac{n_p}{\hat{n}}\rceil(1+\frac{1}{\hat{n}})}{2\lceil\frac{n_p}{\hat{n}}\rceil} \approx \frac{1}{2}.$$

Table 1. Parallel scalability of the modified Gram-Schmidt method over m recurrences for the compact and non-compact high order schemes.

Scheme	Serial(τ)	Efficiency
HO1	$4n(2D+m)\alpha$	$\frac{4n(2D+m)\alpha}{4n(2D+m)\alpha+n_p^2(2\gamma_L^x(m+2\lceil\frac{n_p}{\hat{n}}\rceil)+2\beta(m+2\hat{n}^{D-1}))}$
HOC	$n(D(D+3)+2m))\alpha$	$\frac{n(D(D+3)+2m)\alpha}{n(D(D+3)+2m)\alpha+n_p^2(2\gamma_L^x(m+\lceil\frac{n_p}{\hat{n}}\rceil)+2\beta(m+\hat{n}^{D-1}))}$

The expressions for P can be substituted into equation 3 for each high-order scheme as shown in Table 1. The efficiency measures the ratio of speed-up of one iteration of the modified Gram-Schmidt method with m recurrences when using n_p processors, to n_p. This provides a standard measure of parallel scalability with communication overhead.

Assuming that the total communication cost $\alpha = \gamma,\ \beta \approx 0$, the comparative efficiency ratio of each high-order method is then

$$\lim_{n_p \to \infty} \frac{\epsilon_{HO1}}{\epsilon_{HOC}} \leq \frac{2(2D+m)}{D(D+3)+2m} \leq \frac{2m}{D^2+2m} \leq 1. \tag{4}$$

This efficiency ratio decreases with dimensionality suggesting that the HOC scheme is more scalable.

For practical purposes, the network bandwidth can not be considered infinite. We consider the effect of a typical range of network bandwidth in Figure 2 where $n_p = 16,\ m = 100,\ \hat{n} = 10,\ \gamma_L^x = 10e^{-4}$ and $\beta = \frac{1}{7.5\times 10e^8}$. As the dimensionality is increased, the relative significance of network bandwidth is less than the latency. This means that the additional (stencil diagonal) elements required by the HOC matrix from each processor domain are less significant than P, which is higher for the HO1 matrix. Consequently, the efficiency ratio tends to that predicted in equation 4.

4.3 Computational Overlap

A number of general performance optimisation methods can be applied to the Gram-Schmidt method [7]. A simple and effective approach, that is further considered, is to *hide* the communication cost by overlapping computation. `PETSc` provides the data structure `Mat_MPIAIJ` which defines a local square submatrix of non-zero diagonal entries and a rectangular submatrix of $\sum n(w_i^o)$ off-diagonal matrix elements with remote dependency for each row i of the submatrix. It also provides a local vector of dimension $2w$ used to receive scatters of non-local vector elements during the matrix-vector operation. Only the vector elements needed by the off-diagonal local submatrix will be scattered to the processor.

This data structure is exploited by performing all local matrix-vector products concurrently whilst fetching the remote vector elements. For any row i on

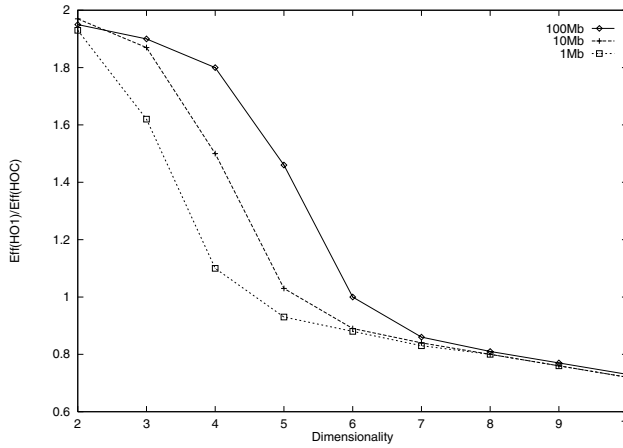


Fig. 2. Theoretical effect of network bandwidth on $\frac{\varepsilon_{HO1}}{\varepsilon_{HOC}}$.

the submatrix A_p, where i denotes the distance from the first or last row, there are remote vector element dependencies when $w \leq i \leq \frac{1}{2}\lceil \frac{n}{n_p} \rceil$ and R_c products, equating to $1+(\frac{1}{2}\lceil \frac{\hat{n}^D}{n_p} \rceil - w)$ (including the central diagonal), which can be overlapped with communication. Since w is $O(\hat{n}^{D-1})$ it follows that R_c increases with dimensionality, improving the parallel efficiency.

A comparison of data dependency distribution across the HO1 and HOC $3D$ submatrices is shown in figure 3. We deduce from the stencil widths of the fourth-order HO1 and HOC scheme that the ratio of R_c^{HOC} to R_c^{HO1} is less than or equal to 2, as confirmed by figure 3. Thus not only is the computation in HO1 matrix-vector multiplication dependent on more processors but consequently the minimum region of overlap is half that of HOC.

5 Numerical Experiments

The results in this Section were obtained using 4 400MHz quad-processor SUN `Ultra-Sparc III` workstations with 4GB of shared memory connected to a 100Mbps token ring Ethernet. Average latency γ_l of messages sent between processors on the same workstation was 117.85 (μs) and 494.75 (μs) between distributed processors. Average network bandwidth($\frac{b}{\mu s}$) B_w was 89.39 internally and 11.70 externally. The parallel HOC generator and solver was developed using `PETSc` 2.1.3 using compressed sparse row (CSR) format and `MPICH` 1.2.4.

Unless otherwise stated, all experiments used 10 nodes per spatial axis and a time step of 0.1. The rate of growth $r = 0.05$, asset correlation $\rho = 0.5$, strike price $K = 10$ and dividends have not been considered. The asset range $x_d = [0, 1]$ and time horizon $T = [0, 1]$.

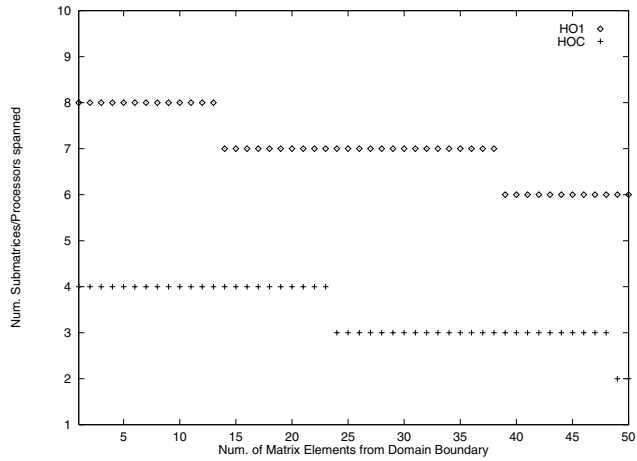


Fig. 3. Comparison of the number of submatrices spanned, P, by a 3D HO1 and HOC stencil at any point in the submatrix.

Table 2 compares the performance of GMRES(m) on the CD, HO1 and HOC matrices using one and twelve processors. The CD scheme is only second-order accurate yet is outperformed by the fourth-order compact scheme, because in this case, the condition number is higher than that of the HOC scheme as indicated by the number of iterations. The `MatVec` is more significant when the number of iterations is low and in general does not scale in parallel as efficiently as the vector updates `VecAXPY` and inner-products `VecDot` kernels.

Table 2. CPU times(s) for components of GMRES(m) on the CD, HO1 and HOC matrices where $\sigma = 0.5$.

Operation	CD		HO1		HOC	
n_p	1	12	1	12	1	12
MatVec	6.324	1.394	9.960	2.526	6.985	1.029
VecAXPY	9.085	0.568	12.627	0.885	2.809	0.390
VecDot	7.397	1.781	10.382	2.523	2.376	0.947
SLESSolve	25.578	3.439	36.744	5.348	13.021	2.50
KSPGMRESOthorg	15.969	2.189	22.415	3.131	4.91	1.06
MPI Scatter	-	0.923	-	1.481	-	0.375
Iterations	10	10	12	12	6	6
Total	38.57	6.064	52.87	8.496	27.31	5.772

Figure 4 compares the parallel scalability of the high-order schemes using Dirichlet conditions for a seven dimensional homogeneous system. Although the

HOC method is more consistent and more scalable, it does not match the estimated efficiency ratios. In this case, the reason is that the HO1 stencil intercepts the boundaries more often than the HOC method resulting in more single scalar stencil terms in the matrix. In accordance with this observation, derivative boundary conditions increase the efficiency ratios between each scheme. Parallel efficiency varies as the processor pool increases. This is determined by the effect of the processor partition on the span of the matrix-vector product and the combination of processors deployed.

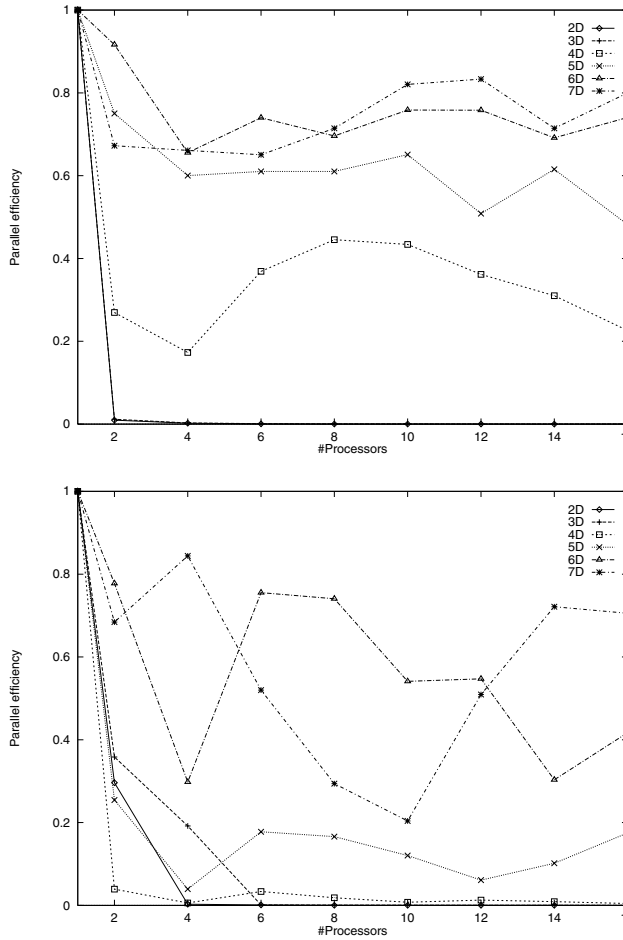


Fig. 4. Parallel efficiency of the homogeneous HOC (above) and HO1 schemes with Dirichlet boundary conditions, $\sigma = 0.5$, $\hat{n} = 10$.

6 Conclusion

In this paper we have demonstrated the superior performance of the fourth-order compact approximation of the d dimensional time dependent convection-diffusion scheme over a non-compact approximation on a distributed computer cluster. It was shown in Section 4 that the compact matrix improves the performance of the rate determining step of the distributed GMRES(m) method, the matrix-vector product. The remote vector elements required for the matrix-vector multiplication of each submatrix A_p are fetched from each processor p in one communication. The number of dependent processors of the HOC matrix-vector multiplication is half that for the HO1 matrix. Consequently, not only is the communication overhead reduced but also the ratio of computation that can be overlapped is increased. This effect becomes more apparent with the spatial dimensionality as demonstrated by results of upto a 7D HOC and HO1 (non-compact) scheme solved at a single time step with GMRES(m) on 16 homogeneous workstations.

References

[1] V. Kumar A. Gupta and A.H. Sameh, *Performance and scalability of preconditioned conjugate gradient methods on parallel computers*, Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing, 1993, pp. 664–674.

[2] J. Barraquand and D. Martineau, *Numerical valuation of high dimensional multivariate American securities*, Journal of Financial and Quantitative Analysis (1995), no. 30, 383–405.

[3] Matthew F. Dixon, *A high-order compact finite difference scheme for pricing multifactor options*, Heuchera Technologies Internal Report, HTNMMFD02.

[4] ———, *Parallel solution of high-order finite difference schemes for pricing multidimensional American options*, M.Sc. Thesis, Department of Computer Science, Reading University, 2002.

[5] M. Heath J. Demmel and H. van der Vorst, *Parallel numerical linear algebra*, Acta Numerica 1993, Cambridge University Press, Cambridge, UK, 1993, pp. 111–198.

[6] C. Kamath and A.H. Sameh, *The preconditioned conjugate gradient algorithm on a multiprocessor*, Advances in Computer Methods for Partial Differential Equations (1984), no. November.

[7] S. Balay, W. Gropp, L. McInnes and B. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, Modern Software Tools in Scientific Computing, E. Arge, A.M. Bruaset, and H.P. Langtangen, eds. (1997), 163–202.

[8] S. Balay, W.D. Gropp, L.C. McInnes and B.F. Smith, *Petsc users manual*, Tech. Report ANL-95/11 - Revision 2.1.1, Argonne National Laboratory, 2001.

[9] Y. Saad and M. Schultz, *GMRES a generalised minimum residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing **7** (1986), 856–869.

[10] W.F. Spotz, *High-order compact finite difference schemes for computational mechanics*, Ph.D. thesis, University of Texas, 1995.

[11] D. Tavella and C. Randall, *Pricing financial instruments*, Wiley, 2000.

[12] P. Wilmott, *The theory and practice of financial engineering*, Wiley, 1998.

Neural Network for Modeling Financial Time Series: A New Approach

Chokri Slim[1] and Abdelwahed Trabelsi [2]

[1] University of Tunis, Bestmod Laboratory, 41rue de la libert
2000 Le Bardo, Tunisia
chokri.slim@iscae.rnu.tn
[2] University of Tunis, Bestmod Laboratory, 41rue de la libert
2000 Le Bardo, Tunisia
abdel.trabelsi@isg.rnu.tn

Abstract. From Takens Theorem, we know that we can characterize the autonomous part of a dynamical system by a sequence of observations. If we want to force a neural net to learn this dynamics, we can use a sequence of time horizons as output. But in normal network structures these different outputs are learned nearly independently. By a new net architecture we allow additional information flows between the different outputs and as a sequence we get a better representation of the underlying dynamical system. The net is based on multilayer feed forward architecture. The model can then be converted to its equivalent nonlinear state space representation. Using this state space form, a Coupled Neural Net algorithm based on Extended Kalman Filter is derived to estimate the state. We analyze the net of a chaotic time series (Logistic map) by using the dynamical invariant that characterizes the attractor, the largest lyaponov exponent. A detailed step by step description of the methodology is presented to facilitate the use of this new method. The pertinence of this model is discussed from the Tunisian Stock Exchange database.

1 Introduction

The most common assumption in analyzing stock price is that successive price changes are independent of the history of previous price changes. In ex-post analyses based on this assumption, price fluctuations are assumed to be purely stochastic, generated by one common noise processes. Until recently, empirical support for this form of the efficient markets hypothesis, in the form of numerous published statistical tests, has been compelling. For a survey of this literature, see [1].

The major problem in such research is the difficulty of distinguishing between deterministic chaos and the purely random process. The most important characteristic of chaotic dynamic systems is their short-term predictability. It is well understood by now that long-term forecasting of the behavior of chaotic systems is not possible due to the exponential divergence of trajectories in such systems [2].

Nonlinear modeling with neural networks offers a promising approach for studying the prediction of chaotic time series. An appreciable amount of recent literature deals with this subject [3], [4], [5], [6]. However, the application of these models to

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 236-245, 2003.
© Springer-Verlag Berlin Heidelberg 2003

problems of nonlinear time series has been dominated by static network architectures trained by gradient descent.

2 Identification Procedure

Nonlinear time series with functional coefficients can be given by:

$$Y_t = f(\mathbf{X}_t, \varphi(\mathbf{X}_t)) + \xi_t \quad (1)$$

Where $f(.)$: is unknown non linear function. Y_t : the time series.
$\mathbf{X}_t = (Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p})$: is a vector of p-past values time series Y_t. This vector reflect some changing environmental conditions that causes the system's parameters to vary; and p represent the time delay of the time series; $\varphi(\mathbf{X}_t)$: is a vector of unknown parameters which are functions of $\mathbf{X}_t$; ξ_t : is a sequence of *iid* random variables with mean 0 and variance R_t.

The main difficulty in using the proposed model in (1) is specifying the functional coefficients. We consider in this section, a special class of neural networks called Tangent Hyperbolic neural Networks **THNN** to identify the model in (1). A complete description of a Neural Networks theory and the application of neural networks to the problem of nonlinear system identification and prediction can be found in [5], [6]. The non linear coefficient $\varphi_i(\mathbf{X}_t)$ in (1) can be approximated by a sub-**THNN**, then:

$$\varphi_i(\mathbf{X}_t) = \sum_{k=1}^{q} w_k^{\varphi i} h_k(\mathbf{X}_t) . \quad (2)$$

Representing each parameter of (2) as a neural network, which is a two layered **THNN** structure. The first layer is composed of sub networks and the second layer is the output given by:

$$Y_t = h(\mathbf{X}_t, \varphi(\mathbf{X}_t)) + \xi. \quad (3)$$

We call this new neural network a stochastic neural net (**SNN**). We propose, now, a Gradient Back propagation algorithm to train the free parameters which are the weights of the sub networks in the first layer of the **SNN**, while the weights of the second layer are viewed as fixed in every iteration of this algorithm. Denote:

$$\beta_t^{\varphi i} = \left[w_1^{\varphi i}, w_2^{\varphi i}, .., w_q^{\varphi i} \right] \in IR^q, \; i = 1, \ldots, m . \quad (4)$$

$$\mathbf{B}_{t-1} = \left[\beta_t^{\varphi_1} \beta_t^{\varphi_2} \ldots \beta_t^{\varphi_p} \right]^{\mathrm{T}} \in IR^{qxp} . \quad (5)$$

$$\mathbf{h}_{t-1} = \left[h_1(\mathrm{X}_t), h_2(\mathrm{X}_t), \ldots, h_q(\mathrm{X}_t) \right]^{\mathrm{T}} \in IR^{\mathrm{q}} . \quad (6)$$

$$\mathbf{F}_{t-1} = \left[Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p} \right]^{\mathrm{T}} \in IR . \quad (7)$$

A time t, the input vector $\mathbf{F}_{t-1}$ is presented to the neural network. In the forward pass the networks calculates the output. The output of the first layer and final output of the **SNN** are given by:

$$\Phi_{t-1} = \mathbf{h}_{t-1}\, \mathbf{B}_{t-1}\,. \tag{8}$$

$$\hat{Y}_t = h(\Phi_{t-1}\mathbf{F}_{t-1}) = h(\mathbf{h}_{t-1}^{\mathrm{T}}\mathbf{B}_{t\text{-}1}\mathbf{F}_{t-1}). \tag{9}$$

The error function to be minimized is:

$$J = \frac{1}{2}\sum_{t-1}^{n}(Y_t - \hat{Y}_t)^2 \tag{10}$$

The algorithm computes the instantaneous gradient of the network's error function (J):

$$\nabla_{t-1} = \frac{\partial J}{\partial \mathbf{B}_{t-1}}\,. \tag{11}$$

And the updates the weights according to the steepest descent equation:

$$\mathbf{B}_t = \mathbf{B}_{t-1} - \mu\nabla_{t-1} \tag{12}$$

Where μ is the learning rate. Let:

$$\varepsilon_t = Y_t - \hat{Y}_t. \tag{13}$$

$h(.)$: is the tangent hyperbolic function.
We can now perform the steepest descent procedure to each weight in the second layer:

$$\beta_t^{\varphi_i} = \beta_{t-1}^{\varphi_i} + \mu\varepsilon_t Y_{t-i}(1-\hat{Y}_t^2)\mathbf{h}_{t-1}^{\mathrm{T}} \qquad \text{for } i = 1,2,...,p \tag{14}$$

3 Non Linear State Space Representation

Having identified the system model using the above neural network algorithm, it is useful to recast the **SNN** in the state-space form to perform state estimation. This requires the coefficients φ_i ($\mathbf{X}_t$) as function of time. In this case the **SNN** can be in the following form:

$$Y_t = h(\sum_{i=1}^{p}\varphi_{i,t}\, Y_{t-i} + \xi_t)\,. \tag{14}$$

The non linear state space of **SNN** is given by:

$$\Theta_{t+1} = \mathbf{H}(\Theta_t, \varphi_t) + \omega_t\,. \tag{15}$$

$$Y_t = h(\Theta_t) + \xi_t \ . \tag{16}$$

$$\Theta_t = \begin{pmatrix} \theta_{1,t} \\ \theta_{2,t} \\ \dots \\ \theta_{p-1,t} \\ \theta_{p,t} \end{pmatrix}; \quad \mathbf{H}(\Theta_t, \varphi_t) = \begin{pmatrix} Y_{t-p+1} = \theta_{2,t} \\ Y_{t-p+2} = \theta_{3,t} \\ \dots \\ Y_{t-1} = \theta_{p,t} \\ Y_t = h(\varphi_{1,t}\ \theta_{p,t} + \dots + \varphi_{p,t}\ \theta_{1,t} \end{pmatrix}$$

Equations (15) and (16) represent respectively the transition equation and the observation equation, with Θ_t is the state vector of the system at time t and Y_t is the observation vector at time t. The ω_t is of innovations, with zero mean and variance Q_t, ξ_t is an additive measurement noise with zero mean and variance R_t. We assume that the noise vectors are uncorrelated with covariance: $P^{\omega\xi} = \begin{pmatrix} Q_t & 0 \\ 0 & R_t \end{pmatrix}$.

4 Inferential Algorithm

The Kalman filer is a set of mathematical equations that provides an efficient computational (recursive) solution of the least-squares method. A complete discussion to the idea of the Kalman filter can be found in [7].

With the above non linear state space model, the application of the Kalman filter is not straightforward because the model is not linear. We are established, a recursive algorithm for the measurement and the state update for the dynamic system represented by the **SNN**. This algorithm can be summarized by the update recursion equations.

$$\hat{\Theta}_{t+1/t} = H\left[(\hat{\Theta}_{t/t})\right] \tag{17}$$

$$P^{\Theta}_{t+1/t} = \mathbf{A}_t P^{\Theta}_{t/t} \mathbf{A}^{\mathrm{T}}_t + Q_t \ . \tag{18}$$

$$K_{t+1} = P^{\Theta}_{t+1/t} \mathbf{G}^{\mathrm{T}}_{t+1} (\mathbf{G}_{t+1} P^{\Theta}_{t+1/t} \mathbf{G}^{\mathrm{T}}_{t+1} + R_{t+1})^{-1}. \tag{19}$$

$$\hat{\Theta}_{t+1/t+1} = \hat{\Theta}_{t+1/t} + K_{t+1}(Y_{t+1} - h(\hat{\Theta}_{t+1/t}) \ . \tag{20}$$

$$P^{\Theta}_{t+1/t+1} = P^{\Theta}_{t+1/t}(\mathbf{I} - K_{t+1}\mathbf{G}_{t+1}). \tag{21}$$

$$\hat{Y}_{t+1/t} = h(\hat{\Theta}_{t+1/t}). \tag{22}$$

Where $P^{\Theta}_{t+1/t}$: the prior estimate covariance of the state;

K_t : the gain filter ; $\mathbf{A}_t = \dfrac{\partial H\left[(\hat{\Theta}_{t/t})\right]}{\partial(\hat{\Theta}_{t/t})}$; $G_{t+1} = \dfrac{\partial h(\hat{\Theta}_{t+1/t})}{\partial(\hat{\Theta}_{t+1/t})}$.

Equations (17) and (18) are the time update equations, these equations are responsible for projecting forward (in time) the current state and error covariance

estimates to obtain the prior estimates for the next time step. Equations (19) to (22) are the measurement update equations, the first task during the measurement update is to compute the Kalman gain, K_t, the next step is to actually measure the process to obtain Y_t (22), and then to generate an a posterior state estimate by incorporating the measurement as in (20), the final step is to obtain an a posterior error covariance estimate via (21). After each time and measurement update pair, the process is repeated with the previous a posterior estimate used to project or predicts the new a priori estimates.

5 Coupled Neural Net Algorithm

By now we are ready to realize the whole system of identification and filtering by combining together the identifier and the stochastic extended Kalman filter algorithms described in the previous sections. In this case, we cannot stop the learning process of the neural network, the whole algorithm of modeling and filtering can be summarized as:

1. Give initial values for the parameter matrix $\mathbf{B_0}$ for the neural network, the state Θ_0 and the covariance P_0^{Θ} .
2. Apply a new $\mathbf{F}_{t-1}$ to the input of the neural network, and evaluate the tangent hyperbolic functions, given the vector. $\mathbf{h}_{t-1}^{T}$.
3. Calculate Φ_{t-1}. (8) and the output of the neural network using (9), calculate the error between the measured system output and that of the final neural network.
4. Use the error to train the parameter matrix $\mathbf{B_t}$ of the neural network by the gradient back propagation learning rule (12).
5. With Φ_{t-1}. , form the state-space equation of (15) and (16).
6. With the state-space equation, calculate the state estimate using the Stochastic Extended Kalman filter algorithm of (20) and (21).
7. Go to step 2.

6 Simulation Study

Th the largest Lyaponov exponent contains information on how far in the future predictions are possible. The Lyaponov spectrum has proven to be one of the most useful dynamic invariants that characterize chaotic dynamic systems. The Lyaponov exponents provide us with a measure of the averaged exponential rates of divergence or convergence of neighbour orbits in phase space [8]. If at least one positive Lyaponov exponent exists, the dynamic system is said to be chaotic, and the initial small differences between two trajectories will diverge exponentially. We will concentrate our study on the largest Lyaponov exponent λ_L. Let us consider an embedding n-dimensional space. After some iteration, a small n-sphere becomes an n-ellipsoid. The i-th one dimensional Lyaponov exponent is then defined in terms of the

length of the ellipsoidal principal axis or the largest eingenvalue α_L of the Jacobean matrix. The formal definition of the Lyaponov exponent is:

$$\lambda = \lim_{t\to\infty}\left\{\frac{1}{2t}\log|\alpha_L(t)|\right\}$$

The Lyaponov exponents reflect the deformation (expansion or contraction) of the map in the different directions of phase space

In case where only experimental data are given but not the function that generates the map, we have used the algorithm of McCaffrey et al [3]. When the analytical form of the map is known (e.g., by neural networks after learning) the Lyaponov coefficient can be calculated directly.

We analyze the chaotic logistic map which has been hypothesized in a number of empirical studies of financial time series (see [9]). **SNN** is used with the coupled learning algorithm described in the previous section. After training, the networks are used in an iterative mode in order to generate a series of estimate outputs, and the dynamic invariants measured for the original series.

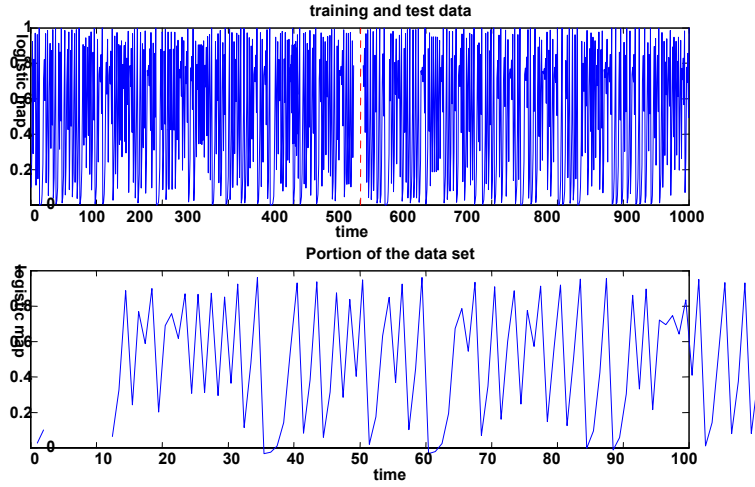


Fig. 1. Data set from the Logistic map.

The data set is displayed in figure1. In this study the chosen model has parameters d=2 and p=2, q= 15 (number of hidden units) respectively, that is:

$$Y_t = h(\sum_{i=1}^{2}\varphi_i(\mathbf{X}_t)Y_{t-i}) + \xi_t \,. \qquad (23)$$

Where $\mathbf{X}_t$ is a function of $\{Y_{t-1}, Y_{t-2}\}$ which are the neural network's inputs. The data set from 1 to 500 is used to train the neural network and from 500 to 1000 to test the neural network performance (figure1).

The network, during the training, learns the correct value of the Largest Lyaponov exponent that is 0.52 (figure2). Our analysis permits us to claim that the **SNN** capture "chaos" because it learns the dynamic invariant of a chaotic dynamic system.

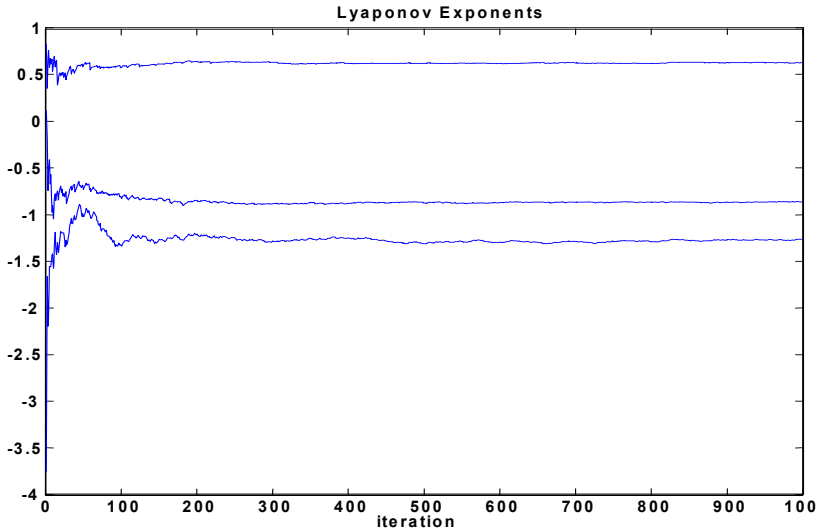


Fig. 2. Lyaponov exponents as a function of the number of iteration for the SNN trained by the Coupled Algorithm of the Logistic map.

7 Empirical Study

To illustrate the application potential of the new **SNN** architecture on real data set, the case study, in this section, involved the modeling of the Tunisian Stock price index **BVMT**. The data set were daily closed values from Sept. 30, 1990 to Jul. 07, 1998. Figure3 shows a temporal plot of the first difference of Tunisian Stock price index.

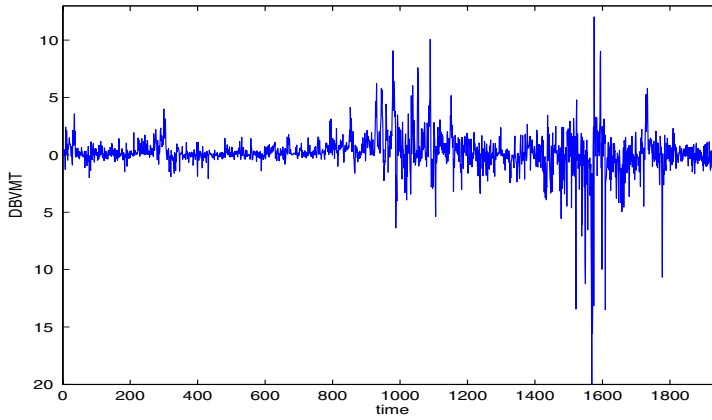


Fig. 3. First difference of the Tunisian Stock Price Index (BVMT).

As we can see, the movements of the index not follow a random walk, and the system dynamics are not smooth, then the linear approximation of this dynamics is

not appropriate. Identification of this non-linear structure by the **SNN** can clearly enable formulation of such system.

The variables of interest in this study are:

1. **BVMT**: Tunisian stock Price Index.
2. **PE** : Price/Earning ratio for Tunisian Stock Price.

These particular set of explanatory variables were selected because they are the types of variables used heavily by professional investors. The first Lag **BVMT** (**BVMT**$_{t-1}$) was also used as input for the **SNN**. The **BVMT**$_t$, **BVMT**$_{t-1}$ and the **PE**$_t$ were transformed by the first difference operator (D). The time series that can be modeled by the **SNN** is:

$$\mathbf{BVMT}_t = f(\mathbf{BVMT}_{t-1}, \mathbf{PE}_t, \varphi(\mathbf{BVMT}_{t-1}, \mathbf{PE}_t)) + \xi_t$$

The network architecture of the **SNN** was determined in part by the domain variables. Since tree variables were selected as input stream to provide a univariate forecast. A common difficulty in applying neural networks lies in over fitting the data. A rule of thumb in the field of statistical modeling specifies that, for a case base of N observations, the degrees of freedom in the model should not exceed $N^{0.5}$. Since our training set is fixed at 1000 observations, an upper bound would be about 32 weights, each corresponding to a degree of freedom in the **SNN** to be trained. Initially, the configuration 3*8*1= (corresponding to 3*8+8=32 weights) was evaluated. However, the 3*7*1 architecture yielded slightly better performance in term of prediction error. In this paper, the value of 7 hidden units was chosen for the **SNN** first layer. The **SNN** was trained and validate by the gradient back propagation and the **BNKF** described in section 3 and 4 (for μ=0.1).

The performance among the predictive model is presented in figure4. The **SNN** output is closed to the desired output. Figure6 shows the evolution of the estimate Lyaponov exponent as function of the number of iterations via the **SNN**. It is important to remark that no overtraining is observed, the largest Lyaponov exponent is 1, 48 (figure5). We conclude that the **BVMT** index is chaotic and stochastic.

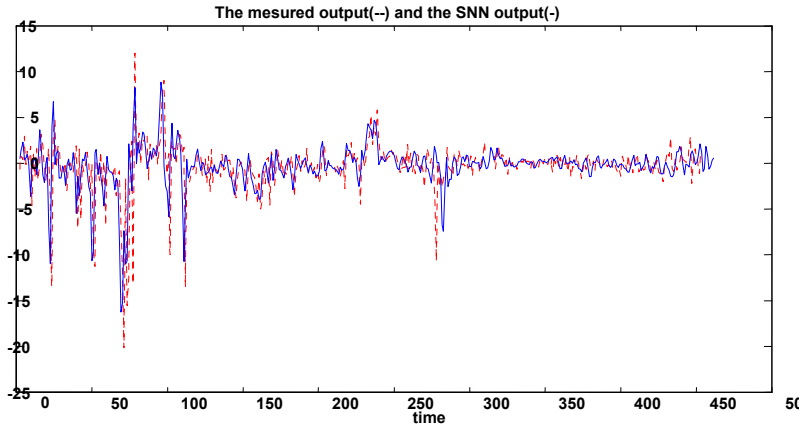


Fig. 4. Comparison of measured output and SNN output

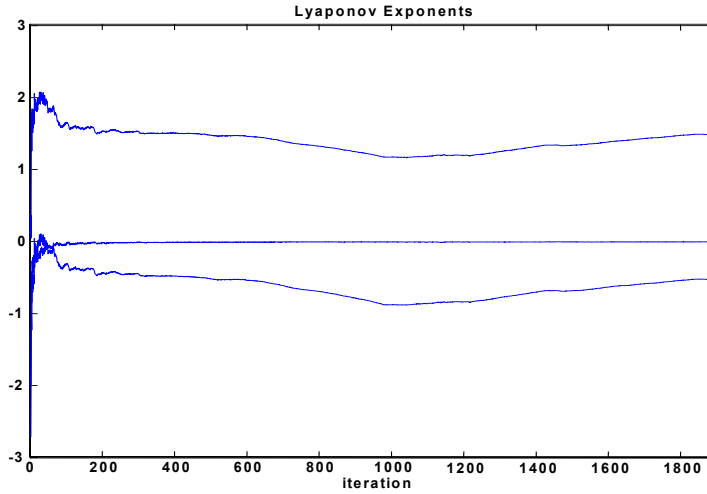


Fig.5. Lyaponov exponents as a function of the number of iteration for the SNN of the BVMT.

8 Conclusion

This paper proposed a new approach based on Extended Kalman filter to help diagnose the dynamic structure of non linear time series.

For a set of input and output, a **SNN** can be used to identify the nonlinear time series of (1). A non linear state space form for this net is presented and a Bayesian algorithm for this model is derived to estimate the state.

Our analysis permits us to claim that the **SNN** capture chaos because it learns the dynamic invariants of a chaotic dynamic system (Logistic map).

The proposed model was applied to the Tunisian stock index (**BVMT**) index). Our findings indicate that the **BVMT** index is chaotic and stochastic; the observed largest Lyaponov exponent is 1.48682.

We believe that the encouraging results obtained herein with respect to the neural modeling of chaotic systems, in combination with existing linear and nonlinear dynamic techniques [10] and [11], has a great potential for the modeling of financial, economic and other time series generated by complex market driven systems.

References

1. Blank, S.C.: Chaos in future Markets? a Nonlinear Dynamical Analysis. The Journal of Future Markets. 11.6 (1991) 711-728
2. Albano, A., Passamante, A., Hediger, T., Farell, M. E.: Using Neural Nets to look for chaos. Physica D. 58 (1992) 1

3. Nychka, D., Ellner, S., Gallant, A.R., Maccaffrey, D.: Findings chaos in noisy systems. Journal of the Royal Statistical Society, Series B. 54 (1992) 399-426
4. Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computing, Santa Fe Institute Studies in Science of Complexity, Amsterdam: Addison-Wesly (1991)
5. Weigend, .A.S., Huberman, B.A., Rumelhart, D.E.: Predicting the Future: A Connectionnist Approach. International Journal of Neural Systems, Vol 1.3 (1991) 193-209
6. Hornik, K., Stinchcombe, M., White: Multilayer feed forward networks are universal approximators. Neural Networks, vol.2 (1989) 359-366.
7. Aoki, M.: State-Space Modeling of Times Series, Berlin: Springer-Verlag (1987)
8. Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A.: Determining Lyaponov Exponents from a time series. Physica D. 16 (1985) 285-317
9. Hsieh, D.A.: Chaos and Nonlinear Dynamics:Application to Financial Markets. The Journal of Finance. 6.5 (1991) 1839-1877
10. Hillmer,S.C, Trabelsi, A.: Benchmarking of Economic Time Series. Journal of the American Statistical Association: Theory and Methods. 82 (1987) 1064-1071
11. Trabelsi, A., Hillmer, S.C.: A Benchmarking Approach to Forecast Combination. Journal of Business and Economic Statistics. (1989) 353-362

Camera Models and Optical Systems Used in Computer Graphics: Part I, Object-Based Techniques

Brian A. Barsky[1,2,3], Daniel R. Horn[1], Stanley A. Klein[2,3], Jeffrey A. Pang[1], and Meng Yu[1]

[1] Computer Science Division
[2] School of Optometry
[3] Bioengineering Graduate Group, University of California, Berkeley, California, 94720-1776, USA
http://www.cs.berkeley.edu/optical
Contact author: barsky@cs.berkeley.edu

Abstract. Images rendered with traditional computer graphics techniques, such as scanline rendering and ray tracing, appear focused at all depths. However, there are advantages to having blur, such as adding realism to a scene or drawing attention to a particular place in a scene. In this paper we describe the optics underlying camera models that have been used in computer graphics, and present *object space* techniques for rendering with those models. In our companion paper [3], we survey *image space* techniques to simulate these models. These techniques vary in both speed and accuracy.

1 Introduction

The goal of increasing realism fuels progress in computer graphics. However, perfect realism is impossible to achieve simply due to the number of interactions that occur when sending light into a scene and collecting the results on a discrete plane. The human eye can discern many details; thus, humans can notice the surreal clarity of most computer-generated scenes. In particular, images often lack depictions of focus, blur, optical aberrations, and depth of field. This paper illustrates the concepts behind modeling computer graphics with a lens system.

The rendering of a 3D scene requires transforming from *object space* to *image space*. In object space, the entire geometry of the scene, with all the materials and surfaces, is present. This transformation is parameterized by a camera angle and location. There are various algorithms that can be used to translate the 3D geometry into a final, 2D image. When the scene is transformed into a 2D image, it often retains a depth map which contains distance measurements to the camera at any given pixel; this may be used for additional post-processing algorithms. Two ubiquitous methods to translate a 3D model in object space into a 2D image are *scanline rendering* and *ray tracing*. However, images generated with these methods are rendered in focus at all distances, as if photographed through a

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 246–255, 2003.
© Springer-Verlag Berlin Heidelberg 2003

pinhole camera with an infinitely tiny aperture. Thus, these standard rendering techniques cannot convey focus nor represent arbitrary lens configurations.

The rendering of images with focal blur is an interesting and important field of research in computer graphics. Several techniques have been proposed to model camera lens systems to varying degrees of accuracy. They range from *object-based* modifications to ray tracing and scanline rendering to *image-based* techniques that convolve and otherwise distort an image after it has been rendered. We discuss object based techniques in this paper, and image based techniques in our companion paper [3], published later in these proceedings. Currently, these techniques offer varying realism and speed in rendering images with camera models. Before studying them, some background on optics will be reviewed.

2 Camera Models

Several camera models have been employed in computer graphics to approximate physical optical systems. Although each has distinctive qualities and produces characteristically different images, they all share some common traits.

A camera model simulates the capture of light from a three-dimensional scene in *object space* onto a two-dimensional image, or *image space*. Most models contain or approximate a system of parallel *lenses* such as that of a camera or the eye. An axis that passes through the geometric center of the system of lenses is called the *optical axis*. In computer graphics, the center of a single lens system is sometimes called the *center of projection* (COP).

Objects in the scene are projected through the lens system to form an *image* located on the opposite side of the system. Each lens has an *aperture* which defines the area through which light is allowed to pass to the image.

Although rendering models usually consider an *image plane* in front of the lens system when forming an image, camera models consider a *film plane* behind the lens system. Like physical optical systems, the image formed on the film plane is inverted. An appropriate image plane can be derived from the location of the film plane in some camera models. The field of view is parameterized by the size of the film plane.

In optics, the standard unit of lens power is the *diopter* [1], a unit which is measured in inverse meters. Although the amount blur is not linear in distance, it is approximately linear in diopters.

2.1 Camera Obscura (Pinhole Camera)

The standard rendering algorithms in computer graphics are equivalent to modeling a *camera obscura* or pinhole camera. In this model, all the light rays from the scene pass through a single point, called the center of projection, or through a lens with an infinitesimal aperture.

Since only a single light ray cast from each point in the scene can pass through the COP regardless of its location in space, each point will be rendered exactly

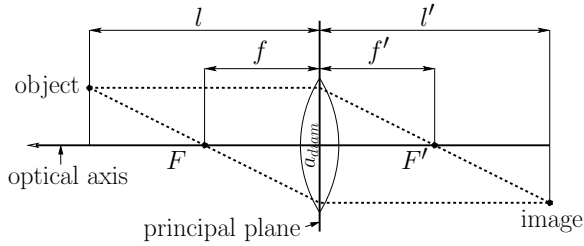


Fig. 1. Thin lens model.

once on the film plane. This causes all objects in the scene to appear in sharp focus on the image produced. The pan of the camera can be parameterized by the location of the film plane along the axis.

This model could not be realized physically since in reality, a point-sized aperture would produce an image on the film plane that would be too dim to observe. However, simply reversing this model yields a ray tracer, where a ray is projected from the COP through each pixel on the image plane and then the colors of the materials it intersects determine the color of the particular pixel. Analytic techniques have also been developed using this model to project objects to image space efficiently [7].

2.2 Thin Lens Approximation

In reality, all lenses have a finite aperture, and hence each point in the scene emits a cone of light which is visible to the lens. In geometric optics, when light rays encounter the boundary between two media, the angle through which light is refracted can be calculated using Snell's law. Since real lenses are composed of a refractive material that has a *finite* and non-constant thickness, Snell's law can be used to project light rays through a lens.

The *thin lens approximation* instead assumes that even though lenses have a finite aperture, they have an *infinitesimal* thickness. In general, thin lenses are also spherical in form, either convex or concave, and *ideal*. A lens is ideal if it has the property that the change in slope for a ray passing through the lens is proportional to the distance from the center of the lens to the point at which the ray encounters it [1]. The plane that is normal to the optical axis at the lens is called the *principal plane* (Figure 1).

Along the optical axis of the lens is the *focal point*, F. Formally, F is an axial point having the property that any ray emanating from it or proceeding toward it travels parallel to the axis after refraction by the lens [8]. A secondary focal point, F' exists on the opposite side of the lens. Analogously, this point is defined as an axial point having the property that any incident ray traveling parallel to the axis will proceed toward F' after refraction (Figure 1).

The distance between the center of the lens and a focal point is called the *focal length*. If the medium is same on both sides of the lens (hence the indices of

refraction, n and n', are equal) and if the lens is symmetric, then these distances, denoted by f and f' are the same.

The law governing image formation through a thin lens in Gaussian form is:

$$\frac{1}{l} + \frac{1}{l'} = \frac{1}{f} \tag{1}$$

where l is the distance from the object to the lens, l' is the distance from the lens to the image, and f is the focal length (Figure 1). By solving for l' in equation (1), we can derive that the projection of an object at a distance l in front of the lens will converge at a distance

$$l' = \frac{fl}{l - f} \tag{2}$$

behind the lens.

The *f-stop* (or F number), which we denote as f_{stop}, is defined as the ratio of the focal length to the diameter of the aperture of the lens, denoted by a_{diam}:

$$f_{stop} = \frac{f}{a_{diam}} \ . \tag{3}$$

When focusing at a particular distance d_{focus} in front of the principal plane, there is a specific location d'_{focus} where the film plane must be located such that each point that is at distance d_{focus} forms its image on that plane (Figure 2). The plane perpendicular to the axis at d_{focus} is called the *focal plane*. To derive the distance d'_{focus} behind the lens to place the film plane, we substitute d_{focus} and d'_{focus} for l and l', respectively, in equation (2), yielding:

$$d'_{focus} = \frac{fd_{focus}}{d_{focus} - f} \ . \tag{4}$$

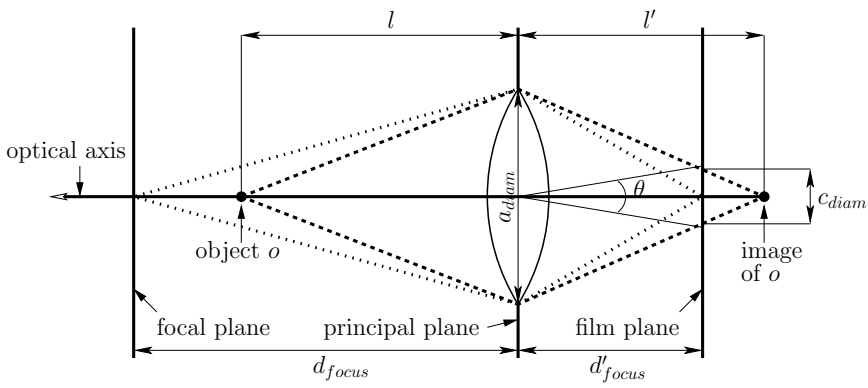


Fig. 2. The circle of confusion c_{diam} and blur angle θ in the thin lens model.

A point that is closer to the lens than the focal plane projects behind the film plane, whereas the projection of a point farther from the lens than the focal plane converges in front of the film plane. Thus, each of these projected points forms a circle rather than a point on the film plane. This is called the *circle of confusion* or blur disk, and it measures how defocused an image point is. Using similar triangles [12] in Figure 2, we can deduce from equations (2) and (4) that the diameter of the circle of confusion c_{diam} for a particular point is:

$$c_{diam} = a_{diam} \frac{l' - d'_{focus}}{l'} \ . \tag{5}$$

In optics, it is often simpler to measure blur in terms of the *blur angle* rather than the diameter of the circle of confusion. The blur angle, which we denote as θ, is the angular size, measured in radians, of a given point's blur [1]. From Figure 2, it can be seen that:

$$\tan \frac{\theta}{2} = \frac{c_{diam}/2}{d'_{focus}} \ . \tag{6}$$

When θ is small, as is usually the case for normal lenses, $\tan \frac{\theta}{2} \approx \frac{\theta}{2}$; thus equation (6) can be approximated as:

$$\theta = \frac{c_{diam}}{d'_{focus}} \ . \tag{7}$$

Substituting equation (5) for c_{diam} yields:

$$\theta = a_{diam} \frac{l' - d'_{focus}}{l' d'_{focus}} \ . \tag{8}$$

Rearranging,

$$\theta = a_{diam} \left(\frac{1}{d'_{focus}} - \frac{1}{l'} \right) \ . \tag{9}$$

The quantity in parentheses, $\frac{1}{d'_{focus}} - \frac{1}{l'}$, measures the change in curvature due to a change in the location of the object or a change in the location of the image. This is called *diopters of defocus* and is denoted by $D_{defocus}$ [1]. Using this notation, equation (9) can be written as:

$$\theta = a_{diam} D_{defocus} \ . \tag{10}$$

2.3 Thick Lens Approximation

The thin lens approximation is appropriate for the case where the thickness of the lens is small relative to the focal length. However, some lenses do not satisfy this assumption and most lens systems that comprise multiple "lens elements," like those in photographic systems, cannot be accurately approximated by a single thin lens. However, it is sometimes possible to treat these systems as ideal

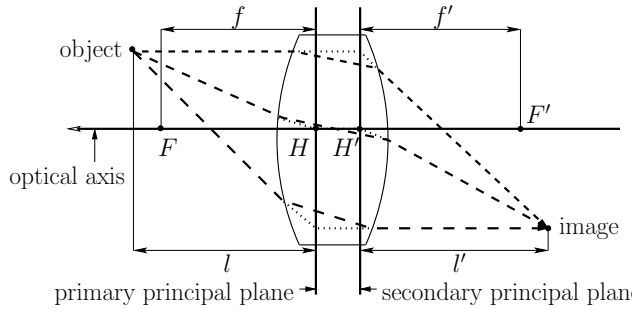


Fig. 3. Thick lens model. The dashed lines show the actual paths of rays passing through the lens. The dotted lines demonstrate that we can calculate exit directions for the rays in a similar manner as in the thin lens case with the addition of a translation between the principal planes.

thick lenses. A thick lens is usually described by a single homogeneous lens with two spherical surfaces separated by an appreciable distance.

A thick lens can be characterized by *primary* and *secondary principal planes*, H and H', and their corresponding *primary* and *secondary focal points*, F and F'. The two principal planes are defined in the same manner as was the single one in the case of the thin lens. The primary principal plane is the plane that is normal to the optical axis where any ray cast from the primary focal point F intersects the corresponding axis-parallel ray exiting the lens. The secondary principal plane is defined similarly with respect to the secondary focal point F'.

As with the thin lens, the primary and secondary focal lengths, f and f' are defined to be the distances from H to F, and from H' to F', respectively (Figure 3).

The principal planes are usually determined by tracing rays through the lens system, although they can also be calculated analytically using lens thickness formulas [10]. Once we know the location of the principal planes and focal lengths, we can determine image formation through the lens. Assuming that the medium is the same on both sides of the lens, the formula for image formation is the same as equation (1) [8]. However, since there are two principal planes instead of only one (as there is in the thin lens model), here l and l' are the distances from H to the object, and from H' to the image, respectively.

Thus, the circle of confusion for a point can be derived in a similar manner to the derivation for the thin lens model (Section 2.2). We merely have to translate each ray entering the lens from the primary to secondary principal planes (in parallel to the axis) before applying the thin lens equations to determine where the corresponding image forms.

2.4 Full Lens Systems

Kolb et al. [10] argued that the previous idealized models cannot accurately simulate the behavior of particular physical optical systems. They developed a

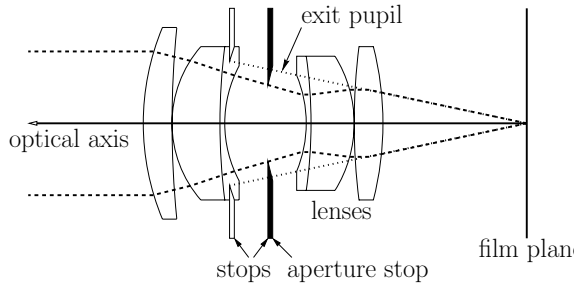


Fig. 4. Example of a full lens system. The black stop is the aperture stop for this particular point on the film plane since it most limits the incoming parallel rays of light. The exit pupil, shown by the dotted lines, defines the cone of rays from the point that pass unobstructed through the lens. (Courtesy of Craig E. Kolb. Adapted from [10] © 1995 ACM, Inc. Reprinted by permission.)



Fig. 5. Full lens simulations can exhibit aberrations that are not possible with thick lens simulations, such as appear in this image taken with a 16mm fish-eye lens. The image shows the lens' signature barrel distortion. (Courtesy of Craig E. Kolb [10] © 1995 ACM, Inc. Reprinted by permission.)

rendering technique based on a physically-based camera model that simulated a system of lenses (Figure 4).

A lens system usually comprises a series of spherical lenses and stops centered on the optical axis. A *stop* is an opaque element with a approximately circular opening that permits the passage of light. The stop that most limits the passage of rays through the system is termed the *aperture stop*. One or more of the lenses in the system usually moves relative to the film plane to change the focal point of the system. One advantage of using a lens system model is that moving a lens changes the field of view as in a physical system. The previous approximations assume that the film plane is always located at the secondary focal point and the lens can be focused at any arbitrary distance without change of configuration.

Full lens geometries can simulate an array of specific physical systems, such as that depicted in Figure 5.

Image formation through a lens system is described by Snell's law. Since the lens geometry is arbitrary, generalizations are not possible, although char-

acteristics of the system such as the *exit pupil* can be derived to aid simulation algorithms.

3 Distributed Ray Tracing

Super-sampling or oversampling was originally employed as a mechanism for dampening aliasing artifacts in ray traced images by increasing the sampling rate to better approximate the analytic model of a scene. Cook et al. [5] demonstrated that distributing ray samples could be used to model other "fuzzy phenomena" as well, including depth of field and motion blur.

Actual lenses have a finite aperture and project cones of light from the scene to each point on the film plane, as described in Section 2. To approximate the cone using the thin lens approximation, the image plane is placed at the focal plane. In computer graphics, the traditional pinhole model traces a *single ray* from the COP to each pixel on the focal plane, whereas a distributed approach traces rays from *many points* on the principal plane to each pixel on the image/focal plane. Cook et al. showed that this model accurately captures the circle of confusion that was described in Section 2.2.

Although this simple model generates photographic-like effects, Kolb et al. [10] argued that the model is not precise enough to accurately approximate specific physical optical systems. They proposed a similar distributed ray tracing technique that traces rays directly through systems of lenses (a similar rendering model was subsequently described by Barsky et al. [2] for a broader set of applications). Their algorithm traces rays from each pixel on the film plane to the surface of the lens in front of it and computes new directions for rays using Snell's law. Rays travel through each lens in the system before exiting the system to then sample the scene. Their full lens model accounts for radiometry as well. The ray tracing algorithm captures the blocking of light by lens elements when rays pass through the system at large angles to the axis; thus, the images produced will correctly exhibit vignetting and other exposure effects.

Figure 6 compares images synthesized using distributed ray tracing with different lens models.

Distributed ray tracing methods for capturing optical effects such as depth of field have several benefits. First, since these approaches integrate these effects with shading and visible surface calculations, they more accurately solve the depth of field problem and do not have problems dealing with occlusion. Second, these methods are easily implemented by simply ray tracing a stochastically chosen set of rays and then averaging the results.

Nevertheless, there are several considerations when applying these methods. The most significant is the problem of deciding the number and nature of samples to be distributed over the lens to obtain an accurate value for each pixel on the image plane. Cook [4] addressed the problems that accompanied uniform sampling at regularly spaced sample points by introducing *stochastic sampling*, or sampling at random intervals. Dippe and Wold [6] analyzed two such sampling techniques: Poisson and jitter sampling. Lee et al. [11] developed a statis-



Fig. 6. Images synthesized using distributed ray tracing using a thin lens model (left), thick lens model (center), and a full lens system (right). The full lens simulation exhibits noticeable barrel distortion. (Left image courtesy of Robert L. Cook [5]. Center and right images courtesy of Craig E. Kolb [10]. All images © 1995 ACM, Inc. Reprinted by permission.)

tical model to probabilistically limit the error in sample variance and discussed stratified sampling techniques to limit the number of samples required for good approximations. Kajiya [9] discussed a Monte Carlo algorithm that he termed *uniform sequential sampling* for sampling the aperture of a lens. This method is compatible with stratified sampling techniques and converges to the analytic model of a scene. Kolb et al. [10] used a stratified strategy based on a mapping of concentric squares to concentric circles.

The sampling problem is magnified by the multi-dimensional nature of the samples. For example, in Kolb et al.'s algorithm, the origins of sample rays must be distributed on the film plane, and target points must be distributed on the nearest lens in the system. To capture motion blur, the dimension of time must be added to the sample space. Hence, the number of samples required for a good approximation can become prohibitive.

Kolb et al. found that 16 rays per pixel were required for their images. Although tracing rays through their lens systems only consumed 10% of the rendering time, a number of additional ray samples were required. For example, the center image in Figure 6 required about 90 minutes to compute on a Silicon Graphics Indigo2.

Furthermore, distributed ray tracing techniques are only applicable when the 3D geometry to a scene is available. In addition, even though these methods correctly model certain attributes of physical optical systems, they do not model arbitrary (position or time dependent) lens aberrations nor many wavelength dependent effects such as chromatic aberration. Finally, all such systems are assumed to be aberration-limited and thus ignore the effects of diffraction.

4 Summary

In this paper, we described the optics underlying camera models that have been employed in computer graphics, and presented *object space* techniques for rendering with those models. The models range from the common pinhole camera to completely specified lens system geometries. Rendering techniques that use these models trade off complexity and efficiency for accuracy and realism. In our com-

panion paper [3], published later in these same proceedings, we survey several *image space* techniques to simulate these models, and address topics including linear filtering, ray distribution buffers, light fields, and simulation techniques for interactive applications.

Acknowledgments

The authors would like to thank Adam W. Bargteil for his helpful comments as well as Craig E. Kolb for providing figures for this paper.

References

[1] David A. Atchison and George Smith. *Optics of the Human Eye*. Butterworth-Heinemann Ltd., Woburn, Mass., 2000.

[2] Brian A. Barsky, Billy P. Chen, Alexander C. Berg, Maxence Moutet, Daniel D. Garcia, and Stanley A. Klein. Incorporating camera models, ocular models, and actual patient eye data for photo-realistic and vision-realistic rendering. Submitted for publication, 2003.

[3] Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, and Meng Yu. Camera models and optical systems used in computer graphics: Part II, Image based techniques. In *Proceedings of the 2003 International Conference on Computational Science and its Applications (ICCSA'03)*, Montréal, May 18–21 2003. Second International Workshop on Computer Graphics and Geometric Modeling (CGGM'2003), Springer-Verlag Lecture Notes in Computer Science (LNCS), Berlin/Heidelberg. (These proceedings).

[4] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[5] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In Hank Christiansen, editor, *ACM SIGGRAPH 1984 Conference Proceedings*, pages 137–145, Minneapolis, July 23–27 1984.

[6] Mark A. Z. Dippe and Erling H. Wold. Antialiasing through stochastic sampling. In Brian A. Barsky, editor, *ACM SIGGRAPH 1985 Conference Proceedings*, pages 69–78, San Francisco, July 22–26 1985.

[7] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice, 2nd Edition*. Addison-Wesley Publishing Co., Reading, Mass., 1990.

[8] Francis A. Jenkins and Harvey E. White. *Fundamentals of Optics*. McGraw-Hill, Inc., New York, 1976.

[9] James T. Kajiya. The rendering equation. In *ACM SIGGRAPH 1986 Conference Proceedings*, pages 143–150, Dallas, 1986.

[10] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. In Robert L. Cook, editor, *ACM SIGGRAPH 1995 Conference Proceedings*, pages 317–324, Los Angeles, August 6–11 1995.

[11] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically optimized sampling for distributed ray tracing. In Brian A. Barsky, editor, *ACM SIGGRAPH 1985 Conference Proceedings*, pages 61–67, San Francisco, July 22–26 1985.

[12] Michael Potmesil and Indranil Chakravarty. Synthetic image generation with a lens and aperture camera model. *ACM Transactions on Graphics*, 1(2):85–108, April 1982. (Original version in ACM SIGGRAPH 1981 Conference Proceedings, Aug. 1981, pp. 297-305).

Camera Models and Optical Systems Used in Computer Graphics: Part II, Image-Based Techniques

Brian A. Barsky[1,2,3], Daniel R. Horn[1], Stanley A. Klein[2,3], Jeffrey A. Pang[1], and Meng Yu[1]

[1] Computer Science Division
[2] School of Optometry
[3] Bioengineering Graduate Group, University of California, Berkeley, California, 94720-1776, USA
http://www.cs.berkeley.edu/optical
Contact author: barsky@cs.berkeley.edu

Abstract. In our companion paper [5], we described the optics underlying camera models that have been used in computer graphics, and presented *object space* techniques for rendering with those models. In this paper, we survey *image space* techniques to simulate these models, and address topics including linear filtering, ray distribution buffers, light fields, and simulation techniques for interactive applications.

1 Introduction

Images generated by common computer graphics algorithms often lack depictions of focus, blur, optical aberrations, and depth of field. In our companion paper [5], published earlier in these proceedings, we described several camera models that have been proposed to address these problems, and presented object space techniques for rendering with those models. In this paper, we present several *image-based* techniques to simulate these camera models. Several of these methods are useful when the 3D geometry of a scene is not available, and they are often more efficient than *object-based* techniques such as ray tracing. We will conclude with a summary of camera models and optical systems used in computer graphics and discuss future directions for research in this area.

2 Image-Based Blur

In 1981, Potmesil and Chakravarty [16] described a linear filtering technique for image-based blur. Their algorithm blurs each pixel with a properly sized blur filter at any given depth. Each pixel is smeared into a disc of a particular radius known as the *circle of confusion* to establish blur. Equation (5) in our companion paper [5] relates the depth at the current pixel in the scene to the diameter of the resulting circle of confusion.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 256–265, 2003.
© Springer-Verlag Berlin Heidelberg 2003

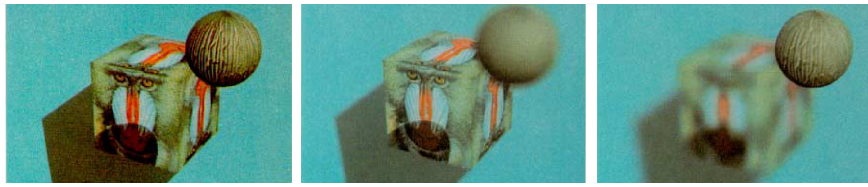


Fig. 1. Image Based Linear Filtering used on an image with a mapped cube and sphere. On the left, the cube appears focused, and on the right the sphere appears focused. (Courtesy of Indranil Chakravarty [16].)

Each point in the circle of confusion is weighted; the function that maps locations in this circle to weights is known as the *intensity distribution function* (IDF) [6, 16]. Although wave optics may be used to compute the precise IDF across the circle of confusion as a wavy Airy disc [6], according to Chen [8], a simple uniform spread of intensity across the circle of confusion can be used when the lens system is not diffraction limited or when the discretization of the pixels is too great. Smearing each pixel with the appropriate circle of confusion at its corresponding depth yields the final image.

On a Prime 750 machine from the early 1980's, a 512x512 image took between 3 and 125 minutes to process depending on the size of the aperture.

To speed the convolution process, Rokita [17] in 1993 presented an argument for approximating convolution with a large circle of confusion by using many convolutions with smaller circles of confusion. A number of faster 3x3 convolution filters can replace any single larger convolution filter. Although this does not result in an exact uniform-intensity circle of confusion, there is not a significant difference to a human observer between images that are generated by the repeated application of the smaller circle and those that are generated by the single application of the larger circle of confusion.

However, there are some deeper problems with Potmesil and Chakravarty's straightforward method of blurring. Blurring with different filters for pixels at different depths results in brightening and has artifacts arising from blurred background objects bleeding into focused foreground objects. When processing an image with a blurred white background and a sharp black foreground object, the final image will be dramatically brighter near the border of the black object as in Figure 4. This effect distorts the colors of the original image, and engenders halos around sharp objects.

The problem stems from the larger issues surrounding occlusion. A lens with a finite aperture allows a camera to view more of the scene than does a pinhole camera as shown in Figure 2, and thus it allows rays from behind an occluded object to arrive at the image plane. However, Potmesil and Chakravarty's algorithm only obtains colors that were present in the original pinhole rendering, and never determines whether a color that was originally occluded is required.

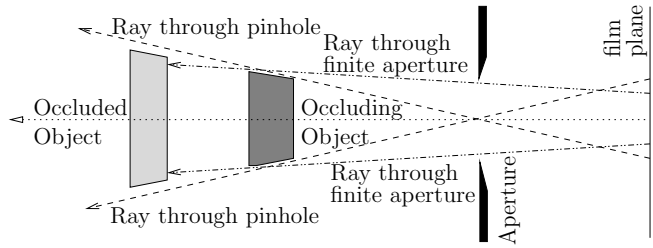


Fig. 2. An object that would be occluded in a scene ray traced through a pinhole appears through a finite aperture.

3 Ray Distribution Buffer

In [18], Shinya proposed a workaround for the inherent occlusion problem. This technique computes the final color of each pixel as an average of the colors deposited by rays entering the lens system from all angles and striking that pixel. For each pixel, the color values used in this computation are stored in a bank of memory known as the *Ray Distribution Buffer* (RDB).

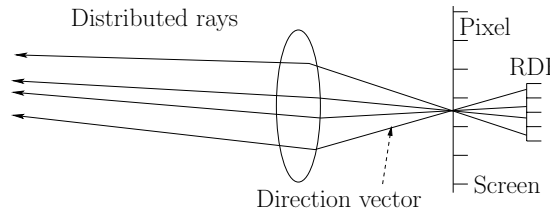


Fig. 3. Each pixel has a sub-pixel Ray Distribution Buffer. An exit direction vector can be calculated for each sub-pixel. (Courtesy of Mikio Shinya. Adapted from [18].)

Shinya's method comprises four stages: First, the image is rendered conventionally with a depth buffer.

The second stage allocates memory for the RDB for each pixel. This memory is filled with many copies of the depth and color of the corresponding pixel in the conventionally rendered image. It is desirable to have rays spreading uniformly across the lens aperture. To achieve this, a direction vector is defined for each of these rays. Each direction vector is stored in an element of the RDB. In a given pixel's RDB, the set of all such direction vectors sprinkles uniformly across the lens aperture.

In the third stage, at each pixel p of the image, the size of the circle of confusion for that pixel is calculated using equation (5) in our companion paper [5]. Then, in the initial image rendered in the first stage, each pixel q_p contained



Fig. 4. A focused black line occludes a defocused book. Left: The standard linear filtering method results in a translucent occluding object, which is incorrect in this scene. Right: The occlusion problem is solved using the RDB, and the black line is properly opaque. (Courtesy of Mikio Shinya [18].)

inside the pixel p's circle of confusion is examined. The direction of the ray emerging from each pixel q_p in the pinhole rendering is calculated; that direction is compared with the directions in the pixel p's RDB, and the corresponding memory slot is located. The depth value in the pixel p's RDB is compared with the pixel q_p's depth value and the lower depth value with its associated color is selected.

After this process is repeated for each pixel, the fourth stage of the algorithm averages the colors of the RDB into the final pixel color. This method successfully overcomes many of the problems with Potmesil and Chakravarty's simpler approach, and the difference is illustrated in Figure 4. Shinya's method requires between 155 and 213 seconds on a 150 MHz IRIS Crimson R4400 depending on the size of the circle of confusion on the majority of the image, as compared with 1321 seconds for the linear filtering method [18].

4 Blurring by Discrete Depth

Barsky et al. [2, 3] developed a method that blurs a scene with precise optics to produce a realistic final image. The method divides a scene into many slices, each slice containing pixels that share the same distance to the camera. Then it blurs each slice of the scene separately, with the appropriate level of blur. Finally the slices are combined for the final image.

The algorithm begins by separating the scene into discrete slices, according to distance from the camera. Note that both size and blur are not linear in distance, but they are approximately linear in diopters. Thus, the slices are not positioned to be equally spaced in depth but rather they are located in equal dioptric increments, ranging from the nearest depth of interest to the farthest.

The use of a scale that is linear in diopters ensures that blur will change linearly with the number of depth filters so that large distances are not covered with superfluous blur filters nor a paucity of them.

Humans can discriminate a change of approximately 1/8 diopter under optimal conditions; therefore, pixels within a slice may share a level of blur. Each slice is thus blurred with a single *blur filter*, calculated from the IDF as described in Section 2. Fast Fourier Transforms can be used in a relatively efficient method for blurring a slice.

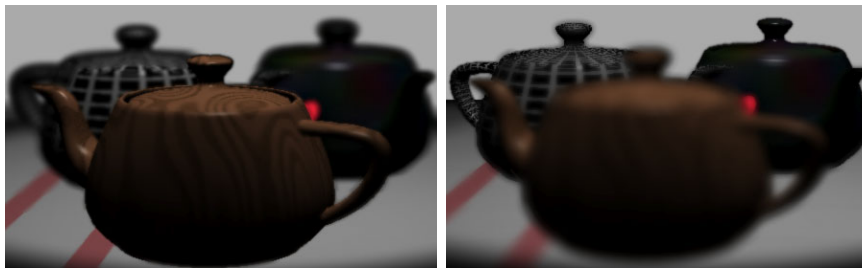


Fig. 5. Image-based blurring by discrete depths [2, 3]. In the left image the foreground is in focus with the background blurred, and in the right image the background is in focus with the foreground blurred.

Additionally, Barsky et al. [4] developed an alternate mechanism to obtain an IDF by measuring a physical optical system. An instrument that measures the path of light directed through an optical system is used to construct an IDF for any given collection of lenses or a patient's eye.

After each slice has been blurred, they are composited. To handle discontinuities between the blur filters at different depths, linear interpolation between the two resulting images is used to eliminate sharp transitions between depth filters.

The algorithm requires 156 seconds to compute blur across 11 depth slices for the teapot image in Figure 5 on a Apple G3 600 MHz iBook, which is significantly faster than distributed ray tracing on modern hardware.

5 Light Field Techniques

Light field rendering [13] and lumigraph systems [9] are alternative general methods to represent the visual information in a scene from a collection of reference images. Both techniques were proposed in 1996, and they share a similar structural representation of rays. Unlike model-based rendering, which uses the geometry and surface characteristics of objects to construct images, light field rendering, as an example of image-based rendering techniques, resamples an array of the reference images taken from different positions and generates new views of the scene. These images may be rendered or captured from real scenes.

In 1995, McMillan and Bishop [15] proposed *plenoptic modeling*, based on the *plenoptic function*, which was introduced by Adelson and Bergen [1]. A plenoptic function is a five-dimensional function that describes the entire view from a given point in space. The five dimensions comprise three parameters indicating the position of the point, as well as a vertical and horizontal angle indicating the direction of the ray from the particular point. Originally, the function also contained wavelength and time variables. Treating time as constant and light as monochromatic, the modeling ignores these two parameters. A *light field*, described by Levoy and Hanrahan [13], is a 4D reduced version of the plenoptic

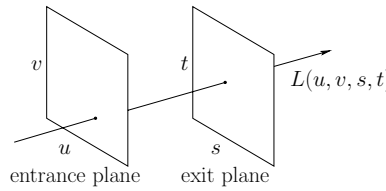


Fig. 6. 4D light field point. (Courtesy of Marc S. Levoy. Adapted from [13] © 1996 ACM, Inc. Reprinted by permission.)

function. Light field rendering assumes the radiance does not change along a line. Lines are parameterized by two coordinates of intersections of the lines on each of two parallel planes in arbitrary positions: the entrance plane and the exit plane. Figure 6 illustrates this parameterization. The system takes as input an array of 2D images and constructs a 4D light field representation. It regards the camera plane and the focal plane as the entrance and exit planes of the two-plane parameterization, respectively. Each image is taken from a particular point on the camera plane and stored in a 2D array (Figure 7). Thus, the light field is the composition of the slices of the 2D images. Once the light field is generated, the system can render a new image by projecting the light field onto the image plane, given a camera position and its orientation.

One of the major challenges of this technique is to sample the light field. Since the light field is discrete, only rays from the reference images are available. When resampling the light field to form a new image, we must approximate rays that are absent from the input images. This approximation may introduce artifacts. Different methods have been suggested for interpolating rays to address the problem. There is also some research focusing on computing the minimum sampling rate of the light field required to reduce the artifacts, such as Lin et al. [14] and Chai et al. [7]. Levoy et al.'s approach to interpolate a ray combines the rays that are close to the desired ray using a 4D low pass filter. However, this integration of rays is based on the assumption that the entire scene lies on a single and fixed exit plane. Therefore, a scene that has a large range of depth will appear blurred.

The lumigraph system, which was introduced by Gortler et al. [9], proposed a different approach to reduce the artifacts arising from the sampling problem. In the algorithm, the light field is alternatively called the lumigraph. The system first approximates the depth information of the scene. Next, instead of using a single exit plane as is the case in Levoy et al. [13], the lumigraph constructs a set of exit planes at the intersections of the rays and the objects, and then projects the rays to the corresponding planes. A basis function based on the depth information is associated with each 4D grid point in the light field. The color value of the desired ray is a linear combination of the color values of the nearby 4D points weighted by the corresponding basis functions. The lumigraph system reduces the artifacts by applying depth information; however, such information can be difficult to acquire from images captured from real scenes.

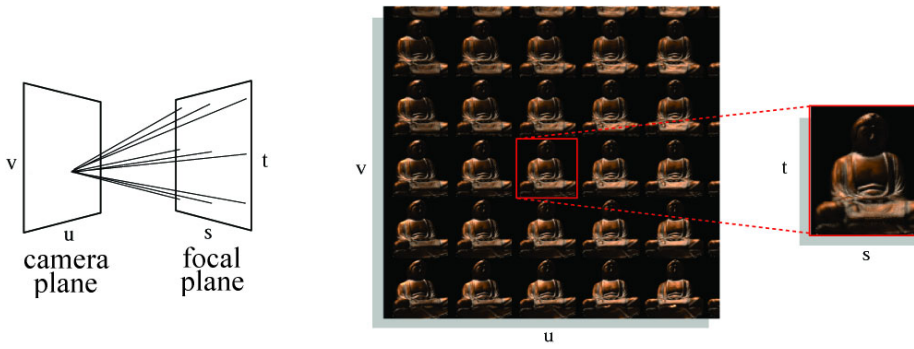


Fig. 7. The visualization of a light field. Each image in the array represents the rays arriving at one point on the camera plane from all points on the focal plane, as shown at left. (Courtesy of Marc S. Levoy. Adapted from [13] © 1996 ACM, Inc. Reprinted by permission.)

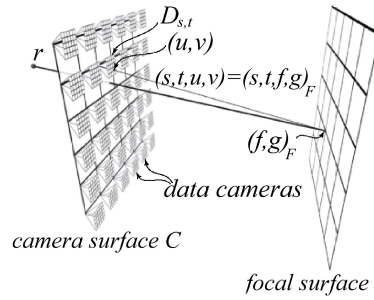


Fig. 8. Isaksen's parameterization uses a camera surface C, a collection of data cameras $D_{s,t}$ and a dynamic focal surface F. Each ray (s, t, u, v) intersects the focal surface F at $(f, g)_F$ and is therefore named $(s, t, u, v)_F$. (Courtesy of Aaron Isaksen. Adapted from [11].)

In 2000, Isaksen [11, 12] developed the dynamically reparameterized light field. This technique extended the light field and lumigraph rendering methods to enable a variety of aperture sizes and focus settings without requiring geometric information of the scene. The system associates each point on the camera plane with a small camera, called a *data camera* (Figure 8). A image is taken from each data camera at the corresponding point. To reconstruct a ray in the light field, the system maps a point on the focal surface onto several data cameras. This results in a ray for each selected data camera for the corresponding point on the focal surface. A mapping function is defined to decide through which data cameras the rays pass. Then the system applies a filter to combine the values of the rays from all data cameras. To vary the aperture sizes, the system changes the shape of the aperture filter. This changes the number of data cameras through which the rays pass. The aperture filter is also used to apply depth information.

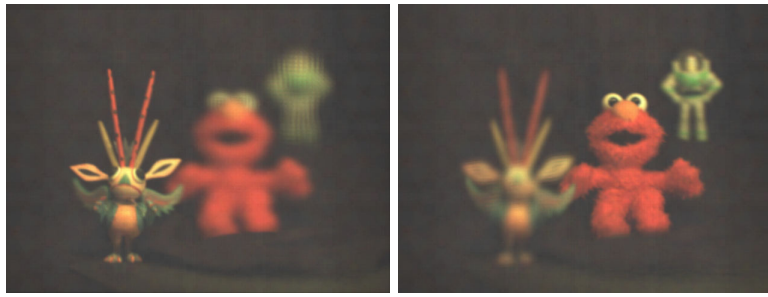


Fig. 9. By changing the shape of the focal surface, the same image can be rendered with varying parts in focus. (Courtesy of Aaron Isaksen [11].)

The rays from several data cameras pass through the aperture and intersect on the focal surface. If the intersection is near the surface of the virtual object, the ray will appear in focus; otherwise, it will not. On the other hand, the range of the depth of field is limited by the number of data cameras on the camera surface. A deeper scene requires more data cameras on the camera surface. Similar to varying aperture sizes, the system changes the focal surface dynamically. The sampling results in the same view of the scene with different focus, depending on which focal surface is chosen. Figure 9 shows the results of the same image, rendered with different focal surfaces. The light field input images are captured by an Electrim EDC1000E CCD camera. Then, the new images are rendered by PC rasterizers for Direct3D 7. The real-time renderer allows users to change the location of the camera and the focal plane, and the aperture size in real time.

An important advantage of light field rendering techniques is the capability of generating images without a geometric representation. Instead, the geometric information is implicitly obtained from a series of reference images, taken from the surrounding environment. Furthermore, rather than simply mapping rays to pixel colors, these techniques map rays to rays and apply a prefiltering process, which integrates over collections of nearby rays.

6 Realistic Lens Modeling in Interactive Computer Graphics

Heidrich et al. [10] introduced an image-based lens model, which represents rays as a light field and maps slices of the scene light field into corresponding slices of the camera light field, instead of using ray tracing. This technique simulates the aberration of a real lens and achieves hardware acceleration at the same time. The model describes a lens as a transformation of the scene light field in front of the lens into the camera light field between the lens and the film. Due to the aberration of the lens, rays from a grid on the film do not intersect in a single point on the image-sided surface of the lens system. In this case, the mapping cannot be described as a single perspective projection. A virtual center of projection (COP) is approximated for each grid on the film. The system

calculates the angle between the original rays and those generated by the perspective transformation through the corresponding calculated virtual COP. If the angle is small enough, the system continues the approximation on the next grid; otherwise, the grid is subdivided and the system calculates a virtual center of projection and repeats the same process for each subgrid. Eventually, all slices of the camera light field, are combined with weights relative to the radiometry of the incoming rays to form the final image. In addition to realistic modeling of the lens, another important contribution of this technique is its application in interactive computer graphics. Light fields, which are composed of 2D images, can be easily generated by computer graphics hardware at high speeds. Once the light field is sampled, a final image is ready to be rendered. Heidrich's experiments generate images on a RealityEngine2 using 10 sample points on the lens with approximately 14 frames per second. The grid size on the film plane is 10 x 10 polygons.

7 Summary and Future Directions

This paper covered the theory underlying lens systems that add realistic vision and camera effects to a scene. There are both object space and image space techniques, and they have different tradeoffs with respect to blur quality, speed, and realism. Future directions of research may involve extending Potmesil and Chakravarty's technique to improve object identification and occlusion detection when adding blur to a prerendered image. Likewise, modifying Heidrich et al.'s technique for rendering a lens with arbitrary geometry would result in a significantly faster technique for rendering fish-eye distortions and other distortions that cannot be approximated with a thick lens. Other directions of future research could include automatically determining how many rays or lens sample points are necessary for a given image quality in the Kolb et al. and Heidrich et al. techniques, respectively.

Acknowledgments

The authors would like to thank Adam W. Bargteil, Billy P. Chen, Shlomo (Steven) J. Gortler, Aaron Isaksen, and Marc S. Levoy for their useful discussions as well as Mikio Shinya and Wolfgang Heidrich for providing figures.

References

[1] E.H. Adelson and J.R. Bergen. *Computational Models of Visual Processing.* The MIT Press, Cambridge, Mass., 1991.

[2] Brian A. Barsky, Adam W. Bargteil, Daniel D. Garcia, and Stanley A. Klein. Introducing vision-realistic rendering. In Paul Debevec and Simon Gibson, editors, *Eurographics Rendering Workshop*, pages 26–28, Pisa, June 2002.

[3] Brian A. Barsky, Adam W. Bargteil, and Stanley A. Klein. Vision realistic rendering. Submitted for publication, 2003.

[4] Brian A. Barsky, Billy P. Chen, Alexander C. Berg, Maxence Moutet, Daniel D. Garcia, and Stanley A. Klein. Incorporating camera models, ocular models, and actual patient eye data for photo-realistic and vision-realistic rendering. Submitted for publication, 2003.

[5] Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, and Meng Yu. Camera models and optical systems used in computer graphics: Part I, Object based techniques. In *Proceedings of the 2003 International Conference on Computational Science and its Applications (ICCSA'03)*, Montréal, May 18–21 2003. Second International Workshop on Computer Graphics and Geometric Modeling (CGGM'2003), Springer-Verlag Lecture Notes in Computer Science (LNCS), Berlin/Heidelberg. (These proceedings).

[6] Max Born and Emil Wolf. *Principles of Optics*. Cambridge University Press, Cambridge, 7th edition, 1980.

[7] Jin-Xiang Chai, Xin Tong, and Shing-Chow Chan. Plenoptic sampling. In Kurt Akeley, editor, *Proceedings of ACM SIGGRAPH 2000*, pages 307–318, New Orleans, July 23–28 2000.

[8] Yong C. Chen. Lens effect on synthetic image generation based on light particle theory. *The Visual Computer*, 3(3):125–136, October 1987.

[9] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In Holly Rushmeier, editor, *ACM SIGGRAPH 1996 Conference Proceedings*, pages 43–54, New Orleans, August 4–9 1996.

[10] Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Proceedings of Graphics Interface 1997*, pages 68–75. Canadian Human Computer Communication Society, May 1997.

[11] Aaron Isaksen. Dynamically reparameterized light fields. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass., November 2000.

[12] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In Kurt Akeley, editor, *Proceedings of ACM SIGGRAPH 2000*, pages 297–306, New Orleans, July 23–28 2000.

[13] Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *ACM SIGGRAPH 1996 Conference Proceedings*, pages 31–42, New Orleans, August 4–9 1996.

[14] Zhouchen Lin and Heung-Yeung Shum. On the numbers of samples needed in light field rendering with constant-depth assumption. In *Computer Vision and Pattern Recognition 2000 Conference Proceedings*, pages 588–579, Hilton Head Island, South Carolina, June 13–15 2000.

[15] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *ACM SIGGRAPH 1995 Conference Proceedings*, pages 39–46, Los Angeles, August 6–11 1995.

[16] Michael Potmesil and Indranil Chakravarty. Synthetic image generation with a lens and aperture camera model. *ACM Transactions on Graphics*, 1(2):85–108, April 1982. (Original version in ACM SIGGRAPH 1981 Conference Proceedings, Aug. 1981, pp. 297-305).

[17] Przemyslaw Rokita. Fast generation of depth-of-field effects in computer graphics. *Computers & Graphics*, 17(5):593–595, September 1993.

[18] Mikio Shinya. Post-filtering for depth of field simulation with ray distribution buffer. In *Proceedings of Graphics Interface '94*, pages 59–66, Banff, Alberta, May 1994. Canadian Information Processing Society.

Moment Based Transfer Function Design for Volume Rendering

Zhang Jiawan[1], Sun Jizhou[1], Sun Zhigang[1], and Wei Zunce[2]

[1] IBM Computer Technology Center, Tianjin University, Tianjin 300072, P.R.China
{jwzhang, jzsun, zhgsun}@tju.edu.cn
http://jwzhang.yeah.net
[2] Department of Computer Science,Concordia University,Montreal, Quebec, Canada
weizunce@eyou.com

Abstract. Local data features play important roles in the transfer function design for volume rendering. By examining relationships among three eigenvalues of *inertia matrix*, a semi-automatic data-driven transfer function design method is presented in this paper. Local features detected by local block based moments, such as flat, round, elongated shapes are used to guide the design of transfer functions. Furthermore, the proposed method can objectively fulfill the function of the previous boundary based method and extend the domain of transfer function to include more data features. Practice experiments are conducted to render real medical data sets by using the proposed transfer function method.

1 Introduction

Direct volume rendering has gained great popularity as a powerful technique for the visualization of 3D volume data sets such as those obtained from computed tomography (CT), magnetic resonance imaging (MRI), computational fluid dynamics (CFD), and volumes which are generated by voxelizing geometric models in recent years.

Unlike surface based method, such as marching cubes[1] which draws geometric surfaces out from the volume data sets using threshold values determined by segmentation, in direct volume rendering[4][2][3][5][6], the transfer function is responsible for the classification of a data set. Its task is to assign optical properties such as color and opacity to values the data set consists of. During the rendering process, the sampled and/or reconstructed data values are passed through the transfer function to determine their contribution to the final image. So transfer functions are crucial to the quality of the final rendering images. Good transfer functions resulted informative renderings which reveal the essence and help people gain insight of the original data sets.

Unfortunately, interactive designing of a high performance transfer function which can produce an informative rendering has been proved difficult. First, in cases, little pri-knowledge makes it difficult to obtain information and gain understanding of the data set. Second, the same data value may belong to different structure or matter, in reverse, the same structure or matter may present the

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 266–274, 2003.
© Springer-Verlag Berlin Heidelberg 2003

same data value due to noise. In another word, automatic segmentation and classification of arbitrary volume data is still difficult in science. Third, complexity of the volume rendering process results in the nonlinear relationship between the optical properties produced by the transfer function and the final rendering image. Trivial tune of transfer function may lead to tremendous change in the final rendering. At last, the time-consuming process of the volume rendering makes it more expensive to evaluation of the selected transfer function.

A new *data driven* transfer function design method which uses three eigenvalues of *inertia matrix* to measure local shape features is presented in this paper. The proposed method extends the domain of transfer function, and can be used efficiently to detect local flat, round, and elongated shape features.

2 Previous Work

The earliest and also the most generally used method to set a transfer function is *trial-and-error*, which is at the same time frustrating and time-consuming.

Much work has been done by the research society in recent years. There exist two distinguish categories in the previous research on transfer function design. One is termed *image-driven* in which the rendering results is exploited to guide the design of transfer function, and the other is *data-driven* in which the user mainly concentrates on exploration of the inherent characteristics of the data sets.

As for image-driven methods, He et al. [7] describes the search of good transfer functions as a parameter optimization problem. One of common genetic algorithms—stochastic search is used to achieve global optimization. The method succeeds in generating good renderings,and frees the user from having to edit the transfer function manually. Marks et al. [15] treat the design of transfer functions as a "parameter tweaking" problem. In this paper, "design galleries" interfaces are implemented to present the user with broad selection, automatically generated and organized of perceptually different graphics or animations. Fang et al. [16] presents an image-based transfer function model based on three dimensional (3D) image processing operations. König and Gröller [8] organize the rendered thumbnails efficiently to guide the transfer function process based on volume hardware.

As for the data-driven methods, Bajaj et al. [9] describe a tool for assisting the user in selecting isovalues for effective isosurface volume visualizations of unstructured triangular meshes for isosurface rendering. Fujishiro et al. [17] use a "Hyper Reed graph" to depict the isosurface topology at any given isovalue, as well as the isovalues corresponding to critical points where the topology changes. Given some assumptions about what topological characteristics signify important isosurfaces, this information can be used as guidance to set isovalues and transfer functions. Or, if isosurfaces at the critical values are assumed to be important, setting isovalues and transfer functions can be largely automated. Kindlmann et al . [11] [12] demonstrates an innovate semi-automatic transfer function design method based on the analysis of a three- dimensional histogram

which records the correlation, throughout the given data set, between data value, gradient magnitude, and the second directional derivative along the gradient direction. The method calculates opacity functions which aim to make opaque only those positions in the transfer function domain which reliably correspond to the boundary between two relatively homogeneous regions. Sato et al. [14] use weighting functions of eigenvalues of the Hessian matrix to measure the shape of local structure in terms of edge, sheet, line, and blob. Two or three of these measures can be used as axes of a transfer function emphasize different structures in the volume according to their shapes, which tends to have biological significance in the context of medical imaging. Pekar et al. [18] propose a method which requires only a single pass through the data set to create a Laplacian-weighted histogram of data values to guide the isovalue selection and opacity function creation.Hladůvka et al. [10] use two-dimensional space of principal surface curvature magnitudes (κ_1, κ_2) to form a transfer function domain, in which is trivial to enhance and color different structures in the volume according to their surface shape.

Of the most relevance to the present paper is the work of Tenginakai et al. [13] Their work is based on histograms of values in a local window. Histograms can be described in terms of their moments, which are model-independent statistical estimators of central tendency. The variance is the second moment, and skew and kurtosis are based on the third fourth moments, respectively. A two-dimensional scatterplot of the relationship between the center voxel's data value, and the associated histogram's statistics,can serve as a "statistical signature" which summarizes the behavior of a statistical measure across the space of data values. By assuming a model of the boundary as a discontinuous junction between two fixed values, one can determine parametric curves which, if present in the statistical signatures, indicate the presence and characteristics of a boundary. The approach proposed in the present paper is also based on similar moment concept, but uses a a completely different method to guide the transfer function design.

3 Method

One trend in the recent researches is to exploit spatial features derived from local neighborhood of each voxel, such as local boundary and curvature, to guide the transfer function design process. Besides the local features mentioned above, orientation and shape are also key information which are widely used in pattern recognition and image analysis researches. The main motivation of this work is the fact that eigenvalues and eigenvectors of the inertia matrix determine the shape and orientation features of the object. In this paper, the eigenvalues and eigenvectors of inertia matrix is computed based on 3D local block based moments(LBBM). This section starts from a brief description of the 3D moments and gives the definition of LBBM. And then we discuss the features derived from relationships among three eigenvalues in more detail. At last, we discuss the similarity between the method and previous researches in boundary detection.

3.1 3D Moments

The $(i+j+k)^{th}$ *order 3D moment* of a density function $f(x,y,z)$with respect to the origin $(0,0,0)$ is defined by the Riemann integrals [19]:

$$M_{ijk} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^i y^j z^k f(x,y,z) dxdydz \tag{1}$$

If the density function is piecewisely continuous and bounded in a finite region in R^3 space, then moments of all order exist. In the context of three dimensional volume, $(i+j+k)^{th}$ *order moment* can be computed by a discrete sum approximation of Equation(1):

$$M_{ijk} = \sum_x \sum_y \sum_z x^i y^j z^k f(x,y,z) \tag{2}$$

Moments can be used to characterize objects. In fact, if all moments of two objects exist and are identical then the two objects are the same.

3.2 Definition of LBBM

Moment is generally a global feature of object. In order to implement localization in spatial domain, in this paper, the moments are computed within small local windows around each voxel. Given a window size, a local coordinate system is built, the voxel being at the origin, as shown in Fig. 1.

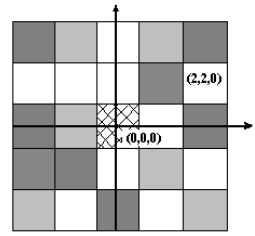


Fig. 1. Local coordinate system for LBBM computation

We call the moment computed with respect to this local coordinate system local block based moment(LBBM).

Let W be the window width. In our algorithm, we always choose the window width W to be odd so that the voxel is centered on a grid point. The LBBM is computed within the block as:

$$M_{ijk} = \sum_{-W/2}^{W/2} \sum_{-W/2}^{W/2} \sum_{-W/2}^{W/2} x^i y^j z^k f(x,y,z) \tag{3}$$

The gravity centroid $g = (g_x, g_y, g_z)$ is determined by first-order moments:

$$\begin{aligned} g_x &= \frac{M_{100}}{M_{000}} \\ g_y &= \frac{M_{010}}{M_{000}} \\ g_z &= \frac{M_{001}}{M_{000}} \end{aligned} \tag{4}$$

Then we can define the *central* $(i, j, k)^{th} order LBBM$ as:

$$M_{ijk} = \sum_{-W/2}^{W/2} \sum_{-W/2}^{W/2} \sum_{-W/2}^{W/2} (x - g_x)^i (y - g_y)^j (z - g_z)^k f(x, y, z) \tag{5}$$

The discrete computation of the set of moments for a given voxel over a finite cube window correspond to a neighborhood operation, e.g. if $W = 3$, we can use the 6 neighborhoods or 26 neighborhoods of the 3D case.

3.3 New Transfer Function Design Method by Using Local Shape Features

We assume that an object is given as a set of block voxels in some image lattice with a ellipsoidal shape. The main difference between two objects is that they have different principal axes and different orientations.

Ellipsoids can be represented by the *inertia matrix*, which is a symmetric matrix defined by the second order central moments:

$$I = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix} \tag{6}$$

where each entry of this matrix is a linear combination of two second order central moments. Since the *inertia matrix* is symmetric, its eigenvalues must be real. If R is the matrix of its eigenvectors, and $(\lambda_1, \lambda_1, \lambda_3)$its eigenvalues, then we can obtain the following equation:

$$R^T I R = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \tag{7}$$

Here, let $(\lambda_1, \lambda_2, \lambda_3)$be the three eigenvalues sorted by value so that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$, and the corresponding eigenvectors are expressed as $n_1, n_2, and\ n_3$. The eigenvector n_1 corresponding to the largest eigenvalue λ_1 determines the direction of the largest dispersion of the object, the second largest eigenvalue λ_2 corresponding to the second largest dispersion, and so forth. Thus three eigenvalues of the inertia matrix can be used to characterize important features of an object. Here, we built up a method to detect shape features by using relationship among three eigenvalues. This method is demonstrated in Table 1. And diagram based description is given in Fig. 2, which shows a round object, a elongated

object, and a flat object,respectively. The arrows indicate the directions of the principal vectors, and the lengths of arrows denote the qualitative magnitudes of eigenvalues.

Our transfer function design method is based on the local shape features indicated by relationships among three eigenvalues of inertia matrix discussed above.

We demonstrate our algorithm steps as follows:

1. Choose proper window width W, and then compute 0 order LBBM;
2. Compute the gravity centroid, then compute all 2-order central LBBMs;
3. Compute all eigenvalues and eigenvectors;
4. Based on the relationships of Table 1, design corresponding transfer function;
5. Using volume rendering algorithm to render the final images;
6. Evaluate the renderings by the user. Algorithm terminates if user satisfies with the results or algorithm turns back to step (1).

From the pipeline of the proposed algorithm above, we can conclude that our method is still a semi-automatic transfer function design.

Table 1. Shapes determined by relationships among three eigenvalues of inertia matrix

Relationship	Shape
$\lambda_1 \cong \lambda_2 \gg \lambda_3$	flat
$\lambda_1 \cong \lambda_2 \cong \lambda_3$	round
$\lambda_1 \gg \lambda_2 \cong \lambda_3$	elongated

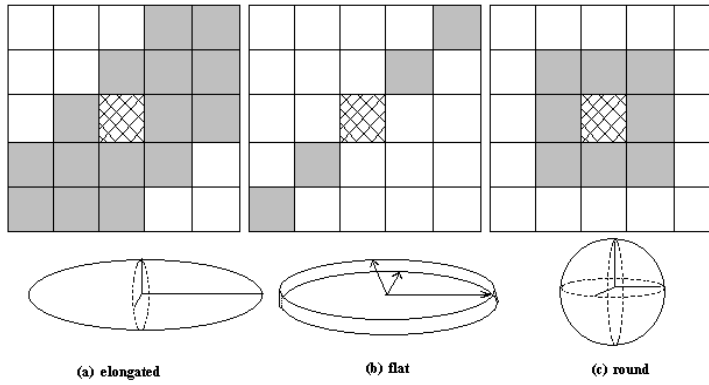


Fig. 2. shape features derived from three principal axes

3.4 Boundary Detection Using Eigenvalue Measures

The proposed transfer function by this paper can also act as a boundary detector. A simple description of the principles of our method is given in this section.

Fig. 3 shows three representative positons where a voxel may fall into, with a ideal isotropic data set. A is internal voxel, B at boudary, and C external voxel. For both A and C and those complete internal or external voxels, $\lambda_1 \neq 0, \lambda_2 = \lambda_3 = 0$. However, for B and voxels at boundaries, $\lambda_1 \neq \lambda_2 \neq \lambda_3$. By using information obtained from analysis just mentioned, one can easily make opaque of those voxels at boundaries of two structures or substances in real data sets.

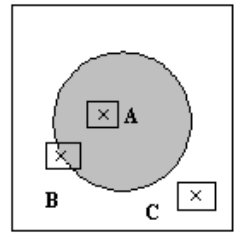


Fig. 3. Boundary detection from inertia matrix eigenvalues

4 Experiments and Results

We implement a volume rendering program which integrates the proposed transfer function algorithm on a PC desktop to detect shape features in different volume data. C++ and OpenGL are used in writing our test program. Four medical data sets are used in our experiments. The first data set is CT scanned tooth(*We download the data set from http://visual.nlm.nih.gov/data*). The second data set is the famous CT engine. The third one is a MRI head(*Data set courtesy of Klaus Engel*). And the last one is a X-ray foot(*Data set courtesy of Philips Research, Hamburg, Germany, downed from www.volren.org*). The rendering results are shown in Fig. 4.

5 Conclusion and Future Work

By examining relationships among three eigenvalues of *inertia matrix*, a semi-automatic data-driven transfer function design method is presented in this paper. Local features detected by local block based moments, such as flat, round, elongated shapes are used to guide the design of transfer functions. The proposed method extends the domain of transfer function to include more data features. Future work includes optimization of moment window width and development of proper user-interfaces for easy exploration and manipulation.

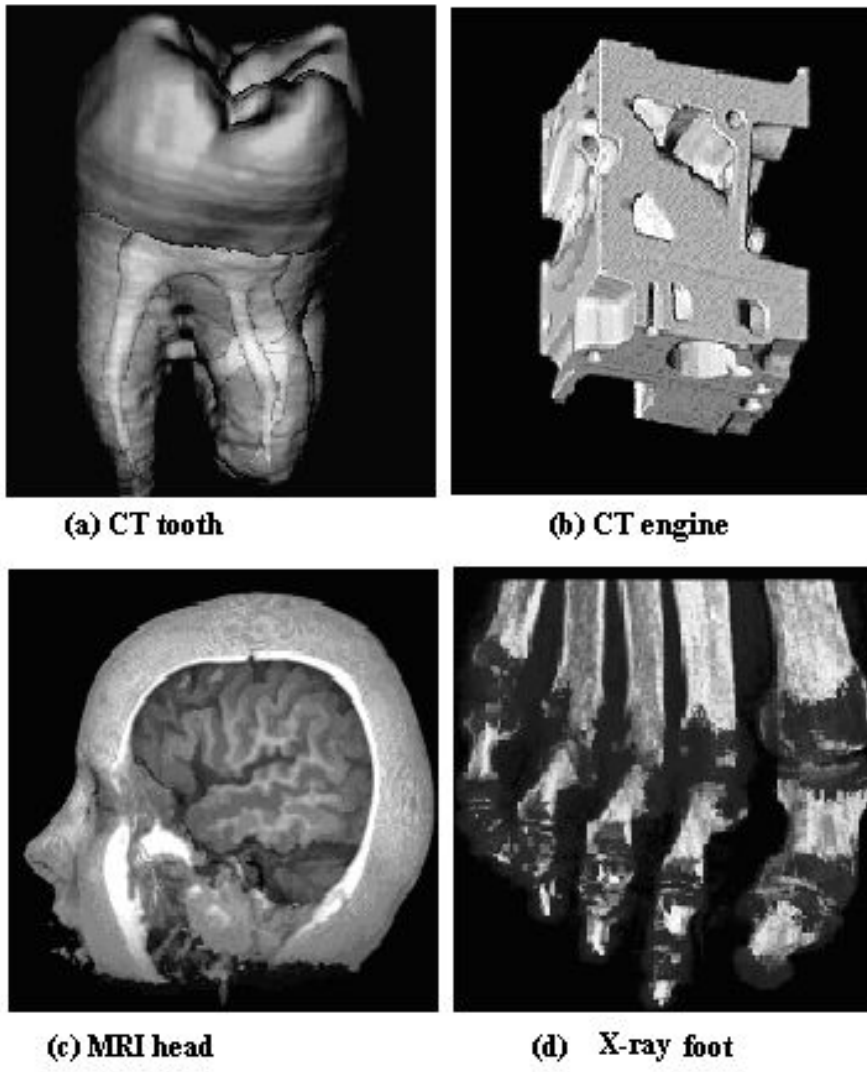


Fig. 4. Rendering results by using the proposed transfer function

References

1. Lorensen W. E., Cline H. E.: Marching Cubes: A high-resolution 3D surface construction algorithm. Computer Graphics. **21** (1988) 44–50
2. Drebin R. A., Carpenter L., and Hanrahan P. : Volume rendering. Computer Graphics. **22** (1988) 65–74
3. Levoy M. : Display of surfaces from volume data. IEEE Computer Graphics & Applications. **8**(1988) 29–37

4. Kajiya J. T., and Von Herzen B. P. : Ray tracing volume densities. Computer Graphics. **18** (1984) 165–174
5. Westover L. : Footprint evaluation for volume rendering. Computer Graphics. **24** (1990) 367–376
6. Lacroute P., Levoy M.: Fast volume rendering using a Shear-Warp factorisation of the viewing transform. Computer Graphics. **8** (1994) 451–459
7. He T., Hong L., Kaufman A., and Pfister H.: Generation of transfer functions with stochastic search techniques. Proceedings Visualization' 96. (1996) 227–234
8. König A. H., and Gröller E. M. : Mastering Transfer Function Specification by Using VolumePro Technology.Proceedings of the 17th Spring Conference on Computer Graphics(SCCG). **21** (2001) 279-286
9. Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore.:The Contour Spectrum. Proceedings Visualization 97. (1997) 167–173
10. Hladùvka,König A. H., and Gröller E. M.: Curvature-Based Transfer Functions for Direct Volume Rendering. Spring Conference on Computer Graphics 2000. **16** (2000) 58–65
11. Gordon Kindlmann. : Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. Masters thesis. Cornell University (1999)
12. Gordon Kindlmann and James W. Durkin. : Semi-AutomaticGeneration of Transfer Functions for Direct Volume Rendering. IEEE Symposium On Volume Visualization. (1998) 79–86
13. Shivaraj Tenginakai, Jinho Lee, and Raghu Machiraju. : Salient Iso-Surface Detection with Model-Independent Statistical Signatures. Proceedings Visualization 2001. (2001) 231–238
14. Yoshinobu Sato, Carl-Fredrik Westin, Abhir Bhalerao, Shin Nakajima, Nobuyuki Shiraga, Shinichi Tamura, and Ron Kikinis. : Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering. IEEE Transactions on Visualization and Computer Graphics. **6**6 (2000) 160–180
15. Marks J. , Andalman B. , Beardsley P.A., and Pfister H. : Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. ACM Computer Graphics (SIGGRAPH 97 Proceedings). (1997) 38–400
16. Shiaofen Fang, Tom Biddlecome, and Mihran Tuceryan. : Image-Based Transfer Function Design for Data Explorationin Volume Visualization. Proceedings IEEE Visualization'98 . (1998) 319–326
17. Issei Fujishiro, Taeko Azuma, and Yuriko Takeshima. : Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topology Analysis. Proceedings IEEE Visualization'99. (1999) 467–470
18. Vladimir Pekar, Rafael Wiemker, and Daniel Hempel. : Fast Detection of Meaningful Isosurfaces for Volume Data Visual-ization. Proceedings IEEE Visualization 2001. (2001) 223–230
19. Gabriele Lohmann.: Volume Image Analysis. John Wiley & Sons and B.G.Tuehner (1998) 53–59

Information-Theory-Based Oracles for Hierarchical Radiosity

Jaume Rigau, Miquel Feixas, and Mateu Sbert

Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain
{jaume.rigau, miquel.feixas, mateu.sbert}@udg.es

Abstract. In the radiosity method scene meshing has to accurately represent illumination variations avoiding unnecessary refinements that would increase the computational cost. In this paper we present three refinement criteria for hierarchical radiosity which are based on the information content of a ray between two patches or elements and the loss of information transfer between them due to the discretisation. The results obtained with our information-theory-based oracles improve on the classic ones based on transported power, kernel smoothness, and smoothness of received radiosity.

1 Introduction

The radiosity method solves the problem of illumination in an environment with diffuse surfaces, by using a finite element approach. The scene discretisation has to accurately represent illumination variations, trying to avoid unnecessary subdivisions that would increase the computational time. A correct strategy should balance accuracy and computational cost.

In the hierarchical radiosity algorithms [1,2] the mesh is generated adaptively: when the constant radiosity assumption on a patch is not valid for the radiosity received from another patch, the refinement algorithm will refine it in a set of subpatches or elements. An oracle or refinement criterion informs us if a subdivision of the surfaces is needed, bearing in mind that its cost should remain acceptable.

In this paper, three information-theory (IT) oracles, based on entropy (information) and mutual information (information transfer), are applied to hierarchical radiosity. These oracles derive from the previous results obtained by the authors on IT applications to radiosity. The results show that our three IT oracles improve on their classic counterpart ones based, respectively, on transported power, kernel smoothness, and smoothness of received radiosity.

This paper is organized as follows: In section 2, we review different refinement oracles used in the hierarchical radiosity setting and also IT tools applied to a scene. In section 3, we present three IT oracles based on the information content of a ray and the loss of information transfer between two patches. In section 4, some experiments compare our methods with the three classic oracles. Finally, in section 5, we present our conclusions.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 275–284, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Previous Work

In this section we review three classic refinement criteria used in the hierarchical radiosity setting and also some information-theory tools applied to the visibility and radiosity of a scene.

2.1 Refinement Criteria

The radiosity method uses a finite element approach, discretising the diffuse environment into N_p patches and considering the radiosities, emissivities and reflectances constant over the patches. With these assumptions, the discrete radiosity equation [3] is given by

$$B_i = E_i + \rho_i \sum_{j=1}^{N_p} F_{ij} B_j \ , \tag{1}$$

where B_i, E_i, and ρ_i, are respectively the radiosity, emissivity, and reflectance of patch i, B_j is the radiosity of patch j, and F_{ij} is the *patch-to-patch form factor*, only dependent on the geometry of the scene. Form factor F_{ij} is defined by

$$F_{ij} = \frac{1}{A_i} \int_{S_i} \int_{S_j} F(x,y) \mathrm{d}A_x \mathrm{d}A_y \tag{2}$$

where A_i is the area of patch i, S_i and S_j are, respectively, the surfaces of patches i and j, $F(x,y)$ is the *point-to-point form factor* [4] between points $x \in S_i$ and $y \in S_j$, and $\mathrm{d}A_x$ and $\mathrm{d}A_y$ are, respectively, the differential areas at points x and y.

A hierarchical refinement algorithm [1] is used to solve the system (1). The application of a good refinement criterion is fundamental for its efficiency. Many oracles have been proposed in the literature [1,5,6,7,8,9,10,11,12] and, for comparison purposes, we review here the three most-used ones[1].

Oracle Based on Transported Power (TP). Introduced by Hanrahan et al. [1], a cheap form factor estimate $\widetilde{F}_{ij}$ which ignores visibility was used to measure the accuracy of an interaction between elements i and j. If $\max(\widetilde{F}_{ij}, \widetilde{F}_{ji})$ exceeds a given threshold ϵ, the larger of the two elements i and j is subdivided using regular quadtree subdivision. Hanrahan et al. also observed that the number of subdivisions can be reduced considerably without affecting image quality by weighting $\widetilde{F}_{ij}$ with the source element radiosity B_j and the receiver element area A_i. Weighting with the receiver reflectance ρ_i also further reduces the number of subdivisions without deteriorating image quality. Thus, the refinement criterion based on *transported power* is given by

$$\rho_i A_i \widetilde{F}_{ij} B_j < \epsilon \ . \tag{3}$$

[1] For a detailed review, consult Bekaert [13].

Oracle Based on Kernel Smoothness (KS). In order to improve on power-based refinement, the variation of the radiosity kernel, i.e., the point-to-point form factor $F(x, y)$, is taken into account. The refinement criterion based on *kernel smoothness* [6], when applied to constant approximations, is given by

$$\rho_i \max(F_{ij}^{\text{avg}} - F_{ij}^{\text{min}}, F_{ij}^{\text{max}} - F_{ij}^{\text{avg}}) A_j B_j < \epsilon \ , \tag{4}$$

where A_j is the source element area, $F_{ij}^{\text{avg}} = F_{ij}/A_j$ is the average radiosity kernel value, and $F_{ij}^{\text{min}} = \min_{x \in S_i, y \in S_j} F(x, y)$ and $F_{ij}^{\text{max}} = \max_{x \in S_i, y \in S_j} F(x, y)$ are the minimum and maximum point-to-point form factors computed with pairs of random points on both elements i and j.

Oracle Based on Smoothness of Received Radiosity (RS). Optimal refinement can be expected by directly estimating how well the radiosity $B_j(x)$, received at $x \in S_i$ from S_j, is approximated by a linear combination of the basis functions on S_i, i.e., by estimating the *discretisation error* directly. This approach was first proposed by Lischinski et al. [7]. For constant approximations, the oracle based on *smoothness of received radiosity* is given by

$$\rho_i \max\Big(F_{ij} - \min_{x \in S_i} F_j(x), \max_{x \in S_i} F_j(x) - F_{ij}\Big) B_j < \epsilon \ , \tag{5}$$

where $F_j(x) = \int_{S_j} F(x, y) \mathrm{d}A_y$ is the *point-to-patch form factor* between point x and element j.

The cheapest and most used oracle has been the TP oracle. However, it results in sub-optimal shadow boundaries and excessive refinement in smoothly illuminated areas receiving a lot of power. The KS and RS oracles were proposed as an alternative to solve this problem. However, the oracle based on kernel smoothness has the problem of unnecessary subdivisions where the kernel is unbounded, and the one based on received radiosity relies on a costly accurate computation of form factors. All in all, the additional cost invested in both smoothness-based oracles, mainly through visibility computations, may not be balanced by the obtained improvements.

2.2 Information-Theory Tools

In Feixas et al. [14,15,16], a scene was interpreted as an *information channel* [17] and the following definitions were introduced. *Discrete scene visibility entropy* was defined as

$$H_{\text{v}}^{\text{d}} = -\sum_{i=1}^{N_p} \frac{A_i}{A_{\text{T}}} \sum_{j=1}^{N_p} F_{ij} \log F_{ij} \ , \tag{6}$$

where A_{T} is the total area of the scene. The value H_{v}^{d} can be interpreted as the average *uncertainty* that remains about the destination patch of a random walk in a scene when the source patch is known. It also expresses the *information* content of a random walk.

Discrete scene visibility mutual information was defined as

$$I_{\mathrm{v}}^{\mathrm{d}} = \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \frac{A_i}{A_{\mathrm{T}}} F_{ij} \log \frac{F_{ij} A_{\mathrm{T}}}{A_j} \tag{7}$$

and it can be interpreted as the amount of information that the destination patch conveys about the source patch, and vice versa. It is a measure of the average *information transfer* in a scene.

Continuous scene visibility mutual information, independent of any discretisation, expresses with maximum accuracy the information transfer in a scene and is defined as

$$I_{\mathrm{v}}^{\mathrm{c}} = \int_S \int_S \frac{1}{A_{\mathrm{T}}} F(x,y) \log(A_{\mathrm{T}} F(x,y)) \mathrm{d}A_x \mathrm{d}A_y \ , \tag{8}$$

where S is the surface of the scene. Moreover, $I_{\mathrm{v}}^{\mathrm{d}}$ and $I_{\mathrm{v}}^{\mathrm{c}}$ can also be considered as scene visibility complexity measures and have been generalized to the radiosity setting [14,16]. These complexity measures are interpreted as the difficulty of discretising a scene: the higher the continuous mutual information, the more difficult it is to obtain an accurate discretisation and probably more refinements are necessary to achieve a predefined precision.

The following results and proposals concerning visibility were presented in Feixas et al. [14,16]: (*i*) continuous mutual information $I_{\mathrm{v}}^{\mathrm{c}}$ is the least upper bound to discrete mutual information $I_{\mathrm{v}}^{\mathrm{d}}$, (*ii*) by refining the patches, $I_{\mathrm{v}}^{\mathrm{d}}$ must increase (or remain the same), (*iii*) among different discretisations of a scene the most accurate one is the one with the highest discrete mutual information, i.e. with minimum loss of information transfer, and (*iv*) a global discretisation error can be given by $\delta = I_{\mathrm{v}}^{\mathrm{c}} - I_{\mathrm{v}}^{\mathrm{d}}$ and it expresses the loss of information transfer in a scene due to the discretisation.

3 Information-Theory Oracles

In this section, three IT oracles are presented in correspondence with the three classic oracles reviewed in section 2.1. Our IT oracles are based on the visibility *information* content of a ray hit and the visibility *discretisation error* between two patches. Similar to the radiosity equation (1), where we observe that the contribution of patch j to the radiosity of patch i is given by $\rho_i F_{ij} B_j$, in the following refinement criteria the information or the discretisation error will be weighted by $\rho_i B_j$.

Oracle Based on Transported Information (TI). From expression (6), the term $H_{\mathrm{v}_{ij}}^{\mathrm{d}} = -\frac{A_i}{A_{\mathrm{T}}} F_{ij} \log F_{ij}$ can be considered as an element of the scene entropy matrix. Each element represents the information content of a ray hit, knowing the source patch. It is always non-negative and, in general, $H_{\mathrm{v}_{ij}}^{\mathrm{d}} \neq H_{\mathrm{v}_{ji}}^{\mathrm{d}}$.

Similar to the TP oracle, the oracle based on *transported information* is given by

$$\rho_i H^{\mathrm{d}}_{\mathrm{v}_{ij}} B_j < \epsilon \ , \tag{9}$$

where the information $H^{\mathrm{d}}_{\mathrm{v}_{ij}}$ has been weighted with $\rho_i B_j$.

Oracle Based on Information Smoothness (IS). Similar to the KS oracle (4), we compute the minimum and maximum point-to-point form factor between two elements using area-to-area sampling, i.e., selecting pairs of random points on both elements. We take, then, the difference between the information values corresponding to minimum and maximum point-to-point form factor and the information term $H^{\mathrm{d}}_{\mathrm{v}_{ij}}$, which is computed using the approximation $F_{ij} \approx A_j \frac{1}{N_s} \sum_{k=1}^{N_s} F(x_k, y_k)$ [13], where N_s is the number of lines cast between elements i and j. Thus, the oracle based on the *information smoothness* is given by

$$\rho_i \max(|H^{\mathrm{d}}_{\mathrm{v}_{ij}} - H^{\mathrm{d}^{\min}}_{\mathrm{v}_{ij}}|, |H^{\mathrm{d}}_{\mathrm{v}_{ij}} - H^{\mathrm{d}^{\max}}_{\mathrm{v}_{ij}}|) B_j < \epsilon \ , \tag{10}$$

where $H^{\mathrm{d}^{\min}}_{\mathrm{v}_{ij}} = -\frac{A_i}{A_{\mathrm{T}}} A_j F^{\min}_{ij} \log(A_j F^{\min}_{ij})$ and $H^{\mathrm{d}^{\max}}_{\mathrm{v}_{ij}} = -\frac{A_i}{A_{\mathrm{T}}} A_j F^{\max}_{ij} \log(A_j F^{\max}_{ij})$.

Oracle Based on Mutual Information (MI). From definition (7), the term

$$I^{\mathrm{d}}_{\mathrm{v}_{ij}} = \frac{A_i}{A_{\mathrm{T}}} F_{ij} \log \frac{F_{ij} A_{\mathrm{T}}}{A_j} \tag{11}$$

can be considered as an element of the symmetric scene mutual information matrix, representing the information transfer between patches i and j. We can observe that negative values appear when $F_{ij} < \frac{A_j}{A_{\mathrm{T}}}$. This situation reflects a very low interaction between the two patches involved.

The information transfer between two patches can be obtained more accurately if we calculate the continuous mutual information between them. Thus, from the continuous visibility mutual information (8), the continuous information transfer between patches i and j is given by

$$I^{\mathrm{c}}_{\mathrm{v}_{ij}} = \int_{S_i} \int_{S_j} \frac{1}{A_{\mathrm{T}}} F(x, y) \log(A_{\mathrm{T}} F(x, y)) \mathrm{d}A_x \mathrm{d}A_y \ . \tag{12}$$

As we have seen in section 2.2, a general discretisation error for a scene can be given by $\delta = I^{\mathrm{c}}_{\mathrm{v}} - I^{\mathrm{d}}_{\mathrm{v}} \geq 0$, but we are interested here in the contribution to this discretisation error of the patch-to-patch interaction i.e., in the discretisation error between two patches.

The computation of (12) can also be done, as in the IS oracle, with random lines joining both elements i and j [16]. For N_s lines, we have

$$I^{\mathrm{c}}_{\mathrm{v}_{ij}} \approx \frac{A_i A_j}{A_{\mathrm{T}}} \frac{1}{N_s} \sum_{k=1}^{N_s} F(x_k, y_k) \log(F(x_k, y_k) A_{\mathrm{T}}) \ , \tag{13}$$

where x_k and y_k are, respectively, the end-points on patches i and j of the k-th line.

From expression (11), $I^{\mathrm{d}}_{\mathrm{v}_{ij}}$ can be written as

$$I^{\mathrm{d}}_{\mathrm{v}_{ij}} = \frac{A_i A_j}{A_{\mathrm{T}}} \frac{F_{ij}}{A_j} \log\Big(\frac{F_{ij}}{A_j} A_{\mathrm{T}}\Big) \tag{14}$$

Now, taking $\frac{F_{ij}}{A_j} \approx \frac{1}{N_s} \sum_{k=1}^{N_s} F(x_k, y_k)$ [13], we obtain the *visibility discretisation error* between patches i and j, $\delta_{ij} = I^{\mathrm{c}}_{\mathrm{v}_{ij}} - I^{\mathrm{d}}_{\mathrm{v}_{ij}}$, which is approximated by

$$\delta_{ij} \approx \frac{A_i A_j}{A_{\mathrm{T}}} \left(\Big(\frac{1}{N_s} \sum_{k=1}^{N_s} F(x_k, y_k) \log F(x_k, y_k)\Big) - \Big(\frac{1}{N_s} \sum_{k=1}^{N_s} F(x_k, y_k)\Big) \log\Big(\frac{1}{N_s} \sum_{k=1}^{N_s} F(x_k, y_k)\Big) \right) . \tag{15}$$

As for non-negative numbers $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ the following inequality [17] holds

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \Big(\sum_{i=1}^{n} a_i\Big) \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \ , \tag{16}$$

we obtain that $\delta_{ij} \geq 0$. Observe that δ_{ij} is symmetric: $\delta_{ij} = \delta_{ji}$.

The fundamental idea in this approach is that the discretisation error gives us the *loss* of information transfer or the *maximum potential gain* of information transfer between two elements. Hence this difference can be interpreted as the *benefit* to be gained by refining and can be used as a decision criterion. It also represents the *variation* of the radiosity kernel. The oracle based on *mutual information*[2] is given by

$$\rho_i \delta_{ij} B_j < \epsilon \ . \tag{17}$$

4 Results

We have implemented the classic and IT oracles on top of the hierarchical Monte Carlo radiosity [2] method of RenderPark [19] software. It should be noted here that our oracles can be used with any hierarchical radiosity method.

In Fig.1 and Fig.2 we present the results of comparing the classic oracles of section 2.1 with the new ones defined in section 3 for a test scene[3]. In Fig.1, a general view of the scene is shown, while Fig.2 displays a detail of it from a different perspective. Note that the chair on the bottom right with the back to the viewer in Fig.1 corresponds to the one on the right-hand side in Fig.2, which shows the shadow cast on the wall. The left column (i) of both Fig.1 and Fig.2

[2] This oracle was introduced, from a different perspective, in Feixas et al. [18].

[3] Additional results can be found in `ima.udg.es/~rigau/cggm03.html`.

corresponds to images obtained with the classic oracles, while the right one (*ii*) corresponds to the new IT-based ones.

In Fig.1.a, the TP and TI oracles have been evaluated with only one line and a cheap form factor estimator. In Fig.1.b, the KS and IS oracles have been computed with 10 random lines. In Fig.1.c, the RS oracle has been calculated with 100 lines (10 point-to-patch form factors have been computed with 10 random lines for each one), while the MI oracle has been evaluated with only 10 random lines.

Observe in the images of Fig.1.*ii* the finer details of the shadow cast on the wall by the chair on the right-hand side. Observe also the better-defined shadow on the chair on the left-hand side and the one cast by the desk. In Fig.2.*ii* we can also see how the IT oracles outperform the classic ones (Fig.2.*i*), especially in the much more-defined shadow of the chair on the right. Observe, also, the superior quality grid created on top of the table by the IS and MI oracles (Fig.2.(b,c).*ii*) as opposed to the classic ones (Fig.2.(b,c).*i*). Comparing our three IT oracles we conclude that TI is an effective, cheap cost oracle and that IS gives a very good quality mesh, although it is outperformed by the MI oracle with the same cost.

5 Conclusions

In this paper we have presented three new information-theory oracles for hierarchical radiosity. These oracles are based on the information content of a ray between two elements (transported information and information smoothness) and the loss of information transfer between them (mutual information). They have been compared with their counterpart classic ones: transported power, kernel smoothness, and received radiosity smoothness. The results obtained show the better behaviour of the IT oracles, confirming the usefulness of the information-theory approach to deal with the radiosity problem.

Acknowledgments

This project has been funded in part with grant numbers TIC-2001-2416-C03-01 of the Ministry of Science and Technology (Spanish Government) and 2001-SGR-00296 and ACI2002-29 of the Ministry of Universities, Research and the Information Society (Catalan Government). All the images have been obtained with the *RenderPark* [19] software (`www.renderpark.be`).

References

1. Hanrahan, P., Salzman, D., Aupperle, L.: A rapid hierarchical radiosity algorithm. Computer Graphics (Proceedings of SIGGRAPH'91) **25** (1991) 197–206 Held in Las Vegas (NV), USA.
2. Bekaert, P., Neumann, L., Neumann, A., Sbert, M., Willems, Y.D.: Hierarchical Monte Carlo radiosity. In Drettakis, G., Max, N., eds.: Rendering Techniques'98 (Proceedings of the 9th Eurographics Workshop on Rendering), New York (NY), USA, Springer-Verlag Vienna-New York (1998) 259–268 Held in Vienna, Austria.

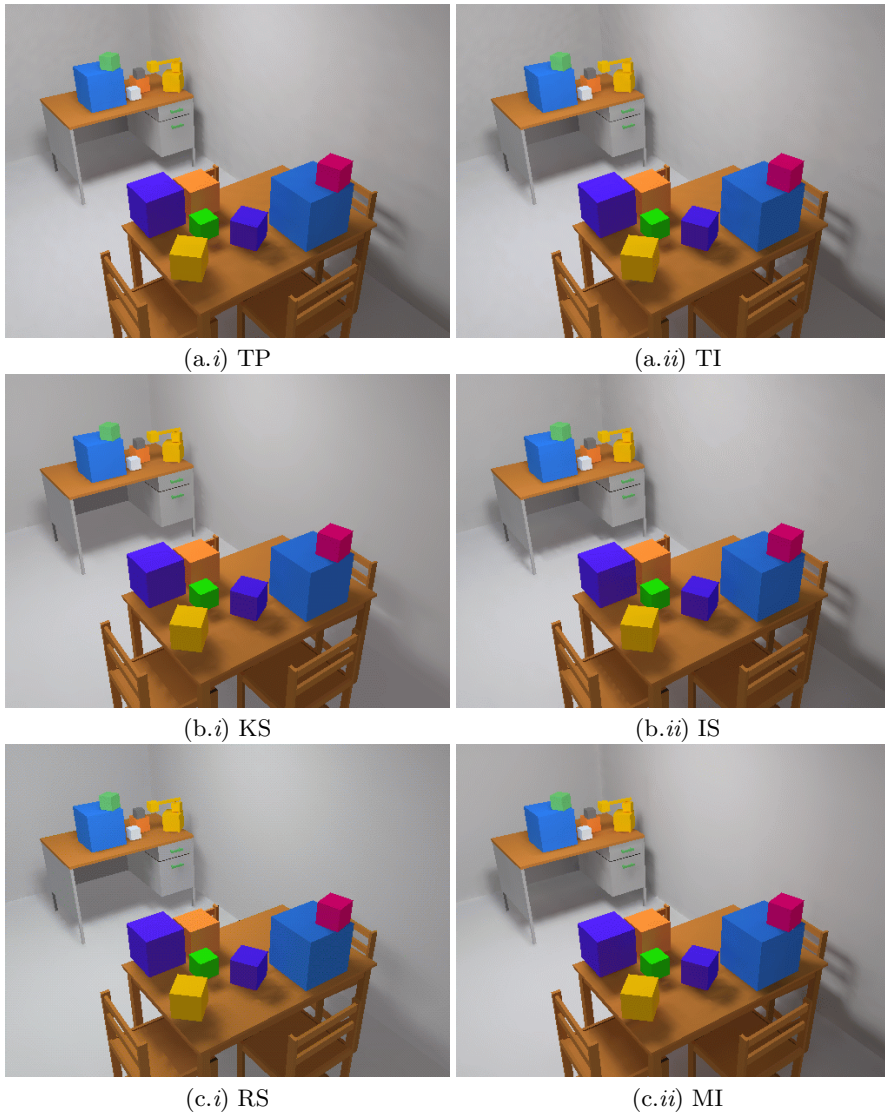
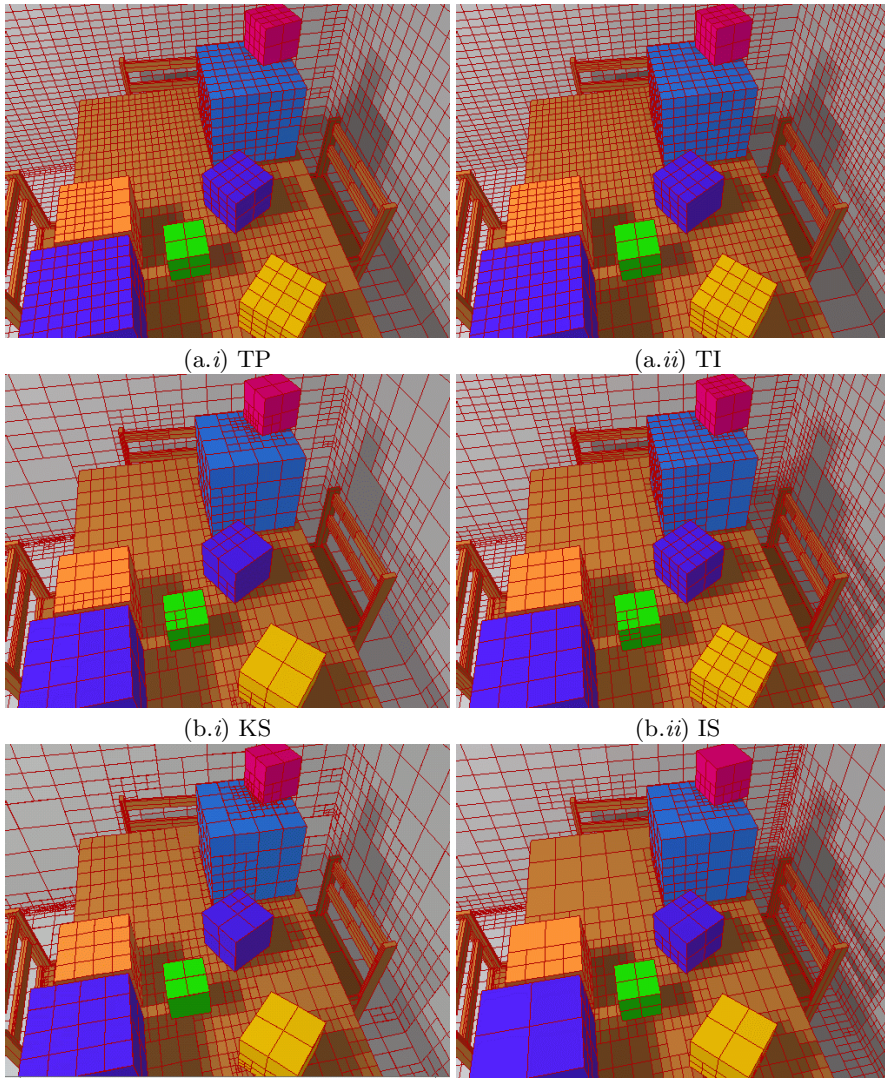


Fig. 1. Comparison of classic (*i*) vs. IT (*ii*) oracles with a Gouraud shaded solution: (a.*i*) transported power (TP) vs. (a.*ii*) transported information (TI), (b.*i*) kernel smoothness (KS) vs. (b.*ii*) information smoothness (IS), (c.*i*) received radiosity (RS) vs. (c.*ii*) mutual information (MI). In (a), the TP and TI oracles have been evaluated with only one line and a cheap form factor estimator. In (b), the KS and IS oracles have been computed with 10 random lines. And in (c) the RS oracle has been calculated with 100 lines (10 point-to-patch form factors have been computed with 10 random lines for each one), while the MI oracle has been evaluated with only 10 random lines. obtaining also 19000 patches. In each case, a total of 268500 rays are cast for the radiosity computation, obtaining approximately 19000 patches



(a.*i*) TP (a.*ii*) TI

(b.*i*) KS (b.*ii*) IS

(c.*i*) RS (c.*ii*) MI

Fig. 2. Different perspective of the scene in Fig.1, showing the grid obtained in the subdivision. The chair on the right-hand side, casting a shadow on the wall, corresponds to the one on the bottom right with the back to the viewer in Fig.1

3. Goral, C.M., Torrance, K.E., Greenberg, D.P., Battaile, B.: Modelling the interaction of light between diffuse surfaces. Computer Graphics (Proceedings of SIGGRAPH'84) **18** (1984) 213–222 Held in Minneapolis (MN), USA.
4. Sillion, F.X., Puech, C.: Radiosity and Global Illumination. Morgan Kaufmann Publishers, San Francisco (CA), USA (1994)
5. Smits, B.E., Arvo, J., Salesin, D.: An importance-driven radiosity algorithm. Computer Graphics (Proceedings of SIGGRAPH'92) **26** (1992) 273–282 Held in Chicago (IL), USA.
6. Gortler, S.J., Schröder, P., Cohen, M.F., Hanrahan, P.: Wavelet radiosity. Computer Graphics (Proceedings of SIGGRAPH'93) **27** (1993) 221–230 Held in Anaheim (CA), USA.
7. Lischinski, D., Tampieri, F., Greenberg, D.P.: Combining hierarchical radiosity and discontinuity meshing. Computer Graphics (Proceedings of SIGGRAPH'93) **27** (1993) 199–208 Held in Anaheim (CA), USA.
8. Pattanaik, S.N., Bouatouch, K.: Linear radiosity with error estimation. In Hanrahan, P., Purgathofer, W., eds.: Rendering Techniques'95 (Proceedings of the 6th Eurographics Workshop on Rendering), New York (NY), USA, Springer-Verlag Vienna-New York (1995) Held in Dublin, Eire.
9. Lischinski, D., Smits, B., Greenberg, D.P.: Bounds and error estimates for radiosity. Computer Graphics (Proceedings of SIGGRAPH'94) **28** (1994) 67–74 Held in Orlando (FL), USA.
10. Bekaert, P., Willems, Y.D.: Error control for radiosity. In Pueyo, X., Schröder, P., eds.: Rendering Techniques'96 (Proceedings of the 7th Eurographics Workshop on Rendering), New York (NY), USA, Springer-Verlag Vienna-New York (1996) 153–164 Held in Vienna, Austria.
11. Stamminger, M., Slusallek, P., Seidel, H.P.: Bounded radiosity-illumination on general surfaces and clusters. Computer Graphics Forum (Proceedings of Eurographics'97) **16** (1997) 300–317 Held in Budapest, Hungary.
12. Holzschuch, N., Sillion, F.: An exhaustive error-bounding algorithm for hierarchical radiosity. Computer Graphics Forum **17** (1998) 197–218
13. Bekaert, P.: Hierarchical and Stochastic Algorithms for Radiosity. PhD thesis, Department of Computer Science, Leuven, Belgium (1999)
14. Feixas, M., del Acebo, E., Bekaert, P., Sbert, M.: An information theory framework for the analysis of scene complexity. Computer Graphics Forum (Proceedings of Eurographics'99) **18** (1999) 95–106
15. Feixas, M., del Acebo, E., Bekaert, P., Sbert, M.: Information theory tools for scene discretization. In Lischinski, D., Larson, G.W., eds.: Rendering Techniques'99 (Proceedings of the 10th Eurographics Workshop on Rendering), New York (NY), USA, Springer-Verlag Vienna-New York (1999) 95–106 Held in Granada, Spain.
16. Feixas, M.: An Information-Theory Framework for the Study of the Complexity of Visibility and Radiosity in a Scene. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Barcelona, Spain (2002)
17. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications (1991)
18. Feixas, M., Rigau, J., Bekaert, P., Sbert, M.: Information-theoretic oracle based on kernel smoothness for hierarchical radiosity. In: Short Presentations (Eurographics'02). (2002) 325–333 Held in Saarbrücken, Germany.
19. Computer Graphics Research Group: RenderPark: A Photorealistic Rendering Tool. Katholieke Universiteit Leuven, Leuven, Belgium. (2000)

A Hybrid Scheme for Interactive Rendering a Virtual Environment

Tong-Yee Lee, Ping-Hsien Lin, and Tz-Hsien Yang

Department of Computer Science and Information Engineering,
National Cheng-Kung University, Tainan, Taiwan, Republic of China
tonylee@mail.ncku.edu.tw
http://couger.csie.ncku.edu.tw/∼vr/CGGM_IBR_2003.html

Abstract. A virtual environment can consist of both simple and complex 3D objects. The number of polygons for complex objects can dominate the total number of polygons in a scene. In this paper, we create depth images for these complex objects to allow interactive rendering a virtual environment. Arbitrary views of these complex objects are synthesized using these pre-rendered depth images as well as original polygons in the scene. To reduce the number images stored, we choose the smallest set from a large set of initial images of these objects. We show the preliminary results from applying the proposed method to accelerate the interactive walkthrough of a virtual environment.

1 Introduction

A virtual environment consists of many 3D objects to be navigated. In such environments like art galleries, there is usually a common ground upon which the 3D objects are distributed. The viewers can navigate around this ground. Virtual environments can consist of simple and complex objects. Usually, complex objects are the major scene spots in a virtual environment. The total number of polygons for these complex objects can become a bottleneck in the rendering performance. However, the number of complex objects is usually much fewer than the number of simple objects. The purpose of this study was to achieve interactive navigation of a virtual environment on a PC platform. In computer graphics, many possible approaches are available for quickly rendering virtual environments including visibility culling, levels of detail and image-based rendering[1]. The proposed method belongs to image-based approach. In the past, a number of image-based approaches have been proposed. A summary of the most related methods is described as follows.

Shade et al.[2] invented image caching technique to accelerate walkthroughs of complex scenes. Chen et al.[3] used a structured set of images to approximate arbitrary views of an object or scene using morphing. Darsa et al.[4] reconstructed meshes from a dense range of maps to approximate a 3D image warp. Shade et al.[5] used the Layered Depth Image (LDI) to deal with the occlusion artifacts in image warping. Later, Chang et al.[6] proposed the LDI tree that combines a hierarchical space-partitioning scheme with the LDI concept.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 285–294, 2003.
© Springer-Verlag Berlin Heidelberg 2003

This scheme preserves the sampling rates of the reference images by adaptively selecting an LDI in the LDI tree for each output pixel.

In this paper, all reference images were pre-rendered from synthetic scenes. We automatically segmented complex scenes into object clusters that could be represented using image-based rendering. Our idea is somewhat similar to Quicktime VR's object model[3], but we used different approaches to manage the object clusters for complex objects. Each reference image contains depth and color information on each pixel. For each image, a mesh composed of connected components, each corresponding to a piece of continuous surface from the sampled geometry, is created. We propose a technique for segmenting different pieces of the surface that reduces the "tear" effect or the so-called "rubber sheet effect".

2 Acquiring Depth Images

As a preprocessing step, we adopt Shade et al's approach[2] to build a BSP-tree like partition of a given virtual environment. This BSP-tree is built only for the complex objects in our design. In addition, each non-leaf node holds a set of reference images where cameras are placed along the camera-sampling circle. Shade et al's approach[2] looks for nodes with an aspect ratio as close to one as possible. This approach attempts to avoid long rectangular shaped nodes. The long rectangular nodes always need larger camera sampling circles to encapsulate the entire node. Larger circles cannot allow us to place viewpoints as close to the node as possible. This approach does not always guarantee the "aspect ratio" of nodes to be one. Therefore, if the "aspect ratio" of a node is large (more than 2.0) or small (less than 0.5), in our design, this non-leaf node will be treated as a *dummy* node and it will not hold a set of reference images in the proposed hierarchy. A *dummy* node reconstructs new views from the reference images held by its child or descendant nodes.

In our implementation, each bound volume is a sphere. Let d be the radius of a node's bounding sphere. The radius of a camera-sampling circle is found by $r = d * \cot(\theta)$, where $\theta = \frac{fov}{2}$ and the view frustum of the camera is *fov*. All cameras for the reference images are placed along this circle, looking toward the center. What is the smallest number of cameras (reference images) required along the camera-sampling circle to generate arbitrary views of complex objects using image-based rendering? This is a very difficult problem. Given this difficult problem, what can be done? We therefore decided to solve a less difficult problem. We initially placed a large set of cameras along the camera circle, evenly distributed. We will assume these initial images are dense enough to generate all possible views using IBR techniques. We then look for a set of images from these initial views such that both their number and the overall synthesis errors are small. The success of the proposed method depends on the initial number of views taken before optimization is sufficient to represent the object. The proposed selection procedure is stated below:

Initially, along a camera-sampling circle, a set of cameras denoted as $P = \{P_i | i = 1, 2, ..., N\}$ are placed in a uniform manner (i.e., one camera per $\frac{360}{N}$ degrees). The following parameters are defined:

- P_i denotes the ith camera placed on the circle and C_i denotes the image rendered at P_i.
- I_{mnk} denotes the rendered image using C_m and C_n to approximate C_k by the proposed IBR method, where $m < k < n$.
- ε_{mnk} denotes the image synthesis error using I_{mnk} to approximate C_k and is defined below.

$$\varepsilon_{mnk} = w_1 H + w_2 D. \tag{1}$$

In equation (1), H and D denote the number of pixels in I_{mnk} where the difference in color and depth in comparison with those of C_k is over a selected threshold and w_1 and w_2 are weights for H and D, respectively. We denote the error sum using E_{mn} for images C_m, C_n to render all images using P_k, where $k = m + 1, m + 2...n - 1$ as follows.

$$E_{mn} = \sum_{k=m+1}^{n-1} \varepsilon_{mnk}. \tag{2}$$

Assume that the selection procedure picks up $|S|$ from the initial N images, where $S = \{P_{S_1}, P_{S_2}, ..., P_{S_n} | S \subseteq P\}$ and each P_{S_i} is a selected position from these N initial positions. We define the error function for selecting these $|S|$ reference images using equation (3).

$$E = \sum_{\bigcup S} E_{P_{S_i} P_{S_{i+1}}}. \tag{3}$$

Where P_{S_i} is in clockwise order and $E_{P_{S_i} P_{S_{i+1}}}$ (equation (2)) is the error sum for selecting $C_{P_{S_i}}$ and $C_{P_{S_{i+1}}}$ to reconstruct the other initial views between cameras P_{S_i} and $P_{S_{i+1}}$. Apparently, equation (3) is minimized as $|P| = |S|$. To find a minimal set of images, we need to have a constraint on $|S|$ and therefore we need to minimize equation (4).

$$\sum_{\bigcup S} E_{P_{S_i} P_{S_{i+1}}} + w_3 \times |S|. \tag{4}$$

Where w_3 is the weight of the number of selected images (i.e., $|S|$). In the course of minimizing equation (4), if the difference in color is over a threshold, we let $\varepsilon_{P_{S_i} P_{S_{i+1}} k} = \infty$ (equation (1)) to ensure the quality of the rendered image. Currently, ***dynamic programming*** techniques [7] are applied to solve equation (4) minimization. We can therefore find an optimal $|S|$ from the N initial images.

```
RenderHierachy_complexobjects(node, viewpoint)
{
    if (node outside the view frustum)
        skip this branch of the tree;
    else if (node is a leaf)
        draw original polygons of the complex objects inside this node;
    else if (viewing position outside this node && node != dummy) {
        select two reference images from this node;
        //view reconstruction
        draw 3D tiny meshes from these two selected images;
    }
    else {
        RenderHierachy_complexobjects(node.leftchild, viewpoint)
        RenderHierachy_complexobjects(node.rightchild, viewpoint)
    }
}
```

Fig. 1. Rendering Algorithm.

3 Rendering Depth Images

Once the hierarchy has been constructed, the following algorithm (Figure 1) is used to display complex objects in a virtual environment. Several stages of this algorithm are now examined in detail.

3.1 Multiresolution

A hierarchy of views is established for complex objects. In some sense, multiple resolutions (partial and depending on the reference image views) of objects are built automatically. This can be considered a kind of view-dependent object *LOD*. In the above algorithm, switching among multiple resolutions automatically depends on the relationship between the node and the viewer's position. If the viewer is outside a node, the reference images stored at this node can provide enough details for the viewer. In contrast, if viewer is inside a node, its reference images could be too coarse to represent the objects seen by the viewer. In this situation, the tree must be traversed further to determine if an appropriate node exists. If a leaf node is visited, all polygons stored at this leaf node are rendered. Note that if dummy nodes (i.e., long rectangular shaped nodes) are visited, the next level of this hierarchy must be recursively visited. Z-buffering is used to resolve visibility in our system.

3.2 Rendering Using Selected Depth Images

As the hierarchy is recursively traversed, invisible nodes are culled. Frustum culling is performed by testing the bounding volume of each node against the planes of the viewing frustum. For each reference image, not all pixels contain visible surfaces and these empty pixels are skipped. For non-empty pixels, surface segmentation is performed to determine the connectedness of consecutive pixels and then mesh is generated for rendering new views. The mesh surface segmentation will be described in Section 3.3.

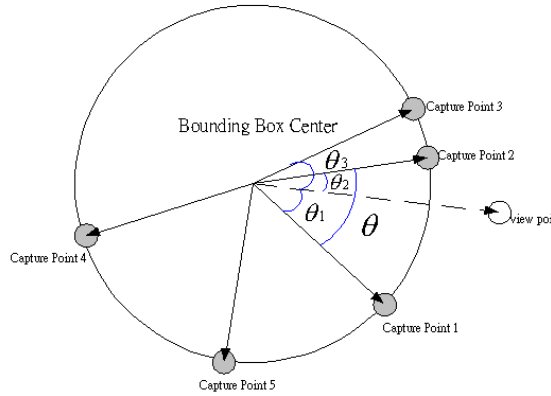


Fig. 2. Two reference images selected per node for display.

As described in Section 2, although each node holds a set of reference images, it is not necessary to render meshes reconstructed from all of these images. Only two of these images are selected to perform rendering, as shown in Figure 2. How these two reference images are selected is described as follows. First, a vector V is formed from the center of the camera-sampling circle to the current viewer position. A vector R_i is formed from this center to the image captured point CP_i. Intuitively, the two captured points that have the smallest angles between V and R_i are selected. In Figure 2, for example, captured points 2 (i.e., CP_2) and 3 (i.e., CP_3) are selected. However, this is not a good selection. A better way is to first find a vector with the smallest angle such as CP_2 and then in the counter-direction find the nearest captured point from CP_2 such as CP_1 in this example with $\theta = \theta_1 + \theta_2$ as indicated in Figure 2. If captured points CP_2 and CP_3, are selected, then θ(i.e., $=\theta_3$)$\neq \theta_2 + \theta_3$.

3.3 Mesh Surface Segmentation

Each pre-rendered image is segmented into different pieces of continuous surfaces. 3D meshes of each continuous surface are then generated by inverse projection of the connected pixels into 3D space. To reconstruct a new view, these tiny meshes associated with the two reference images are conventionally rendered using graphics hardware. For each reference image, surface segmentation can be done searching for discontinuities in disparities or depths associated with the two neighboring pixels. For consecutive pixels, if the difference in their disparities or depths is beyond a threshold, then a discontinuity is detected. Manuel et al [8] mentioned that the threshold must be changed adaptively depending on the distance of an object from the image plane.

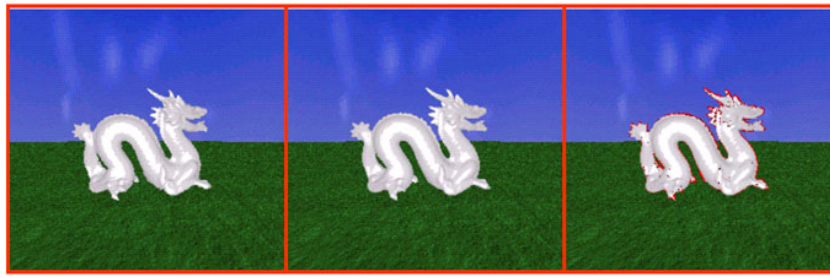
Our surface segmentation approach is very simple but effective for the common cases. On a reference image, we project two adjacent non-empty reference image samples into two neighboring reference-image planes. These two neighboring reference images are located on the same camera circle as described in

Section 2. We then check if the image-plane distance between the two projected samples is smaller than a fixed threshold. If this distance test is true for two neighboring reference-image planes, then the two samples are considered to represent the same surface. A reference-image viewpoint is very close to its neighboring reference-image viewpoints. If unfortunately we incorrectly connect two samples, the fixed thresholds that were used to segment the samples bound the error caused with the rubber sheet effect on the destination-image plane. In our practice, the threshold is computed using the computing splat size method proposed in the LDI paper [5]. In the LDI approach, for continuous surface samples, LDI computes the appropriate splat size of each sample for the destination image. Using the LDI method, we compute the splat sizes of two neighboring samples on each neighboring reference image plane. If we fail to use these two splat sizes to cover two samples (i.e., are less than the image-plane distance between two projected samples), then these two samples are segmented into two different surfaces. It is impossible, however, for any approach to solve all cases. However, the proposed approach in practice works well for the common cases. In case it does not correctly segment samples, at least, the proposed method produces smaller perceived errors bounded by the computed thresholds. Later in our experiment, we choose threshold as the maximum of (c, splat size [5]), where c is two-pixel to avoid that we might have a small splat size using [9]. In case we wrongly segment surface using c threshold, the rubber sheet effect is bounded within two pixels wide.

4 Results

The proposed methods were implemented on a Pentium II 350 PC with a 224M main memory and a GeForce2 GTS/32M graphics card. The prototype system was implemented and tested using a virtual environment originally containing about 1.5 million triangles. In this environment, many Stanford Dragon models were exhibited with about 50 thousand polygons each as the complex objects. Many other objects were exhibited as the simple objects with hundreds of polygons each. The Dragon model itself is a very complex model: it demonstrates very much self-occlusion. We can use it to fairly test the proposed methods of optimization and surface segmentation. The number of simple objects was greater than the number of complex ones. The reference images were rendered using the OpenGL library. Each reference image has 256×256 pixels. The meshes described in Section 3 were reconstructed only for the Dragons in this tested virtual environment.

In the first experiment, the dragon was placed at the center of a camera-sampling circle and viewed from different positions. In this experiment, 18 cameras (a camera per 20°) were initially placed. In this experiment, only eleven out of the 18 reference images were left ($w_1 = 10000$, $w_2 = 10$, $w_3 = 80$). For a complex model like Dragon, we want to properly sample it, so we emphasize on w_1 to reduce crack caused by poor sampling. These weights were experimentally obtained from our earlier work [9]. Our earlier work extensively evaluated



(a)Proposed method (b)Pure Polygons (c)Difference

Fig. 3. Image difference between the proposed method and rendering by the pure polygons. In (c), there are many pixels visible as red points and these pixels represent the difference between (a) and (b) over a selected threshold.

different 3D models and obtained these weights from experimental experience. Each reference image was rendered using OpenGL without highlight. Later in our walkthrough sequences, the highlight was turned on and the sky and ground were included. To judge the quality of our methods, we create an animation sequence of viewing this Dragon model along the camera-sampling circle. To view this animation sequence, please view this paper web site at

`http://couger.csie.ncku.edu.tw/~vr/CGGM_IBR_2003.html`

Figure 3 shows an example of the proposed method for reconstructing a view. In (a) and (b), we show rendered result using the proposed method and using pure polygons. In (c), there are many pixels visible as red points and these pixels represent the difference between (a) and (b) over a selected threshold. Next, we will experimentally verify the proposed method for surface segmentation. Figure 4 (a) shows "tear" or "rubber sheet" effect due to the wrong surface segmentation, (b) segmented results by the proposed method and (c) is rendered result by original geometry. Comparing (b) and (c), the proposed method does not have too much difference perceived by visual sense. We should mention that we also used $|\delta_i - \delta_{i-1}| = |\delta_{i+1} - \delta_i| + \varepsilon$ [7] to perform a similar segmentation, where δ_i, δ_{i-1}, δ_{i+1} were the disparity values for three consecutive pixels and ε was the disparity threshold. However, due to much self-occlusion of Dragon model, we failed to use this heuristics to segment surface by a unique ε. But, in contrast, the proposed method can segment surface well. In case the proposed method unfortunately makes wrong surface segmentation, the effect of "rubber sheet" is within a threshold. In the remaining experiments, we used the proposed approach to segment all reference images and we find that "tear" effect is always eliminated successfully.

In tested virtual environment, our hierarchy consists of six levels. We initially placed 18 reference images per node (Stanford Dragon Model) on the lowest level of this hierarchy. We increase the number of reference images per node on the upper part of this hierarchy as the radii r of their camera sampling circles increases. Currently, when r is increased by double, we will increase 10% of



(a)Tear effect (b)segmented by our method (c)original model

Fig. 4. Surface segmentation using the proposed method.

number of initial reference images per node on the upper parts. Since the size of camera circles on the upper part is much larger than those of the lower parts, we increase the number of initial cameras. Under this arrangement, the initial number of images for root node is 53. After optimization, the amount of image storage is reduced to about 53% of the storage of original images.

We rendered a 2500-frame walkthrough path through the test scene (see some of the whole animation sequences at the same above paper web site) and each frame was rendered at a resolution of 256×256. The plots in Figure 5 show the rendering speedups of the proposed method in relative to the performance obtained from rendering all of the original geometry contained in visible leaf nodes. In this experiments, there were two variants implemented and tested: method 1 and method 2. The method 1 used algorithm presented in Section 3. The method 2 does not adaptively render hierarchy from root and it statically renders the meshes stored on the lowest levels instead. On the average, the method 1 yields 15 to 20 times faster (8-11 fps) and the method 2 yields 3 to 10 times (2-5 fps) faster speedup for the entire sequence. In this plot, we can see a very sharp drop from 73 to 30 times in the sequence from 1 to 200. At this interval, the viewer is marching toward the arrays of Dragons and most of Dragons are with the view frustum. Therefore, the proposed method is significantly faster. Then, we will see the proposed method is 10 to 15 times faster as the viewer walks into the crowds of Dragon models. This can be easily explained, since we render some Dragon models at full details, many nodes on the lower or lowest levels and few nodes on the upper levels. So, the performance is not significantly faster. We should note that the Dragons are in proximity to each other in this tested scene. Most Dragons are drawn using the higher level of hierarchy. Obviously, the benefit of the proposed IBR implementation occurs when the number of tiny polygons reconstructed is much less than the number of original poly-gons rendered for the selected reference images. Otherwise, there is less benefit and even this situation performs worse. Currently, we not yet perform any simplifications on meshes stored on each level. We expect the reported performance can be increased further if tiny mesh simplification is implemented.

The purpose of the method 2 is to know how faster we perform using the lowest level of hierarchy. The detail of geometry on this level is the closest to original geometry than the other levels. We are interested in knowing how the

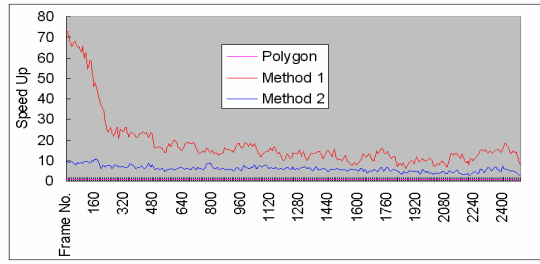


Fig. 5. Speedup: the proposed methods v.s. rendering by pure polygons.

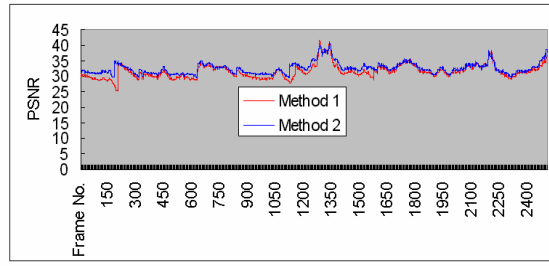


Fig. 6. PSNR (DB) for two variants of the proposed method.

method 1 and method 2 behave in their image qualities. A plot in Figure 6, we used the PSNR (peak signal-to-noise, unit is db) value to analyze the quality of images by the proposed method versus by pure polygon rendering. The PSNR is a popular method to evaluate the quality of image in image processing. We should note that PSNR might not be an absolutely correct method to compare two rendered images. But, it is the method we can quantitatively analyze image quality right now. There is no surprise to see that the PSNR increases when we always render our test scene by the images held on the lower nodes instead of those of the upper nodes. But, to our surprise, the PSNR for both method 1 and method 2 are very close and on the average are about 30-33 db. This result is very encouraging and implies that the proposed method appears to be successful in term of quality by rendering a hierarchy of views. We do not need to render all geometry at full details or the less details on the lowest levels. Please see some of the whole animation sequence at the same above web site.

5 Conclusions and Future Work

This paper introduced a hybrid image-based method for reducing the rendering computation for virtual environments. We proposed a hierarchy of views and in this hierarchy in which each non-leaf node holds a set of selected reference images. During rendering, the switching between IBR and geometry based rendering is a function of the viewer's relationship to the node in a hierarchy. Our IBR method

calculates novel views by rendering meshes reconstructed from the reference images. To reduce the number of reference images, we presented an optimized method for selecting some of the initial images. A simple but effective method for searching surface discontinuities on the reference images and to avoid the "tear" effect was presented. Experimental results show the proposed method can yield an interactive rendering performance for the test virtual environment. There are several extensions to be done in near future as follows. Currently, the viewer is limited to the ground plane in that the reference images per node are captured on a 2D camera-sampling circle. All cameras are placed on the same plane. This restricts the viewer's movements as well as the visual quality if the viewer is allowed to move freely. In the future, we plan to extend the camera-sampling circle to a sphere in 3D. If the initial number of images is not sufficient to generate arbitrary views, we should admit that the crack due to undersampling still occurs. A new and better approach will be designed for this issue.

Acknowledgement

The authors would like to thank the National Science Council, R.O.C. under Grant No. NSC 91-2213-E-006-077 for the partially support of this work.

References

1. Tomas Moller and Eric Haines, "Real-Time Rendering," A.K. Peters Ltd., ISBN 1-56881-101-2, 1999.
2. Jonathan Shade, Dani Lischinski, David H. Salesin, Tony DeRose and John Snyder, "Hierarchical Image Cach-ing for Accelerated Walkthrough of Complex Environments," in Proceedings of SIGGRAPH 1996, pages 75-82.
3. S.E. Chen, "Quicktime VR-an image-based approach to virtual environment navigation," in Proceedings of SIGGRAPH 1995, pages 29-38.
4. Darsa, Lucia, Bruno Costa Silva, and Amitabh Varshney, "Navigating Static Environments Using Image-Space Simplification and Morphing," Proceesings of 1997 Symposium on Interactive 3D Graphics, April 27-30, 1997, pp. 25-34.
5. Jonathan Shade, Steven Gortler, Li-wei He and Richard Szeliski, " Layered Depth Images," in Proceedings of SIGGRAPH 1998, pages 231-242.
6. Chang, Chun-Fa, Gary Bishop, Aneslmo Lastra, "LDI Tree: A Hierarchical Representation for Image-based Rendering," In Proceedings of SIGGRAPH'99, pp.291-298.
7. Thmomas H. Cormen, Charles E.Leiserson and Ronald L. Rivest, "Introduction to Algorithm," The MIT Press, 1990, ISBN 0-07-013143-0 (McGraw-Hill).
8. Manuel M. de Oliveira Neto, and Gary Bishop, "Dynamic Shading in Image-based Rendering," technical report: TR98-023, May 31, 1998, University of North Carolina at Chapel Hill.
9. T.C. Chen, "Dynamic Image-based Rendering techniques," Master Thesis, July 2000, Computer Graphics Group/Visual System Lab. Dept. of Computer Science and Information Engineering, National Cheng-Kung University, R.O.C., Taiwan.

Fast Adaptive Selection of Best Views

Pere-Pau Vázquez[1] and Mateu Sbert[2]

[1] Dept. LSI - Universitat Politècnica de Catalunya,
Campus Sud - Ed. ETSEIB, Diagonal 647, 8ena Planta, E - 08028 Barcelona, Spain.
ppau@lsi.upc.es
[2] IIiA, Universitat de Girona
Campus Montilivi, EPS, E-17071 Girona, Spain
mateu@ima.udg.es

Abstract. Automatic computation of best views of objects is very useful. For example, they can be used as the starting point of a scene exploration, or to enrich galleries of objects available through Internet by adding an image a model that may help to decide if it is worth downloading. To select the most interesting viewpoint of an object, we use the so-called *viewpoint entropy*. The best view is the one which gives the most information of the object being inspected. In this paper we present an adaptive method to compute best views. Our adaptive scheme allows to improve over previous approaches the time of the selection of best views by an order of magnitude, and achieve a nearly interactive rate.

1 Introduction

The automatic selection of best views of objects may be very useful. Best views can be set as the starting point of a scene exploration, for example in medical applications. They can also be used to reduce the time devoted inspecting 3D models and to understand them. The simple addition of informative images of models might avoid to spend longtime downloading (translating) and inspecting objects. Furthermore, applications to games to aid in camera control are also possible. We present in this paper a new method for the adaptive selection of best views. Our algorithm is fast and can obtain a good view of an object in a couple of seconds.

The rest of the paper is organized as follows. In Section 2 we analyze previous work on good view selection. Section 3 explains the previous brute-force approach that uses viewpoint entropy. In Section 4 we present our new algorithm. In Section 5 we show the results and discuss them, and finally, in Section 6 we conclude and point out possible future work.

2 Previous Work

Colin [1] presents a method to select a good view to observe a scene modeled with an octree. This method chooses the view which shows the highest amount of voxels, according to two different visibility measures, one exact and another

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 295–305, 2003.
© Springer-Verlag Berlin Heidelberg 2003

one approximate. Kamada and Kawai [2] consider a viewing direction to be good if in a projection, parallel line segments on a plane in 3D project as far away from each other as possible. Intuitively, the viewer should be as orthogonal as possible to every face of the 3D object. As this is not possible, they suggest to minimize (over all the faces) the maximum angle deviation between a normal to the face and the line of sight from the viewer. However, this method fails when comparing scenes with equal number of degenerated faces and it does not ensure that the user will see a large amount of detail [3]. Plemenos and Benayada [4] extend Kamada's definition. They consider a direction to be good if it minimizes the maximum angle deviation between a normal to the face and the line of sight from the viewer *and* it also provides a high amount of detail. If only the first condition is satisfied, it does not ensure that the view will not hide important information of the scene. Therefore, they add another parameter to the maximizing function which accounts for the detail seen. The parameter added is the number of visible faces from a viewpoint. Then, a function which combines both parameters is created. Moreover, an adaptive method is implemented, but the refining criterion is the number of visible faces. Barral *et al.* [3] present a method for the automatic exploration of objects or scenes. This method is based on the good view concept definition presented in [4]. In this case, the goodness of a view is computed by defining a new importance function that depends on the visible pixels of each polygon. An extension of this method can be found in Dorme [5], where an extra parameter, the so-called *exploration parameter* is added in order to avoid taking into account faces that have already been shown to the user. However, they admit that they have not been able to determine a good weighting scheme for the different factors. This causes some problems with objects containing holes, as these are not captured properly by the algorithm.

Marchand and Courty have presented a general framework that allows the automatic control of a camera in a dynamic environment [6]. The proposed method is based on the *image-based control* or visual servoing approach. Bourque and Dudek [7] describe a system for the automatic construction of image-based virtual realities to represent a real environment. The criterion used to determine the importance of a view is based on human attention. In the robotics literature, the goal of selecting a small set of cameras which allow to observe all object surfaces has been studied under the name of sensor planning. For example Wong *et al* [8] present an algorithm that searches all possible viewpoints, and selects the next best view as the one that can carve the most empty space voxels. Their system computes the next best view by optimizing an objective function that measures the quantity of unknown information in each of a group of potential viewpoints. Although the system is effective, such an approach may also result in views that observe surfaces at very oblique angles, which is undesirable in some areas such as Image-Based Rendering, as it yields poor sampling of colour in those surfaces. Freeman has exploited the *generic viewpoint assumption* to address shape from shading problems. The *generic viewpoint assumption* states that an observer is not in a special position relative to the scene [9]. It is commonly used to disqualify scene interpretations that assume special viewpoints, thus, it can be used to

avoid ambiguities [10]. The Design Galleries^{TM} (DG) system [11] is a method to automatically set parameters for computer graphics and animation. They focus on the problem of parameter tweaking. The parameters which are studied by this technique are: light selection and placement for image rendering; opacity and colour transfer-function specification for volume rendering; and motion control for particle-system and articulated-figure animation.

Vázquez *et al.* [12] have presented a measure, *viewpoint entropy*, based on Shannon's entropy [13]. Gumhold [14] has presented a method for the automatic light source placement which also uses an entropy-based function. The function is intended to measure the information coming from the illumination of a scene. Some experiments with users have been carried out in order to improve the measure with perceptual information. An optimization method for the automatic light positioning is also presented.

3 Best View Selection Algorithm

Given a certain viewpoint, the goodness is evaluated by using the *viewpoint entropy* function, that is:

$$I(S,p) = -\sum_{i=1}^{n} p_i \log p_i \tag{1}$$

where the logarithms are taken in base two, and p_i is the relative projected area (A_i/A_t) of face i (A_i is the projected area of face i and A_t is the total projected area). The best view of an object will be the most informative one, the one with maximum viewpoint entropy. Determining the best view is not easy. A first approach presented in Vázquez *et al.* [12] uses a brute-force algorithm that analyzes the entropy in a dense set of viewpoints placed all around the object. The view with maximum entropy is selected as the best view. This method is depicted in Algorithm 1.

Algorithm 1 Computes the view with the highest entropy of an object.

```
Select a set of points placed in regular positions all around the object
maxI ← 0
viewpoint← 0
for all the points do
  aux ← Compute the viewpoint entropy
  if aux> maxI then
    maxI ← aux
    viewpoint ← current point
  end if
end for
Write maxI and viewpoint
```

The use of a brute-force method is justified by the essence of the entropy function. It is not continuous and makes therefore difficult to predict maximum

reachable entropy values in the neighborhood of already analyzed points. The denser the set of views, the smaller the probability of missing important views.

4 Adaptive Best View Selection

In Section 3, we have presented the brute-force algorithm to select the best view of an object. This algorithm ensures that the best view is not missed. More concretely, it makes sure that the possible error committed is under an user-defined threshold. This is, the smaller the distance between two points, the lower the difference in entropy, so the denser the set of views analyzed, the smaller error we will commit. This is true although the function is not continuous, as the number of different faces that we will see if we move the camera will be small if the camera movement is also small. The user decides the number of views to analyze, the higher the number of views the smaller the probability of missing the best view. However, this algorithm is very demanding, as every view computation the OpenGL buffer must be read back and processed. In order to accelerate the computation of best views, an adaptive method is compulsory.

In [14] an adaptive method that predicts the best position for an entropy-based measure is developed. In this case the authors define the lighting entropy as a method to measure the information captured from viewpoint coming from a lit scene. It is used to determine the best light positions in a scene given an user position and viewing direction. In order to use a global optimization method, the authors assume that the lighting entropy function at the resolution they use is Lipschitz continuous.

We address the problem by a totally different perspective by the design of an adaptive system. We will start from a coarse set of views that will be recursively refined depending on the visible faces of each view. We will take advantage of the fact that two nearby positions usually *see* similar sets of polygons to *estimate* entropy values of new positions. The program will estimate an entropy in a new position by using the entropies of its neighbors and the set of faces that can be seen from the set of neighbor points. We have applied our method for single objects but a similar adaptive scheme can be designed for indoor scenes. Initially, six cameras are placed around the object in orthogonal positions (see Figure 1). At these positions the entropy is measured and stored. Moreover, we also store an array with the projected area of each face visible from the camera position. Once these six views are calculated, the set of points is triangulated and we predict the entropy

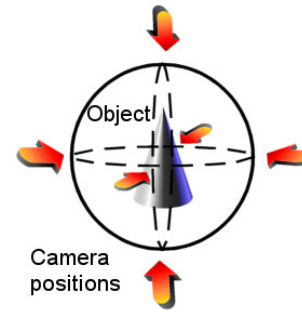


Fig. 1. The six initial camera positions around an object.

that could be captured from the middle point of the edges using a conservative estimator. To build this estimator we make the following assumptions:

- The set of visible faces from the new position is the union of faces that can be seen from the two endpoints of the edge.
- We see the faces at a better projection than in the neighbors.

The first assumption is used to ensure that we are not going to miss any face from the new view. For nearby points, a new view placed in the middle of them is likely going to see almost all the faces seen by two views. The second assumption is justified by the fact that in some situations, the number of visible faces does not increase but the amount of projected area of those faces does grow. These conditions make our estimation conservative in order not to miss important points. Although a new position could show faces that are not seen from the neighbors, some of the previously seen faces will hide. Moreover, the projected area of some of the faces will also decrease. As we estimate as if all of them grew, this will compensate for the apparition of new faces. Moreover, because estimated entropy could be always higher than the entropy of the two nearby points, we avoid the selection of infinite in between views by adding a constraint: a new view will be analyzed only if the angle between the new view and the already computed ones is larger than a given threshold (we use five degrees). The algorithm stops when none of the estimated values is higher than the values of entropy correctly computed.

4.1 Entropy Estimation

For every initial camera position, we compute its entropy value (I). We also code and store in an array the contributions of the visible faces from the camera position to the entropy. To evaluate the entropy from each new view we need to know the relative projected areas of all the visible faces from that point. This information is unknown, and the only information we have relative to the new position is the distance that separates it from the two initial views, and the entropy of these neighbors (together with the contributions to entropy of each visible face from the initial views). For the purpose of building our estimator, we assume that the new view will *see* the union of faces that can be seen from the two neighbors, and that the new position sees these faces at a better projection. As the solid angle depends on the rotation angle between the initial positions and the currently estimated viewpoint, we will use this angle to estimate the new projected areas. A good estimator for the projected area A could be then: $A = (1+|sin\alpha|/2)$, where α is the angle from the new position and the neighbor. However, this results in a too pessimistic estimator in the sense that it yields too high values and therefore many views have to be then analyzed. After some tests, we found empirically that a good entropy estimator (I_p):

$$I_p = -\sum_{i=0}^{n} N f_p((1 + |sin\alpha|/2) * \log((1 + |sin\alpha|/2)/N_f) \quad (2)$$

where N_{f_p} is the *estimated number* of faces seen from the new view and N_f is the number of faces visible from one neighbor. When two neighbors see the same face, we select the term which adds a higher value to entropy.

4.2 Algorithm

We have designed an algorithm that computes quickly the best view of an object. It performs basically four steps:

1. Evaluate the viewpoint entropy of the initial views. Build a triangular mesh with the views as its vertices.
2. Predict the entropy of the middle points of each edge, using equation 1.
3. If the highest estimated value is higher than the already computed values, evaluate its real entropy and add the new view. Go to step 2.

We have chosen a set of six initial views, placed on the intersecting points of the X, Y, and Z axis with a bounding sphere that contains the object (see Figure 1). Furthermore, instead of using a triangular mesh, we use a mesh of spherical triangles, as we want all the views to be placed at the same distance from the object (see Figure 2*a*). The edges will be arcs, and the middle points of the edges are the views whose entropy is estimated adaptively. This method is sketched in Algorithm 2. The estimation of entropy is only performed for edges whose endpoints are placed at a distance over a threshold (we have used a threshold of five degrees). The algorithm stops when none of the predicted values is higher than the already computed ones.

Algorithm 2 Adaptive computation of the best view of an object.

```
Select the initial six points on the three edges
Evaluate the entropy on these points
Triangulate the set of points
Predict the entropy on the middle point of all edges
maxPred← maximum of predicted entropies
while maxPred > maximum of computed entropies do
   Compute the real entropy of the highest predicted value and insert the new view
   Predict the entropy on the middle point of the new edges
   maxPred← maximum of predicted entropies
end while
Select the view with maximum entropy
```

In Figure 2 we can see how the viewpoints are progressively inspected. In Figure 2*a* we see the initial mesh, and in Figures 2*b* to *d* we see how the views are inserted (the lighter the point, the higher its entropy). The performance of our algorithm strongly depends on the amount of views from which we have to correctly compute viewpoint entropy. This is due to the fact that view analysis requires reading back the colour and the depth buffer to main memory and then inspecting it. The complexity of both operations depends on the size of the image. In order to accelerate the computation of the viewpoint entropy we may reduce image size. This could be accomplished using two different strategies:

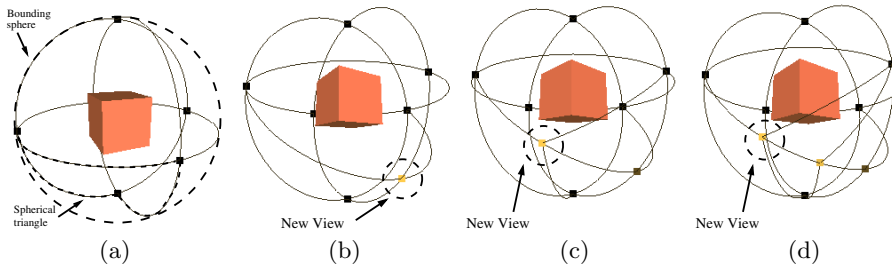


Fig. 2. The four initial steps of the best view selection for the scene of the cube.

- Reduce the resolution of the image.
- Reduce the amount of pixels read and analyzed.

The first solution could cause to worsen the precision of the entropy computation. Although the loss in precision could be unimportant, as only very small faces would disappear, we have decided to use the second strategy. A safe effective way to reduce the amount of pixels to be read back to main memory consists in predicting how much of the object *really* projects to the resulting image. This can be simply achieved by the use of a bounding box of the object. The bounding box is computed while the scene is being loaded and then used every frame that has to be analyzed to predict the real size of the projection. This process is depicted in Figure 3. The region to analyze corresponds to the projection of the bounding box of the object on the viewing plane (dashed line). This way we avoid reading all the pixels of the rendered view, which is a slow operation, and we also analyze a smaller image. This method dramatically improves the performance of each entropy evaluation thus reducing the time of the overall algorithm up to a 80%.

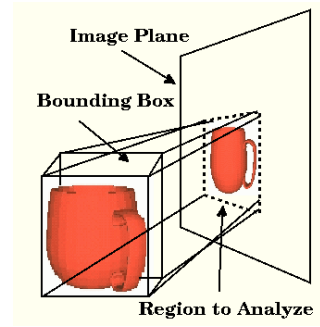


Fig. 3. Projection of the bounding box of a mug. The dashed line denotes the region that will be read and analyzed.

5 Results

The method presented above adequately computes the best views of objects of several thousands of polygons in less than one second, while the brute-force method needs one or two minutes depending on the model and the number of views analyzed. In general, the views selected by both methods are the same. Figure 4 shows the views analyzed for a set of objects: a cube, a cow, and two

chairs. The results of the new method are compared with the brute-force method in Table 1. The second column shows the computation time for the brute-force method and the last one the timings for the adaptive strategy. To achieve a similar result than with the adaptive method, the brute-force method needs to analyze about 1500 views. In this case we obtain speed-ups of up to 396:1 for the case of the cube.

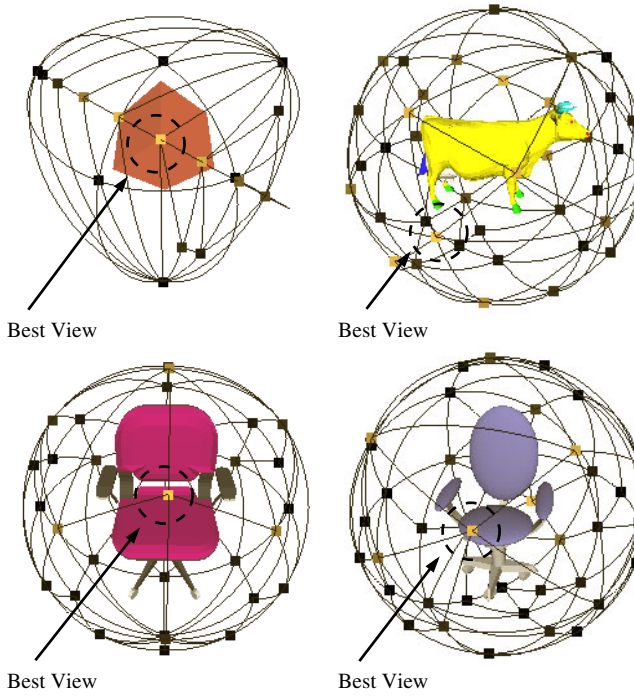
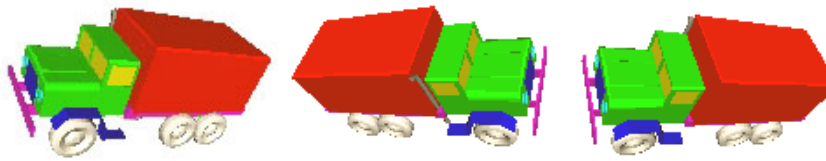


Fig. 4. The different analyzed views for each of the objects. The colour of the viewpoint encodes the entropy, the lighter the colour the higher the entropy.

Another example gives more curious results. In Figure 5*a* we can see the best view of a lorry selected with the brute-force method. Its entropy is 0.192. On the other hand, the best view using the adaptive method is the one that appears in 5*b*, whose entropy is 0.1957 slightly higher. As the lorry is practically symmetric the views are very similar. However, with the adaptive method even the second best view is better (exactly 0.1948) than the brute-force one. In this case its position is very close to the one of the best view as selected with the previous algorithm. This view is shown in 5*c*. Although throughout the calculation process we use a conservative estimator, the resulting values of estimated entropy are in a feasible range (we do not generate estimated values which are very high in comparison with the real values of entropy captured), and consequently, the system performs very well, only a small number of images has to be actually analyzed. We obtain the same best view positions than with the

Model	*Brute-Force Time (ms)*	*Adaptive Time (ms)*	*Reduction ratio*
Cube	47540	120	396:1
Cow	28320	1490	19:1
Chair 1	22170	720	31:1
Chair 2	29700	1730	17:1

Table 1. Comparison of results of the adaptive method and the brute-force method.



(a) $I = 0.192$ (b) $I = 0.196$ (c) $I = 0.195$

Fig. 5. The best views selected by our two algorithms. *(a)* was selected with the brute-force method, *(b)* and *(c)* with the adaptive. Note that even the second best view chosen by the adaptive method, *(c)*, is better than *(a)*.

brute force method, and sometimes some views that the previous algorithm had missed. Moreover we reduce the time of computation from one or two minutes for a high number of views to less than one second in most cases. As this estimation is conservative, in the sense that the predicted value will be slightly higher than the resulting entropy, the probability of missing important positions will be low. Though we have not been able to find any particular case, our system could fail if the object had such a shape that some of the faces were only visible from a special point, and this was the point of maximum entropy. If the positions close to this point saw a small number of faces and had a low entropy value, we could miss this position. However, this is not likely to happen, as some faces that are only visible from a very concrete region will project to a small area and therefore will have a low entropy rate.

6 Conclusions and Future Work

In this paper we have presented a new method for the automatic selection of best views of objects. Best views can be used as a technique to improve galleries of objects providing images of the models that help the user decide which of those are interesting. Moreover, they can be used as starting points for navigation systems. In the future we want to address the problem of view selection using perceptual issues. Instead of taking into account only the geometry of the scene, best views could also be selected according to the illumination of a scene.

This can help in automatic camera positioning for video capture or automatic navigation in realistic rendering systems.

Acknowledgements

This project has been partially supported by TIC-2001-2416-C03-01 of the Spanish government, SGR2001-00296 grant from Catalan Government, and financial support from the Ministry of Universities, Research and the Information Society from Catalan Government.

References

[1] C. Colin. Automatic computation of a scene's good views. In *Proc. MICAD*, February 1990.

[2] T. Kamada and S. Kawai. A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 41(1):43–56, January 1988.

[3] P. Barral, G. Dorme, and D. Plemenos. Scene understanding techniques using a virtual camera. In A. de Sousa and J.C. Torres, editors, *Proc. Eurographics'00, short presentations*, 2000.

[4] D. Plemenos and M. Benayada. Intelligent display in scene modeling. new techniques to automatically compute good views. In *Proc. International Conference GRAPHICON'96*, July 1996.

[5] G. Dorme. *Study and implementation of 3D scenes comprehension techniques*. PhD thesis, Université de Limoges, 2001. In French.

[6] E. Marchand and N. Courty. Image-based virtual camera motion strategies. In P. Poulin S. Fels, editor, *Proc.of the Graphics Interface Conference, GI2000*, pages 69–76, Montreal, Quebec, May 2000. Morgan Kaufmann.

[7] E. Bourque and G. Dudek. Automatic creation of image-based virtual reality. In *Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, volume 3209, pages 292–303, Bellingham, WA, October 1997. SPIE - The International Society for Optical Engineering. ISBN 0819426415.

[8] L. Wong, C. Dumont, and M. Abidi. Next best view system in a 3-d object modeling task. In *Proc. International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 306–311, 1999.

[9] W. T. Freeman. Exploiting the generic view assumption to estimate scene parameters. In *Proc. 4th International Conference on Computer Vision*, pages 347–356, Berlin, Germany, 1993. IEEE.

[10] A. L. Yuille, J. M. Coughlan, and S. Konishi. The generic viewpoint constraint resolves the generalized bas relief ambiguity. In *Proc. of Conference on Information Scienes and Systems (CISS 2000)*, Princeton University, March 15–17 2000.

[11] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 389–400. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[12] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In T. Ertl, B. Girod, G. Greiner H. Niemann, and H.-P. Seidel, editors, *Proceedings of the Vision Modeling and Visualization Conference (VMV-01)*, pages 273–280, Stuttgart, November 21–23 2001. IOS Press, Amsterdam.
[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.
[14] S. Gumhold. Maximum entropy light source placement. In *Proc. of the Visualization 2002 Conference*, pages 275–282. IEEE Computer Society Press, October 2002.

Perception-Based Illumination Information Measurement and Light Source Placement

Pere-Pau Vázquez[1] and Mateu Sbert[2]

[1] Dept. LSI - Universitat Politècnica de Catalunya,
Campus Sud - Ed. ETSEIB, Diagonal 647, 8ena Planta, E - 08028 Barcelona, Spain.
ppau@lsi.upc.es
[2] IIiA, Universitat de Girona
Campus Montilivi, EPS, E-17071 Girona, Spain
mateu@ima.udg.es

Abstract. The automatic selection of good viewing parameters is very complex. In most cases, the notion of *good* strongly depends on the concrete application. Moreover, when an intuitive definition of good view is available, it is often difficult to establish a measure that brings it to the practice. Commonly, two kind of viewing parameters must be set: the position and orientation of the camera, and the ones relative to light sources. The first ones will determine how much of the geometry can be captured and the latter will influence on how much of it is revealed (i. e. illuminated) to the user. In this paper we will define a metric to calculate the amount of information relative to an object that is communicated to the user given a fixed camera position. This measure is based on an information-based concept, the Shannon entropy, and will be applied to the problem of automatic selection of light positions in order to adequately illuminate an object.

1 Introduction

The selection of the adequate viewing parameters is a very complicated problem that is usually solved through a large process of test and error that requires long-time human dedication. Obviously, the necessity of human intervention becomes impractical for large collections of models. Moreover, many applications from scientific visualization are often employed by users with little or null experience in Computer Graphics. It is desirable then to find a way to automatically setting adequate viewing parameters. Recently, research on the automatic placement of cameras has attracted the attention of the Computer Graphics community, but only few papers focus on the important problem of correctly illuminating a scene. See

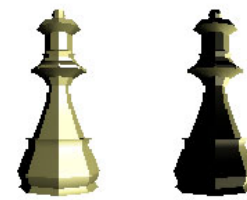


Fig. 1. Good and bad lighting of a chess queen.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 306–316, 2003.
© Springer-Verlag Berlin Heidelberg 2003

for instance Figure 1, where a good and a bad illumination of a chess queen are shown. Our aim in this paper is to present a perception-based metric that evaluates the amount of illumination information contained in a view and we apply it for the automatic selection of light positions that adequately illuminate an object or scene.

The rest of the paper is organized as follows. In Section 2 we review previous work on parameter tweaking and the related work of automatic camera placement, Section 3 presents our perception-based measure of illumination information. In Section 4 a method for the automatic positioning of a single light source is developed. We also show how the extension to several light sources is straightforward. Finally, Section 5 concludes our work and points out some lines for future work.

2 Previous Work

The selection of viewing parameters for scene rendering is a complex and tedious process. Parameter tweaking can be divided into two phases: a) Camera position and direction setting, and b) Light source selection and positioning. The problem of good camera positioning has become an active field of research mainly due to the emergence of the so-called Image-Based Rendering methods. Light source selection has usually been treated from the point of view of inverse lighting. Now we proceed to review the previous work on these two areas of research.

2.1 Camera Placement

Colin [1] presents a method to select a good view to observe a scene modeled with an octree. Kamada and Kawai [2] define a criterion for the quality of a view for orthogonal projections. Plemenos and Benayada [3] extend Kamada's definition, considering the amount of detail shown in a view as the number of visible faces. Barral *et al.* [4] present a method for the automatic exploration of objects or scenes. In this case, the quality of a view is computed by defining a new importance function that depends on the visible pixels of each polygon.

Vázquez *et al* [5] have presented a new measure based on the Shannon entropy [6], *viewpoint entropy*, to evaluate the amount of *geometric* information seen from a point. It has been successfully applied to some scientific visualization problems such as automatic selection of good views of molecular models [7]. Takeuchi and Onishi [8] measure the entropy of an image based on histograms of intensities in order to find the complex parts of a scene.

2.2 Light Source Selection

Adequate lighting selection research can be divided in two subfields: inverse lighting, and maximum information communication. Here we review recent work on these fields.

In inverse lighting, the user specifies how the scene should look like and the adequate parameters are searched. Therefore it is assumed that the user has a knowledge on the object shape and material properties. Although there is a broad bibliograpy in inverse lighting (for a survey see [9]), we only cite here some examples. Schoeneman *et al.* [10] describe an interactive system that, given a set of lights with fixed positions, determines their colours and intensities in order to match a target image painted by the designer. Kawai *et al.* [11] control light emissions and directions, as well as surface reflectances for designing the illumination in an environment rendered with a radiosity based method. As in the former case, the user has to specify how the final image should be illuminated. None of these methods automatically sets the light source positions. Costa *et al.* [12] have implemented an automatic method of light placement and intensity selection. Their objective is to obtain a configuration that determines a given radiance distribution. Although it is a powerful approach, the objective function needs to be specified by the user with a scripting language, and therefore is not easily applicable when the user is not expert. Poulin and Fournier [13] and Poulin *et al.* [14] manipulate highlights and shadows in order to define a resulting illumination. These modifications are translated to the corresponding changes to the positions of the light sources. Jolivet *et al.* [15] present a Monte-Carlo method for the selection of light positions in direct lighting. They use a declarative paradigm in order to help the users to describe in a more intuitive way the lighting goal.

Opposite to inverse lightning problems, some methods seek the adequate light sources configuration that reveals the maximum of information to the user, by means of adequately placing light sources, no matter which object or scene is inspected. The Design Galleries^{TM} (DG) system is a method to automatically set parameters for computer graphics and animation. They automatically compute and organize sets of views or animations which are perceptually different from each other. The resulting images are presented to the user to choose among them. Apart from some parameters concerning to material properties, they also study light selection and placement [16]. Gumhold has also explored the problem of automatic parameter setting [17]. He has presented a method for the automatic light source placement which also uses an entropy-based function, the *lighting entropy*. He defines the unit of information $-\log p_i$ as a function of the measured brightness of the visible pixels. The brightness of a pixel is computed as the Y tristimulus value of the CIE 1931 standarized colour model. It is calculated with the following formula: $Y = 0.21262 \cdot R + 0.71514 \cdot G + 0.07215 \cdot B$. Then, the lighting entropy is defined as $H(X) = -\sum_{i=1}^{m} p_i \log p_i$, where the probability p_i is defined as the number of pixels whose brightness falls into interval i (the logarithms are taken in base 2 and $0 \log 0 = 0$ for continuity). The author defines the unit of information $-\log p_i$ as the number of pixels that fall into interval i where index i is computed as $i = \lceil m \cdot (Y + \frac{1}{2}) \rceil$. Therefore, their measure is maximum when the number of different brightness values in an interval is uniform across the scene. The number m of intervals chosen by the author is 30. The *lighting entropy* is similar to the *viewpoint entropy* presented by Vázquez *et*

al. [5]. The latter uses as unit of information the relative projected area of each face, while the former use the normalized number of pixels that have the same brightness (in an interval). Surprisingly, the tests with users revealed that the best views as selected by his method were discarded by them due to the fact that they presented too large specular regions. With the results of the study he has improved the method taking into account their comments. Some fast methods for light positioning are also presented.

Shacked and Lischinski [18] propose a quality function formed by six terms that are weighted by the user. Their system optimizes these parameters based on a perceptual quality metric. Their objective is to effectively communicating information on the scene: shapes, materials, and their relationships. The quality metric they build is composed by six factors. Each of them is devoted to a different kind of information (such as edge detection or variance reduction), and some of them may have contradictory effects such as the histogram equalization term that, when applied, tends to increase the variance (which is controlled by the variance reduction term). Therefore, manual calibration is necessary for every scene whereas they have empirically found some weights that perform well for a certain number of models.

3 Illumination Information Measurement

3.1 Introduction

From the previous work, only Gumhold [17] has defined a fully automatic method for lighting parameter tweaking which needs no user intervention. The probability distribution of the entropy measure presented is based on the normalized number of pixels whose brightness falls in an interval. Unfortunately, such a distribution function may cause some problems. First, the use of brightness values does hide the information concerning to the colour that is perceived by the human visual system. Consequently, some colours that appear different to us are measured as the same one (see Figure 2*a*). If only a single material and a single light source (or several light sources with the same emission colour) is present in the scene, this would not be important. However, scenes with two differently coloured light sources might yield contradictory results, as an illumination which is best according with this method may be communicating a low amount of information than another one. This is exactly what happens in Figure 2*b*. The left cone is illuminated with a pink and a green light sources while the right one is is illuminated with a blue and a green light sources. When displayed, our visual system distinguishes easier between green and pink than between blue and green, and therefore the left cone communicates a higher amount of information. As blue and green on the right cone have similar brightnesses, with Gumhold's method the right cone would be chosen. Second, the normalization by the number of projected pixels does produce a lateral effect: a scaling of a model, under the same lighting conditions does yield the same entropy (see the Section 3.2).

We propose in this paper a new metric for illumination information measurement. It differs from previous approaches in that we measure the perceptual-



(a) (b)

Fig. 2. In (a) we can see two different colours which have the same Y value. (b) shows two different illuminations of a cone. The first one, with a pink and a green light sources provides a higher amount of information because those colours are more easily distinguisible for us than blue and green in the right image. However, as in the right cone blue and green have similar brightnesses and therefore some regions that appear different to us, are summed together, this one would be chosen by Gumhold's system.

based spectrum of LUV-colours. Moreover, we take into account spatial issues, that is, all the pixels with the same LUV colour will accumulate if they form a single continuous region (see Section 3.2), as otherwise they are perceived as separate sources of information by an observer.

3.2 A Perception Based Measurement of the Illumination Information

An accurate positioning of a light source must reveal as much information as possible of a scene. The problem is how to define a formula that indicates the quantity of information present in an image. Vázquez *et al.* [5] have developed an entropy based measure that can be used to determine the amount of geometric information captured from a point. This value is calculated by projecting the scene onto a bounding sphere of the viewpoint and using as distribution function the relative projected areas of each face. Therefore its value is maximum when all the faces are seen with the same relative projected area. Although this effectively solves the problem of measuring geometric information, when we render a scene the current shading may hide some faces of the object. Thus, to measure the illumination information that arrives to a point, it is necessary to build a distribution function based on the visual stimulus that arrive at this point. Moreover, as the human visual



(a) $H_L = 0.08865$ (b) $H_L = 0.06389$

Fig. 3. In (a) we can see a zoom-in of (b). If the background was not used, both would yield roughly the same quality value. With illumination entropy we have a noticeable difference.

system is limited, the measure must be sensitive to this and only take into account perceptible information. With these conditions we can build an entropy by using as unit of information the relative area of each region *whose colour is different from its surrounding*. The expression of this measure will be:

$$H_L(X) = -\sum_{i=0}^{m} A_i \log A_i,$$

where m is the number of different regions, A_i is the relative area of the region i. The background is represented by region index 0. Remark that it is important to take the background into account because the probability distribution must add up to 1. Some approaches do not use the background information and instead they normalize the probabilities dividing by the number of valid pixels (see Gumhold [17]). But this normalization may hide some information, as a zooming-in of a certain view (provided that the object still remains inside the viewing frustum) will give the same value despite we are really seeing the object better and therefore this fact should be detected (see Figure 3).

Observe that the background plays another important role. Usually, the background has a different colour than the rest of the object, nevertheless, if the scene does not contain an ambient term, it might happen that the object is completely black under a certain light position. If the background is black, the information present is zero, as we are not seeing anything. On the other hand, if the background is not black, it will help us to see the silhouette of the object, and this must be somehow taken into account. In our examples we considered the background white and therefore the silhouette of a completely dark object is not zero. In any case, the colour of the background must be taken into account when measuring the illumination entropy, as it might be necessary to add its area if some parts of the scene have the same colour in any lighting configuration. Another important problem is the colour regions measurement. Gumhold measures the lighting entropy by adding the number of pixels with the same (in an interval) brightness. However, the distribution of the luminances across the scene may also be informative. So, instead of accumulating the relative area of the pixels which have a colour that can not be distinguished by the human visual system, our method takes as information unit, the relative area of each isolated region of the same colour.

The colours are transformed to CIE LUV format. In order to detect if two neighboring colours are the same, we use the CIE LUV colour difference formula: $\Delta E^* = \sqrt{(\Delta L^*)^2 + (\Delta u^*)^2 + (\Delta v^*)^2}$. It is known that colours that have a CIE LUV difference of less than 1 appear to be the same [19], so this is the criterion we have used to distinguish between regions. For a real complex environment, the difference may raise to 6 or even more, but we are dealing here with a simpler scenario, a window in a computer screen. Note that with this criterion a region labeled with the same colour could have some pixels that compared to each other yield a difference value of more than one. In order to avoid this, when a colour is compared to a neighbor to determine if it belongs to the same region than the previously computed, the current colour is assigned the colour of the

neighbor to which it has been compared, if its difference is less than one. This would prevent a very smooth colour gradient traversing a big scene to be labeled as a single region. The use of such a probability function avoids the problem of mixing perceptible different colours in a same measure, as it happens in Figure 2. Gumhold's method would select right cone (lighting entropy of 0.1229 versus 0.1216) while our method yields a smaller value for the right cone (illumination entropy of 0.1127 versus a value of 0.1625 for the left one).

The neighborhood criterion that has been applied is the following one: Two pixels are neighbors and therefore belong to the same region if their colour difference is below one and if these pixels have an edge in common. With this method a typical checkerboard texture will not be considered as two regions but at a higher number of them, depending on the number of squares (we consider here a situation of constant shading along the texture, otherwise, the illumination will also introduce a higher number of different regions). In Figure 4 we can see this with an example. Figure 4*a* has a higher illumination entropy than Figure 4*b*, which corresponds to our perception of four regions in the first case and two in the second.



(a) $H_L = 0.1423$ (b) $H_L = 0.1308$

Fig. 4. (a) shows a checkerboard of four squares. Note that we intuitively identify four squares, while in (b) only two regions are perceived. Therefore, (a) is more informative than (b).

4 Perception Based Automatic Light Source Placement

The selection of the best position for a single light source was fully implemented as a brute-force algorithm. We place the light source at a set of different positions on a bounding sphere of the object and measure its illumination entropy. The position with the highest quality is selected. This method is general and can be applied whichever the shading algorithm is used, as the calculations are performed on the resulting image. In our case we have used OpenGL's default lighting mode. The highest cost of our method is incurred by the rendering tool and the capture of the data. Each time a new illumination has to be analyzed, the scene must be rendered and the image must be read back to main memory. However, several accelerations can be added, depending on the rendering tool. Gumhold [17] has presented a fast lighting scheme suitable for an OpenGL lighting, but other techniques can be applied if the rendering is not OpenGL based, as the quality criterion works for any kind of shading. In order to accelerate the computation we reduce the size of the window read back to main memory by reading the depth buffer at the beginning and inspecting it in order to obtain the bounding box of the object projection. Moreover, the lighting space can be restricted to the hemisphere where the user is placed as the lights placed on the other hemisphere will not illuminate most of the polygons facing the camera. This allows to obtain a proper light position in two to ten seconds for scenes of

several thousands of polygons. We have also implemented a software renderer which uses Phong lighting. Initially, the object is projected and the depth buffer is read. Then, the illumination calculations are performed only on these captured pixels. This results in a reduction of 4:1 in computation time.

As seen in [18], it is difficult to visually evaluate the quality of a rendering. On the other hand, the optizimed images with their method also yield high values of illumination entropy with our system, with the advantage that we do not need a preprocess of calibration and different tuning for each model or view. In Figure 5 we can see some examples of adequately illuminated objects.

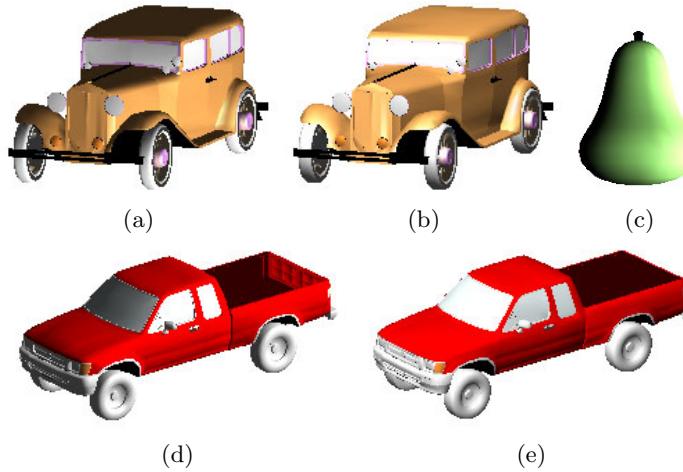


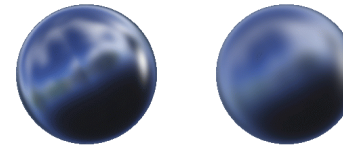
Fig. 5. Comparison of our method with Gumhold's system. (a) and (d) were generated with our method, and (b) and (e) with Gumhold's measure. Notice the highly specular regions that hide some shape detail (note the back of the car in (d) and (e)), as reported by authors. (c) shows the optimal configuration of two light sources for a pear.

4.1 Discussion

The measure we have presented here tries to maximize the information revealed to the user given an illumination of the scene. Its value is the highest when all the regions of the same colour are of the same relative area. In contrast to this, large regions of the same colour give low entropy, because of the logarithmic nature of the entropy measure. This is intuitively correct as a large flat polygon will be perceived better if there is a gradient in the illumination that shades it. The six quality terms employed by Shacked and Lischinski focus similar goals [18]. For instance, the histogram term seeks to equalize the amount of quantities of each luminance value appearing on the scene, although without taking into

consideration the part of the scene where they appear. Hence, it is not surprising to discover that the images the authors select as the best ones are often the same that with our method.

On the other hand, our measure has several advantages. It is compact and general and thus, it must not be manually calibrated for every scene, because it gives a quantity that can be compared with successive renderings of the same scene. As it is a measure of the amount of information present in an image, it can also be used to compare between similar images. See for instance Figure 6. It shows two parabolic environment maps (actually only one of their paraboloid parts). The one on the left is filtered with an exponent of 100 while the second is been filtered with an exponent of 100. They are used to simulate different glossy objects [20]. Intuitively, we can see that the amount of information present on the second one is lower than in the former, but how different, we do not know. With our method we can measure the amount of information of both maps and prove that the result corresponds with intuition.



(a) $H_L = 0.6117$ (b) $H_L = 0.5979$

Fig. 6. Two glossy spheres rendered using Phong prefiltered environment maps with exponents of 100 and 25. The higher the exponent the closer to totally specular, and the higher the entropy.

4.2 Several Light Sources

We have tested the selection of a good illumination with more than one light source. The extension of our algorithm to several light sources is straightforward. It has the disadvantage that it scales badly with the number of light sources. The bottleneck is at the rendering process and reading the generated information back to main memory. An increase in the number of light sources increases exponentially the number of views that have to be analyzed. With a single light source, it requires several seconds to find the adequate position. Therefore, for the case of several light sources, it becomes necessary to accelerate it by means of a global optimization method or an fast adaptive strategy. In Figure 5 right we can see an example of a pear optimally illuminated by two light sources.

5 Conclusions and Future Work

In this paper a perceptual based measure of the illumination information of a view has been developed. It is simple and robust and has a mathematical foundation on Information Theory. An extension to a set of light sources is also presented. Although it does not scale well with the number of light sources as the number of positions to analyze grow exponentially, some accelerations can be foreseen. In particular as the bottleneck is on the rendering process, and the viewpoint is fixed, a software renderer of the visible region reduces the

computation to a 25%. In the future we will do a deeper research on this problem. Moreover, a study with trained and not trained users is necessary to evaluate the suitability of our method for applications such as scientific visualization.

Acknowledgments

This work has been supported by TIC-2001-2416-C03-01 of the Spanish government, and SGR2001-00296 and ACI2002-29 grants from Catalan Government.

References

[1] C. Colin. Automatic computation of a scene's good views. In *Proc. MICAD*, February 1990.

[2] T. Kamada and S. Kawai. A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 41(1):43–56, January 1988.

[3] D. Plemenos and M. Benayada. Intelligent display in scene modeling. new techniques to automatically compute good views. In *Proc. International Conference GRAPHICON'96*, July 1996.

[4] P. Barral, G. Dorme, and D. Plemenos. Scene understanding techniques using a virtual camera. In A. de Sousa and J.C. Torres, editors, *Proc. Eurographics'00, short presentations*, 2000.

[5] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In T. Ertl, B. Girod, G. Greiner H. Niemann, and H.-P. Seidel, editors, *Proceedings of the Vision Modeling and Visualization Conference (VMV-01)*, pages 273–280, Stuttgart, November 21–23 2001. IOS Press, Amsterdam.

[6] R. E. Blahut. *Principles and Practice of Information Theory.* Addison-Wesley, Cambridge, MA, 1987.

[7] P.-P. Vázquez, M. Feixas, M. Sbert, and A. Llobet. Viewpoint entropy: A new tool for obtaining good views for molecules. In D.Ebert, P.Brunet, and I.Navazo, editors, *Data Visualisation 2002 (Eurographics/IEEE TCVG Symposium Proceedings)*. Eurographics/IEEE, May 27-29 2002.

[8] Y. Takeuchi and N. Ohnishi. Active vision system based on information theory. *Systems and Computers in Japan*, 29(11):31–39, 1998.

[9] G.Patow and X.Pueyo. A survey on inverse rendering problems. Technical Report IIiA 00-07-RR, IIiA, University of Girona, 2000.

[10] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with light. In James T. Kajiya, editor, *SIGGRAPH 93 Conference Proceedings*, Computer Graphics Proceedings, Annual Conference Series, pages 143–146. ACM SIGGRAPH, ACM Press, August 1993.

[11] J. K. Kawai, J. S. Painter, and M. F. Cohen. Radioptimization - goal based rendering. In *Proc. of SIGGRAPH-93: Computer Graphics*, pages 147–154, Anaheim, CA, 1993.

[12] A. Cardoso Costa, A. Augusto de Sousa, and F. Nunus Ferreira. Lighting design: A goal based approach using optimization. In *Rendering Techniques '99*, pages 317–328, New York, NY, 1999. Springer Wien.

[13] P. Poulin and A. Fournier. Lights from highlights and shadows. *Computer Graphics*, 25(2):31–38, March 1992.

[14] P. Poulin, K. Ratib, and M. Jacques. Sketching shadows and highlights to position lights. In *Proceedings of Computer Graphics International 97*, pages 56–63. IEEE Computer Society, June 1997.

[15] V. Jolivet, D. Plemenos, and P. Poulingeas. Inverse direct lighting with a Monte Carlo method and declarative modelling. 2330:3–??, 2002.

[16] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 389–400. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[17] S. Gumhold. Maximum entropy light source placement. In *Proc. of the Visualization 2002 Conference*, pages 275–282. IEEE Computer Society Press, October 2002.

[18] R. Shacked and D. Lischinski. Automatic lighting design using a perceptual quality metric. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, volume 20, pages C–215–C–226, September 2001.

[19] L. Neumann, K. Matkovic, and W. Purgathofer. Perception based color image difference. *Computer Graphics Forum*, 17(3):233–241, September 1998.

[20] J. Kautz, P.-P. Vázquez, W. Heidrich, and H.-P. Seidel. A Unified Approach to Prefiltered Environment Maps. In B. Peroche and H. Rushmeier, editors, *Eleventh Eurographics Rendering Workshop 2000*, pages 185–196. Eurographics, June 2000.

A Study on the Dynamic Painterly Stroke Generation for 3D Animation

Hyo Keun Lee, Young Sup Park, and Kyung Hyun Yoon

Department of Image Engineering
Graduate School of Advanced Imaging Science, Multimedia and Film
ChungAng University, Seoul, Korea
{xcross, cookie, khyoon}@cglab.cse.cau.ac.kr
http://cglab.cse.cau.ac.kr

Abstract. We present a technique that provides a frame-to-frame coherence in 3D non-photorealistic animations. It is considered a very important subject for non-photorealistic animations to maintain frame-to-frame coherence so that the resulting frames do not randomly change every frame. We maintain coherence by using 3D particle systems. Each particle means a brush stroke in the resulting image. Since we have located particles on the object's surface, the coherence is maintained when the object or camera is moving in the scene. Of course, the coherence is maintained when camera is zooming in/out. However, the brush strokes on the surface also zoom in/out. This result in too large or too small brush strokes that are not considered hand-crafted brush strokes. Meanwhile, frame-to-frame coherence can be preserved during camera zoom in/out by dynamically managing the number of brush stroke and maintaining its size.

1 Introduction

Computer graphics has been dominated by the goal of generating images that mimic the effect of a traditional photographic camera. However, current rendering technologies were limited to producing unrealistic images which motivated the research on non-photorealistic rendering.

Non-photorealistic rendering(NPR) can be characterized by describing the scene such as a hand-crafted look. There are many fields in NPR such as Painterly Rendering[1][4][9][5], Pen-and-Ink Rendering[2][3], Toon Rendering, Technical Sketch[16], Mosaic, and Indian Ink Painting. We present a technique to maintain a frame-to-frame coherence for 3D painterly animations. We should determine the location, orientation, length, and width of the stroke for 3D painterly animation. We use a 3D polygon mesh model, so we located strokes on a model's surface, and oriented a stroke's orientation through the surface's curvature. The stroke's length was determined through the surface's curvature to make a curved line, and the stroke's width was determined enough to cover all the faces. We also provide a frame-to-frame coherence by using a 3D particle system and dynamical stroke managing system.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 317–325, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Related Work

There are two methods for rendering a stroke-based scene. One method is to create a stroke set. This method gathers the strokes into a single texture for 3-dimensions. In 2001, Emil Praun et al. presented a 3D real-time hatching using textures[7]. They made strokes to one texture and performed real-time rendering through hardware-based texture mapping. They generated a TAM(Tornal Art Map) according to the stroke's intensity and density. The TAM was generated to include an adjacent texture, as to maintain a frame-to-frame coherence.

Another method for rendering a stroke-based scene is to make each stroke so that render it respectively on the surface. In 1996, Babara J. Meier presented a method to render a 3D object by generating each stroke[9]. She rendered a haystack by applying a painting brush to a simple cone resting on a cylinder. She made it possible to maintain frame-to-frame coherence by using particles that contain attributes of the stroke, such as location, orientation, size, and color. She tessellated the surface into triangles that approximate the surface. Then, for each triangle, she computed its surface area and randomly distributed particles within it. After the initial particle placement, other attributes are determined from reference pictures that represent orientation, size, and color. The scene was then rendered the scene using brush texture. However, her system was not effective when the camera was zooming in/out. Since when camera was zooming in/out, the size of the brush stroke became too large or too small so that the brush stroke might not be considered hand-crafted brush stroke.

3 Stroke-Based 3D Rendering System

Our system, that renders stroke-based 3D scene, consists of four parts of a 3D model manager, stroke modeling, stroke rendering, and dynamic stroke manager (Fig 1).

In 3D model manager stage, a input 3D model is transformed into a polygon mesh model organized with a vertex list, an edge list, and a per-vertex color part.

In stroke modeling stage, we determine the attributes of each stroke. Our stroke is a painterly brush stroke. So our stroke has width, and its shape is a long and curved line. We determine the location, orientation, length, width and color of the stroke in this stage.

In the stroke rendering stage, we show the stroke in screen depending on the stroke's attributes that was determined during the stroke modeling stage. Each stroke is rendered in shape of a long curved line with a width.

In the dynamic stroke manager stage, we manage strokes dynamically. In this stage, we maintain a frame-to-frame coherence when an object is zooming in/out.

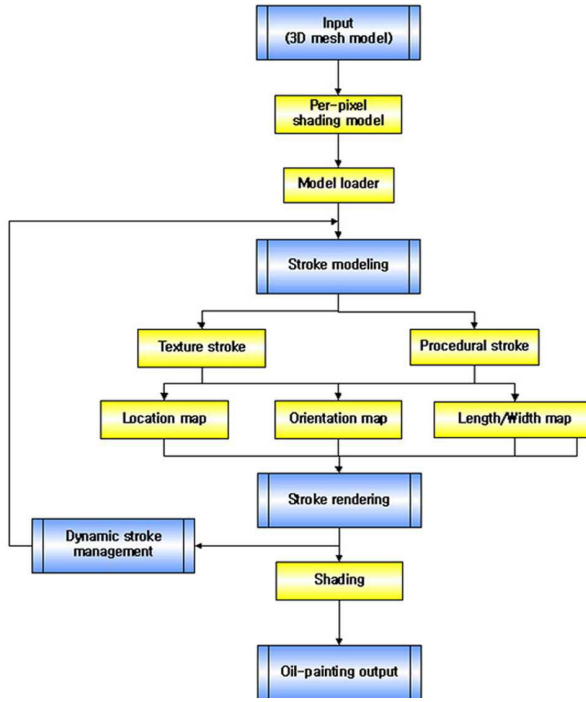


Fig. 1. Stroke-based 3D Rendering System

4 Stroke Modeling

We use a 3D polygon mesh model to place strokes on a 3D polygon mesh model. We should place a stroke's location on each triangle's surface. Then, we extend a stroke along the triangle's surface. We use a 3D particle system for placing the strokes's location. We place the particle on the triangle's surface using the triangle's centroid and median (Fig 2). First, we place a particle on the triangle's centroid. Then, we place six particles from the centroid along the three medians. We place particles through this method until we have enough space on a triangle's surface. Each particle represents a stroke's location and includes stroke's attributes, such as orientation, width, length.

The orientation is determined according to the surface curvature. Ahna Girshick used a principal direction of the surface to represent surface shape more accurately[11]. At any point on a 3D surface, each of the orthogonal first and second principal directions are exist. The first principal direction is the direction of the maximum magnitude of normal curvature, and the second principal direction is orthogonal to the first. The psychology literature suggests that lines in the principal direction of curvature may communicate surface shape better than

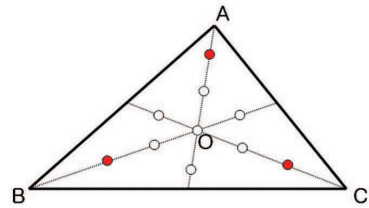


Fig. 2. Placement of particles using triangle's centroid and median

lines in other directions. We use the first principal direction to represent a 3D surface shape more accurately. However, for a polygonal model that we use, the principal direction of the surface within a triangle is all uniform. Therefore, to find the principal direction, we present a different method from the method for a general curved surface. We calculate first principal directions using the normal curvature approximation method[13]. Suppose W is the unknown Weingarten matrix with respect to local coordinates L at vertex p. Suppose there are n vertices adjacent to p and let qi denote the ith adjacent vertex. We denote by yi the unit vector obtained by projecting the vector pqi (expressed in local coordinates L) onto the plane tangent to $N'p$ and normalizing the result. Then the normal curvature in the direction yi is given by equation (1).

$$k_{y_i} = y_i^T W y_i \tag{1}$$

An approximation to this normal curvature is given by:

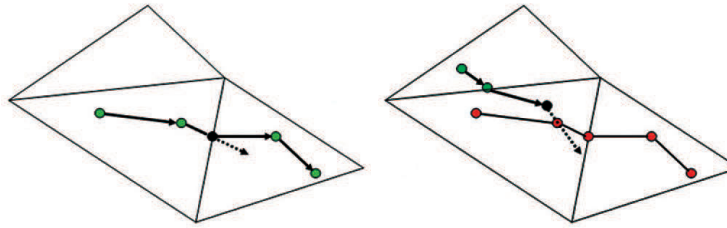
$$k'_{y_i} = 2\frac{N_p \cdot (p - q_i)}{(p - q_i) \cdot (p - q_i)} \tag{2}$$

It is the curvature of the unique osculating circle passing through p and qi with normal $N'p$ at p. We calculate the first principal direction at each vertex of the triangle using this method. The first principal direction at any given point on a triangle's surface is determined by trilinearly interpolating the principal directions of the triangle's vertices.

Each stroke is composed of a set of control points. The criterion for each valid control point is that it lies at a minimum distance threshold from all existing strokes. The direction of the stroke is updated at frequent distance intervals as well as when a stroke crosses a polygon boundary. Strokes are terminated if they approach the minimum distance threshold. This process is shown in Fig. 3. At each control point, we extend the stroke's width and we connect control points at each side for the thick stroke.

5 Stroke Rendering

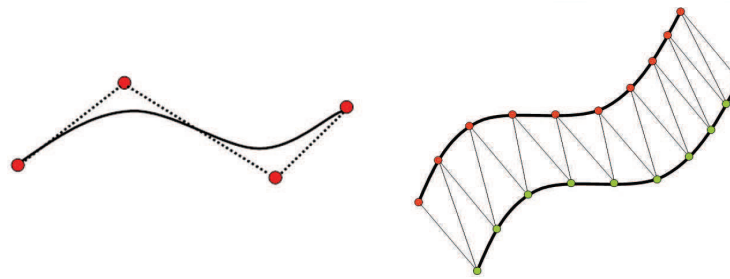
In the stroke rendering stage, we render strokes according to the stroke's location, orientation, length, and width, which are determined in stroke modeling stage.



(a) (b)

Fig. 3. Extension of the stroke

We render the stroke with a Bezier curve for representing a curved stroke[12]. Fig 4(a). shows the curved stroke using a Bezier curve. Four points are control points forming the long stroke and the dotted line represents the long stroke. We estimate the curved stroke using a Bezier curve with four control points. The solid line represents this curve stroke. We extend the stroke for widening the stroke. Therefore, we estimate the curved stroke for each side. We then render the stroke in triangle strip form(Fig 4(b)).



(a) (b)

Fig. 4. (a)Curved stroke generation, (b)Rendering the long, thick, and curved stroke

6 Frame-to-Frame Coherence

We present the method for the frame-to-frame coherence in this section. There are two cases for frame-to-frame coherence in stroke-based 3D rendering. One case occurs when the object is moving or rotating. In this case, strokes should move or rotate with the object. Another case occurs when object is zooming in/out. In this case, strokes are scaled-up or scaled-down when the object is

zooming in/out. This makes the stroke either too large or too small. To solve this problem, we present a dynamic stroke generation technique.

6.1 Particle System

Meier used a particle system to maintain the frame-to-frame coherence[9]. She rendered a haystack by applying a painting brush to a simple cone resting on a cylinder. She tessellated the surface into triangles that approximate the surface. Then, for each triangle, she randomly distributed particles within it. Each particle means the location of the stroke. Since the particle is fixed on the surface, she could maintain the frame-to-frame coherence when the object is moving or rotating. We maintain the frame-to-frame coherence using this particle system. Fig 5. shows the particle placement.

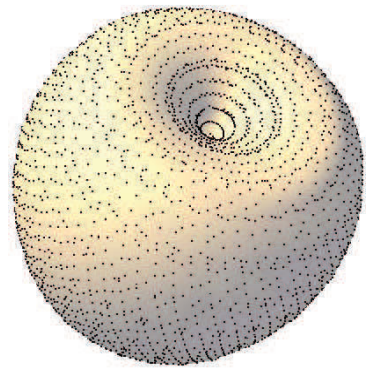


Fig. 5. Distribution of particles

6.2 Dynamic Stroke Generation

The particle system for maintaining the frame-to-frame coherence has a weak point. The problem occur when the object is zooming in/out. Although the number of particles is sufficient enough to render the surface, the brush stroke, represented by particle, appeares too large or too small. This can not be a hand-crafted look. To solve this problem, we present the dynamic stroke generation technique. The key idea of this technique is to preserve the size of the stroke in screen space when the object is zooming in/out. Equation (3) compute the width of the stroke in world space according to the distance between the viewer and the object. By this equation, we preserve the width of the stroke in screen space.

$$w = \lceil \frac{d}{d_0/w_0} \rceil + 1 \tag{3}$$

7 Result

Fig 6. shows the result of our system. We use the apple model consisted of polygons. (a) shows a 3D polygonal model, and (b) shows the per-vertex shading model. (c) shows the particles on the surface. (d) shows the orientation of the particles. Figure (e) is a rendered image by strokes. Fig 7. and Fig 8. shows the result of animated scene. Fig 7. shows the case when the object is rotating, and Fig 8. shows the case when the object is zooming in. While you see the zoomed area by blue box, You can see that strokes maintain coherence in each frame when the object is rotated or zoomed. In both case, we can maintain the frame-to-frame coherence of the stroke. You can see the animation results in the following web cite HTTP://cglab.cse.cau.ac.kr/

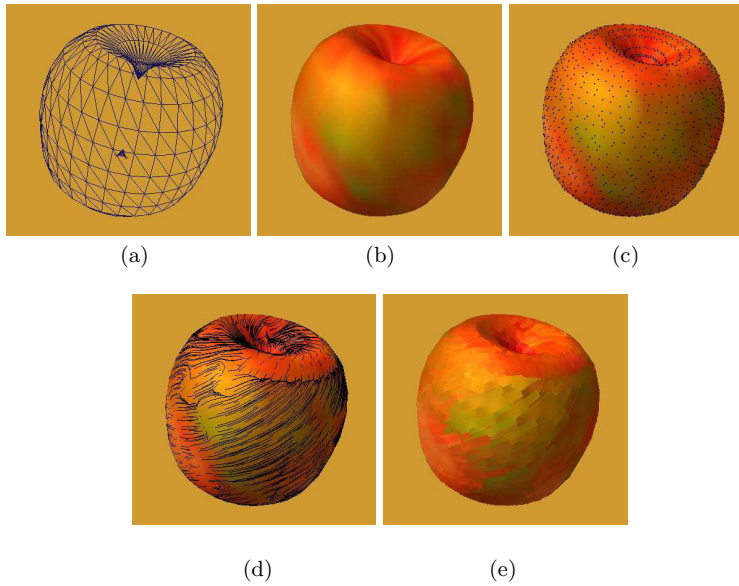


Fig. 6. Result, (a)3D polygonal model, (b)Per-vertex shading, (c)Distribution of particles, (d)Orientation of strokes, (e)Stroke rendering

8 Conclusion & Future Works

It is considered a very important subject for non-photorealistic animations to maintain a frame-to-frame coherence. If this is not maintained, the resulting frames randomly change every frame; or the "shower door" effect occurs in which an animation appears as if it were being viewed through textured glass because brush strokes stick to the viewplane and not to the animating surfaces. We

employed the 3D particle system to solve this problem. We also maintained the frame-to-frame coherece by dynamic stroke while the object is zooming in/out. By doing so, we maintained the frame-to-frame coherence as well as the quality of the result image. We rendered the scene non-photorealistically using brush strokes. However, we should research the stroke itself(i.e. the stroke style of Vincent van Gogh or Impressionists). Furthermore, we can emphasize the shadow or highlight by applying a general oil painting process. The color's attributes, such as thickness, blending, spreading, etc, are also subjects for future study.

Acknowledgement

This paper was supported by the ITRI Strategic Research Fund of Chung-Ang University in 2003.

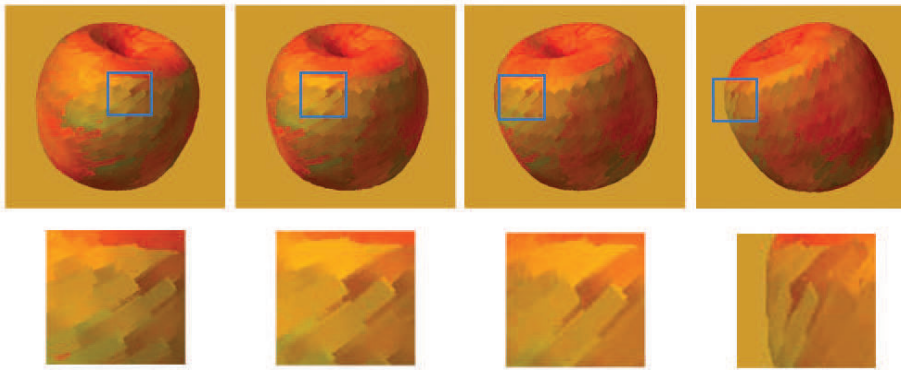


Fig. 7. Rotated Result

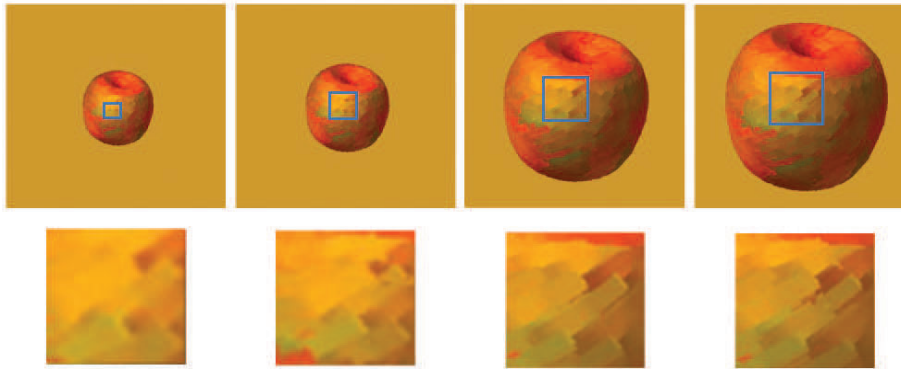


Fig. 8. Zoomed Result

References

[1] P. Haeberli : Paint By Number:Abstract Image Representations. In Siggraph 1990 Conference Proceedings, pages 207–214 (1990)

[2] G. Winkenbach, D. H. Salesin : Computer-Generated Pen-and-Ink Illustration. In Proceedings of Siggraph 1994, pages 91–100 (1994)

[3] M. P. Salisbury, S. E. Anderson, R. Barzel, D. H. Salesin : Interactive Pen-and-Ink Illustration. In Proceedings of Siggraph 1994, pages 101–108 (1994)

[4] Aaron Hertzmann, Denis Zorin : Illustrating smooth surfaces. In Proceedings of Siggraph 2000, pages 517–526 (2000)

[5] Aaron Hertzmann : Paint by relaxation. In Computer Graphics International 2001, pages 47–54, ISBN 0–7695–1007–8 (2001)

[6] Aaron Hertzmann, Ken Perlin : Painterly Rendering for Video and Interaction. In Proceedings of the First Annual Symposium on Non-Photorealistic Animation and Rendering (2000)

[7] E. Praun, H. Hoppe, M. Webb, A. Finkelstein : Real-time hatching. In Proceeding of Siggraph 2001 (2001)

[8] G. Gorla, V. Interrante, G. Sapiro : Texture Synthesis for 3D Shape Representation. IEEE Transactions on Visualization and Computer Graphics (2002)

[9] Babara J. Meier : Painterly Rendering for Animation. In Siggraph 96 Conference Proceedings, pages 477–484 (1996)

[10] Victoria Interrante, Chester Grosch : Visualizing 3D Flow. IEEE Computer Graphics and Applications, 18 (4):49–53,199

[11] Ahna Girshick, Victoria Interrante, Steven Haker, Todd Lemoine : Line direction matters: An argument for the use of principal directions in 3D line drawings. In NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering, pages 43–52, ACM Siggraph / Eurographics (2000)

[12] Aaron Hertzmann : Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In Siggraph 98 Conference Proceedings, pages 453–460 (1998)

[13] J. Goldfcather, V. Interrante : Understanding Errors in Approximating Principal Direction Vectors. ACM Transactions on Graphics (2003)

[14] B. Jobard, W. Lefer : Creating Evenly-Spaced Streamlines of Arbitrary Density. In Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing, pages 45–55 (1997)

[15] Aaron Hertzmann : Fast Paint Texture. In NPAR 2002: Proceedings of the Second Annual Symposium on Non-Photorealistic Animation and Rendering (2002)

[16] Bruce Gooch, Peter-Pike Sloan, Any Gooch, Peter Shirley, Richard Riesenfeld : Interactive Technical Illustration. In Proceedings of ACM Symposium on Interactive 3D Graphics (1999)

A Directional Stroke Generation Algorithm for Painterly Rendering

Jeong Seob Cha, Young Sup Park, and Kyung Hyun Yoon

Department of Image Engineering
Graduate School of Advanced Imaging Science, Multimedia and Film
ChungAng University, Seoul, Korea
{jscha, cookie, khyoon}@cglab.cse.cau.ac.kr
http://cglab.cse.cau.ac.kr

Abstract. This paper presents a brush stroke direction creation method to distill painterly expressions from picture images. In actual paintings, each brush stroke has the same directionality in one segmented area, but directionality changes in the area of image border edges. This study employed the following methods to implement the directionality of the brush stroke area by: (1) detecting edge in input images, (2) region splitting and merging based on quad-tree division method, and (3) composing a direction map to create brush strokes with the same directionality in the segmented area. The results of the implementation enabled the simulation of the directionality of the stroke area shown in actual paintings.

1 Introduction

Numerous studies were conducted to create realistic images in the computer graphic rendering field. Recently, its focus has been shifted to NPR(Non - Photorealistic Rendering). Unlike previous rendering techniques that created realistic rendering(i.e., a Ray tracing, Radiosity, etc), NPR attempts to display artistic images created by human hands using the computer. NPR is divided into the following categories: Painterly rendering [1,2,3,4,8,9] , Watercolor [5], Pen and Ink [7], Toon rendering, etc. Among them, this study limited its focus to painterly rendering that simulates hand painting effects.

As shown in [Fig.1], in actual paintings, brush strokes have the same directionality in one segmented area, but directionality changes in the area of image border edges. To illustrate the directionality of brush strokes by the area, this study detected edge that uses Sobel filter in input images and splitting and merging the area according to the quad-tree division method. We also created a direction map to create brush strokes with the same directionality. In the direction map, we composed the average of image gradient in the direction of each pixel in the divided area and used it to determine the direction when creating brush strokes. We created brush stroke using Bezier curves and determined the starting point of strokes and their numbers according to the size of the segmented area. We also expressed the quality and texture of brush strokes created by the procedures above using the Phong illumination model.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 326–335, 2003.
© Springer-Verlag Berlin Heidelberg 2003



Fig. 1. Field of Spring Wheat at Sunrise, Vincent van Gogh, 1889

2 Related Work

In general, brush strokes are used to simulate painterly rendering. As an early study on painterly rendering, Paul Haeberli suggested a method to determine the position and direction of brush strokes through mouse input in picture images. This was used to simulate abstract effects; however, user input is required [1].
Litwinowicz created brush strokes with lines and textures [4]. The direction of a brush stroke was determined by image gradient and various strokes were created through the addition of random noise values to each stroke property. Moreover, considering image outlines, a stroke clipping method was introduced on the border edge. However, since the strokes used are straight lines, it made the image dull and flat and further failed to provide the effects of hand painting as much as desired.
Hertzmann presented the autonomized method that creates painterly effects on picture images [2]. Furthermore, this method created strokes along the edge with image gradient and spline curves. Brush stokes have different sizes and their layers vary according to the size. Rendering is performed on layers from biggest to smallest to enhance the quality of final images. However, it fails to express the quality and texture of strokes shown in actual paintings. In addition, it cannot create strokes along the edge because the image gradient in the complex image does not show consistent directionality.
To compensate for the weaknesses of the previous studies, this study examined a method to create brush strokes with the directionality by dividing and combining areas and further simulating the effects of hand painting. The previous research determined the direction of brush strokes by using user input or image gradient. However, these are not immune from problems. The former consumes too much of user's time and energy. The latter makes picture images complex, which prevents image gradients from having consistent directionality. That is, it cannot simulate the directionality of brush strokes by the area shown in actual paintings. To eliminate the problems above, this study created a direction map by dividing areas based on quad-tree and created brush strokes according to the direction map. Moreover, this study suggested the quality and texture of brush strokes displaying with the directionality of the area in actual painting, as well as a blending method.

3 System Flow

In general, brush strokes are applied to input images to bring out the same results from picture images as the actual paintings. [Fig.2] represents the system flow used to create and apply brush strokes. The overall procedure is as follows: (1) detect edge by using Gaussian blurring and Sobel filters. The detected edge is used to preserve the outlines of images by clipping strokes from edges when rendering strokes. (2) Create a direction map to determine brush strokes. The direction map is created based on quad-tree division for input images and area combination. (3) Set up Bezier curves and control points which serve as a basis of stroke creation in reference to the created map. The brush strokes used in actual paintings are in general curves; therefore, Bezier curves are used to express brush strokes. (4) Clip the created brush strokes from the edge and blend them in reference to the background color or apply them to images using Phong illumination model.

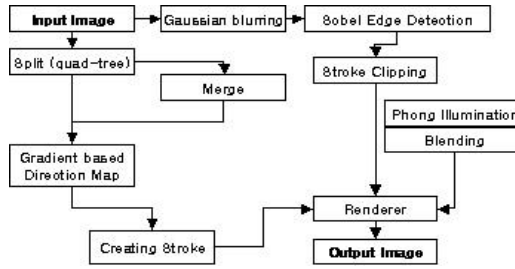


Fig. 2. System Flow

4 Creating the Direction of a Brush Stroke

As shown in [Fig.1], in general, the brush strokes in actual painting have the same directionality by the area, but the stroke direction is different along the border edge. To implement the directionality by the area, the region-oriented segmentation of input image is required. This study created a direction map based on region-oriented segmentation to control the direction of brush stokes and maintained the brush strokes to the same direction by the area using the direction map.

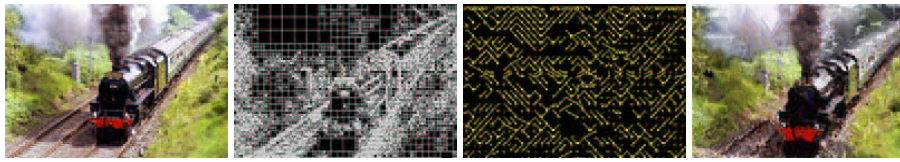
4.1 Direction Map Based on Region Splitting Using Quad-Tree

In order to create a direction map, which is a reference image used to determine the direction of brush strokes, we segment an input image using region splitting method. As a quad-tree division method, the area is divided by the following

method: combining parts with the values that do not exceed threshold values in the gray level image in one area. As shown in [Fig.3](b), after the quad-tree division is completed, the direction $D(x, y)$ in one area is set to a mean direction of image gradient $G(x, y)$ as in equation (1) and the direction map was composed as shown in [Fig.3](c).

$$D(x,y) = [\sum_{startpoint}^{end_point} G(x,y)]/(area_width \times area_height) \quad (1)$$

In case of the direction map that employed a quad-tree division method, when the differences in gray level values between pixels are huge, one pixel composes one segmented area. In general, picture images have a tendency to contain many small areas in the region-oriented segmentation, which prevents the composition of a direction map with the consistent direction.
However, the presence of many large segmented areas compose a direction map that flows to one direction in general. To prevent this from happening, the minimum size of the quad-tree is restricted to 4 × 4 pixel and the maximum to 32 × 32 pixel. [Fig.3](d) shows the result from strokes creation and rendering after composing a direction map that employed the region-oriented segmentation method.
When applying the direction map above, if it is segmented into two areas or more that it looks like a one area, strokes with different direction is created. This is because the segmentation results vary depending on threshold values set up for quad-tree division. However, setting up an exact threshold value according to features of input images is a difficult task.



(a) (b) (c) (d)

Fig. 3. Direction map based on region splitting using quad-tree and result image of applying the stroke, (a) Source image, (b) Quad-tree, (c) Direction Map, (d) Result image

4.2 Direction Map Based on Region Splitting and Merging

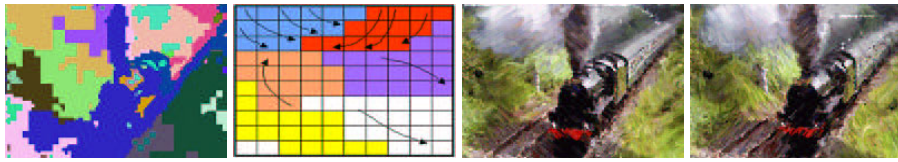
In case of the direction map using region splitting method, images cannot be divided exactly along the border of edge, so it creates strokes with different directions in the same area. From this point of view, we use region splitting and merging method as shown in [Fig.4](a) and create the direction map that shows

a consistent direction by the merged area. The direction of the first starting pixel in the segmented area is set to image gradient direction as shown in equation (2). For the next pixel, as shown in equation(3), Perlin's Noise [10] value is added as much as α to the direction X and β to the direction Y so that it would have a consistent direction in one segmented area. (Refer to [Fig.4](b)). [Fig.4](c) shows the results from the use of the direction map where the noise value is not applied to the direction X and Y in equation (3).

$$D_{0,0}(x,y) = \frac{G(x,y)}{\sqrt{gx^2 + gy^2}} \tag{2}$$

$$D_{i+1,j+1}(x,y) = \frac{D_{i,j}(x+\alpha, y+\beta)}{\sqrt{(x+\alpha)^2 + (y+\beta)^2}} \qquad (|\alpha| < 1, |\beta| < 1) \tag{3}$$

These result shows the directionality by the area, but all the strokes are created in straight lines, which does not contain the feelings of hand paintings. In contrast, [Fig.4](d), which resulted from the direction map - the addition of noise like equation (3). This case shows both the curve stroke creation and the directionality by the area.



(a) (b) (c) (d)

Fig. 4. Direction map based on region Splitting & Merging and result image of applying the stroke, (a) Region Splitting&Merging, (b) Direction Map, (c) Result(Perlin's noise is not applied), (d) Result(Perlin's noise is applied)

5 Brush Stroke

In general, when drawing a picture, one completes a rough sketch and then adds a detailed touch. For a part that takes up a large area such as background, a long stroke is used and a short and thick stroke is used to add details to a complex area. In addition, brush strokes with different properties can be applied to distinguish each object in images.

This study defines the properties of brush strokes by the following: a starting point, direction, width, length, shape and color. And it creates strokes by determining the values for each property enumerated above.

5.1 Creating a Brush Stroke

For the starting point for a brush stroke, one segmented area is divided by 3 × 3 pixel grid in the image divided by quad-tree and the center point of the grid is selected. The brush stroke shape is in general represented in curve, and thus, we use Bezier curve. The control point of Bezier curve is selected in reference to a direction map and 4 control points are drawn following the direction map from the stroke starting point. First, the stroke starting point is set to the first control point $C_0(x, y)$ of the Bezier curve. The second control point is set to have an interval of the length as previously set up according to the direction vector $D(x, y)$ of the direction map as shown in equation (4).

$$C_1(x, y) = C_0(x, y) + D_0(x, y) \times length \tag{4}$$

The rest of the control points are calculated by the same method. The width of brush strokes is obtained by expanding it to the given width upward and downward or left and right based on the Bezier curve that follows the 4 control points. In the Bezier curve expanded like this, control points are selected at the interval of 3 pixels and a polygon will be created. Furthermore, the inside of the polygon is filled with a color that corresponds to the stroke starting point in the input image by using a scan line fill algorithm.

5.2 Rendering the Brush Stroke

When creating and rendering brush strokes to simulate painterly rendering, one needs to consider the following factors: a rough sketch on canvas to fill a hole, stroke clipping to maintain image outlines and determination of stroke properties. In this study, for the rough sketch, input image blurred by a Gaussian filter is used. We also use the stroke clipping method suggested by Litwinowicz[4]. In painterly rendering, the simulation results differ depending on the properties of the brush strokes. Among them, the one that affects the results most is the opacity level, the quality, texture and its length. The opacity level is set by the method to interpolate strokes, lines and shapes as shown in equation (5).

$$PaintingColor = C_{StrokeColor} \times \alpha + B_{BgColor} \times (1 - \alpha), (0 \leq \alpha \leq 1) \tag{5}$$

[Fig.5](a) shows the rendering results by adjusting the opacity level of strokes. When the opacity level is set to low, the directionality of the area, which is suggested by this study, did not exist. However, as demonstrated in the rendering result, it brings out watercolor effects.

In actual paintings, one can notice the quality and texture in the over-painted area along with the directionality of brush strokes. For brush strokes to possess the quality and texture, hight values needs to be applied to each stroke or a illumination model needs to be used for the entire image. In this study, we use Phong illumination model to bring out over-painted effects and to express the quality and texture of strokes. Oil colors and water colors do not have specular reflection when dried. For this reason, Phong illumination model used for this study take into consideration ambient light and diffuse reflection except the



(a) (b)

Fig. 5. Rendering of brush strokes, (a) Opacity = 50%, (b) Result using Phong illumination model

specular reflection. This study suggests the following quality and texture creation method: first, obtain a intensity map, which is an edge image rendered without applying a illumination model and then obtain a normal vector and a lighting vector for the image pixel to apply the Phong illumination model. Here, the specular reflection is not simulated, and thus, a viewing vector is not needed. The normal vector is created by using Perlin's noise and an intensity map, lighting is fixed to a certain level and the Phong illumination model is applied to [10]. [Fig.5](b) shows the rendering result after calculating the intensity of each pixel using equation (6). It also shows the simulation of the quality and texture shown in actual paintings.

$$Intensity = Ka + \overrightarrow{Light} \cdot \overrightarrow{normal} \tag{6}$$

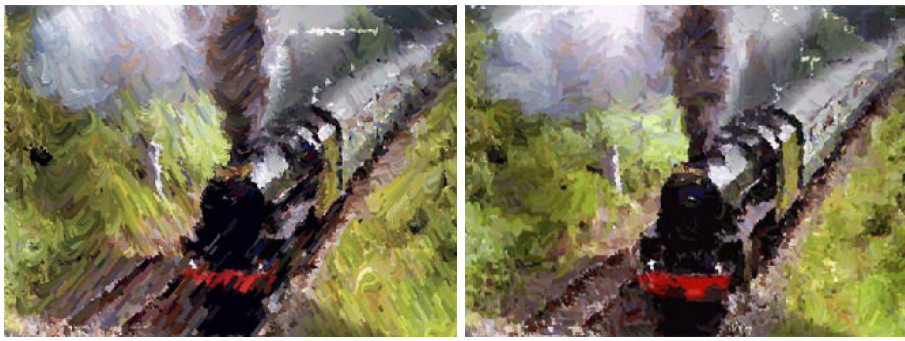
$$\overrightarrow{Light} = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}), \quad \overrightarrow{normal} = (ix, iy, Perlin'snoise)$$

$$(ix, iy) = IntensityMap(x, y)$$

6 Comparision

In this section, we compared the result of Hertzmann's and our method. [Fig.6](b) represents the result implemented by the method which creates the brush strokes to a normal direction in image gradient [2] as suggested by Hertzmann. This method creates strokes in the same direction as image edges. However, in reality, picture images are complex, which prevents them from having a consistent direction. For this reason, strokes with consistent directionalities cannot be created.

The method employed by this study suggested that stroke directions and final images differ according to a direction map creation method for a input images. [Fig.3](d), which resulted from the use of the direction map created by region splitting method, shows the strokes with different directionalities even in the



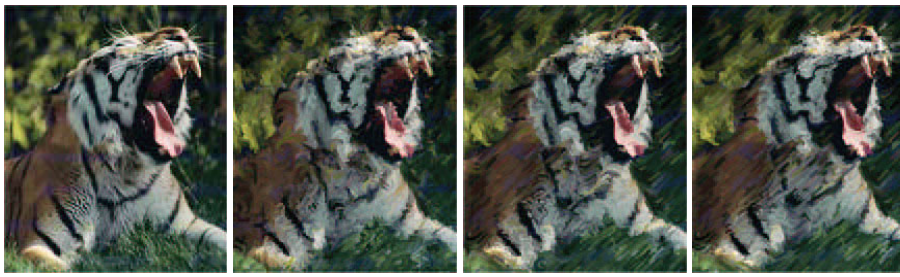
(a) (b)

Fig. 6. Our work versus previous work, (a) Using direction map based on region Splitting & Merging, (b) Result using Hertzmann's method[2].

same area. This method becomes strikingly similar to the [Fig.6](b), which is the result of the method suggested by Hertzmann as input images become complicated and accordingly, the number of areas with the minimum size increases in the segmented area.

To compensate for the method above, this study suggested to create a direction map by merging areas and presented the result shown in [Fig.6](a). Creating a direction map by merging areas is based on quad-tree division method, and thus, various direction maps can be created according to the quad-tree level.

At this point,the quad-tree level is determined by threshold values. [Fig.7] shows the result that change as threshold values change. When the threshold value is low, it divide images into more segmented areas, making stroke directions more complicated. However, when the threshold value is high, the number of segmented areas is small, displaying stroke directionality by the area. When the threshold value is too high, as illustrated in [Fig.7](d), the stroke is created in one direction. In this study, threshold values for region-oriented segmentation ranging from 40 to 70 is found to be appropriate to create directionalities of strokes by the area. [Fig.8] shows the results when adjusting the quad-tree level at the 60 threshold value. When the minimum size is the same as (b) or (c), changes in the direction map hardly occurred. This indicates that changes in the maximum size do not have much influence on determining quad-tree levels since the quad-tree level is decided by the complexity of the image and threshold values. (d) shows that stroke is created in one direction as the quad-tree level increased when adjusting the minimum size. This is because the number of areas to be divided decreased as the minimum size of the quad-tree increased just like threshold values increased. The analysis revealed 4×4 pixel was found to be suitable for the minimum size of the quad-tree level to express the directionality by the area while maintaining strokes to a certain direction and 32×32 pixel was found to be suitable for the maximum size.



(a) (b) (c) (d)

Fig. 7. The result that change as threshold values change, (a) Source image, (b) Threshold value = 50 , (c) Threshold value = 70 (d) Threshold value = 90



(a) (b) (c) (d)

Fig. 8. The results when adjusting the quad-tree level at the 60 threshold value, (a) Source image, (b) min=4 × 4, max=8 × 8, (c) min=4 × 4, max=32 × 32, (d) min=32 × 32, max=32 × 32



(a) (b) (c)

Fig. 9. Opacity and quality and texture of brush stroke, (a) Source image, (b) Water color, (c) Expression of a quality and texture using Phong illumination model

7 Conclusion and Future Work

This study presented a brush stroke direction creation method to achieve painterly expressions from picture images. To create a direction map used to determine brush stroke directions, area division and combination for input images were used and by doing so, this study implemented the directionality of strokes by the area as shown in actual paintings.
The algorithm presented in this paper is dependent on features of input images. If an input image is complex, it could not get a good result. So, it suggested appropriate threshold values for the direction map by drawing various results for the identical input images that change according to threshold values. However, the suggested threshold values are not appropriate for all the input images, and thus, one needs to find the threshold values for the quad-tree division according to the property of the image by conducting in-depth researches.
Furthermore, brush strokes shown in actual paintings possess characters such as quality and texture, grazing effects, color mixture, etc besides the directionality by the area. To illustrate these effects, analysis of the hydromechanical features of actual paint and their simulation are needed.

Acknowledgement

This paper was partially supported by the National Research Laboratory program of the Korean Ministry of Science and Technology (No. 2000-N-NL-01-C-285).

References

1. Paul Haeberli. Paint by Numbers : Abstract image representations, SIGGRAPH 90 Proceeding, pp 207-214, 1990
2. Aaron Hertzmann, Painterly rendering with curved brush strokes of multiple sizes, SIGGRAPH 98 Proceeding,pp 453-460, 1998
3. Aaron Hertzmann, Fast Paint Texture, NPAR 2002 Proceeding, pp 91-96, 2002.
4. Peter Litwinowicz, Processing Images and Video for an impressionist effect, SIGGRAPH 97 Proceeding, pp 407-414, 1997
5. Cassidy J. Curtis, Sean E. Anderson, Joshua E. Sems, Kurt W. Fleischer, David H. Salesin, Computer-Generated Watercolor, SIGGRAPH 97 Proceeding, pp 421-430, 1997
6. Thomas Strothotte, Stefan Schlechtweg, Non-Photorealistic Computer Graphics Modeling, Rendering, and Animation, Morgan Kaufmann Publishers,2002
7. Mike Salisbury, Scan Ahdeison, Ronen Barzel, Avid Salesin, Interactive Pen-and-Ink Illustration, SIGGRAPH 94 Proceeding, pp 101-108, 1994
8. Bill Baxter, Vincent Scheib, Ming C. Lin, Dinesh Manocha, DAB: Interactive haptic painting with 3D Virtual Brushes, SIGGRAPH 01 Proceeding, pp 461-468, 2001
9. Adobe System, Adobe Photoshop 5.0
10. Ken Perlin's Web Page, http://mrl.nyu.edu/ perlin/

A Simple Method for Ray Tracing Diffraction

Emmanuel Agu[1] and Francis S. Hill Jr.[2]

[1] Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609, USA
emmanuel@cs.wpi.edu

[2] Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01002, USA
hill@ecs.umass.edu

Abstract. Diffraction and interference are optical phenomena which split light into its component wavelengths, hence producing a full spectrum of iridescent colors. This paper develops computer graphics models for iridescent colors produced by diffractive media. Diffraction gratings, certain animal skins and the crystal structure of some precious stones are known to produce diffraction. Several techniques can be employed to derive solutions to the diffraction problem including: (1)Electromagnetic boundary value methods (2)Applying the Huygens-Fresnel principle (3)Applying the Kirchoff-Fresnel theorem (4)Fourier optics. Previous work in developing diffraction models for computer graphics has used boundary value methods and Fourier optics but no models using Huygens-Fresnel principle have been published. This paper derives a set of diffraction solutions based on the Huygens-Fresnel principle, which are then used to extend well-known illumination models. We then use our new models to render images in a ray tracer.

1 Introduction

Humans live in a world filled with beautiful colors. Iridescent colors refer to the different colors which some surfaces radiate at different light source or viewer angles. New colors which were neither visible in the incident light nor the object being observed appear to have been created. These surfaces are sometimes said to "shimmer" as they are rotated. Common sources of iridescent colors include rainbows, shiny CD-ROM surfaces, opals, hummingbird wings, some snake skins, oil slicks and soap bubbles. Four mechanisms are known to produce iridescent color, namely, dispersive refraction, scattering, interference and diffraction.

Diffraction occurs when light encounters some obstacle or aperture of a dimension comparable with its wavelength. For example, a diffraction grating has a series of finely ruled parallel grooves. Different wavelengths are diffracted at different angles; hence different colors are produced at different angles, leading to the phenomenon of iridescence. Common sources of iridescent colors include diffraction gratings, opals and some liquid crystals, hummingbird winds and some snakeskins. Diffraction, also referred to as a wavefront splitting phenomenon is distinguished from interference, an amplitude-splitting phenomenon [Gon94].

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 336–345, 2003.
© Springer-Verlag Berlin Heidelberg 2003

A common problem in optics is that of determining the outgoing light intensity, wavelength and color, given that light is incident on the diffraction surface at a certain angle and intensity and is comprised of specified wavelengths. In particular, a closed-form expression relating incoming and outgoing light permits diffraction surfaces to be elegantly modelled in computer graphics. Several techniques have been employed in the optics literature to derive solutions to this problem [Str67] including (1) Solutions derived by applying electromagnetic boundary value methods (2) Solutions based on Huygens-Fresnel principle (3) Kirchoff-Fresnel based solutions (4) Solutions using Fourier optics.

Two main bodies of work have been identified which attempt to model diffraction in computer-generated imagery. Thorman [Tho96] first developed a simple computer graphics model for diffraction by using the grating equation derived by applying electromagnetic boundary value methods. Stroke [Str67] derives the geometrical conditions for iridescence. Thorman [Tho96] then uses Stroke's results and addresses the specific issue of modelling iridescent colors produced by diffraction in computer graphics. Thorman's work focuses on determining directions in which the grating equation would produce peak responses, and hence render images accordingly. Specifically, the grating equation used by Thorman is not a continuous function but only gives the directions for perfect constructive interference. No information is given on the behavior of the reflected light at angles which are away from those of perfect constructive interference. Also, Thorman erroneously assumes that all peak intensity values are equal. A continuous function needs to be derived which gives the behavior of light in all outgoing directions.

Stam [Sta99] has published work using Fourier analysis. Fourier optics solutions are believed to give the most accurate but also most complex solutions. However, detailed information about both the scene configuration and diffraction surface profile are vital before Fourier solutions can be evaluated and in fact, each solution is valid only for one configuration. In order to arrive at a Fourier optics solution, Stam makes assumptions about the scene configuration, as well as the profile and distribution of the grating surface. The fact that Fourier solutions are configuration-dependent increases their complexity.

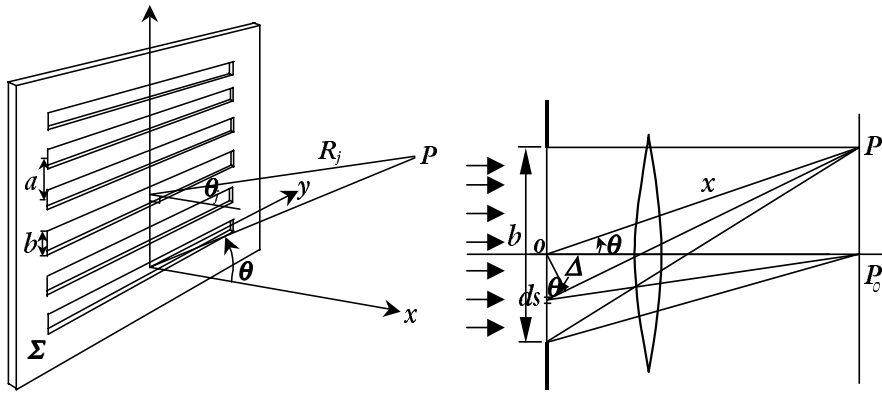
In what follows, we will apply the Huygens-Fresnel principle to derive a continuous function that defines the behavior of light in all outgoing directions and then include it in a complete illumination model, showing clearly how to use the model to render pictures of surfaces with diffraction.

2 Our Optics Model

The Huygens-Fresnel principle states that every unobstructed point of a wavefront, at a given instant in time, serves as a source of spherical secondary wavelets (with the same frequency as that of the primary wave). The amplitude of the optical field at any point beyond is the superposition of all these wavelets (considering their amplitudes and relative phases). This principle arises because all vibrating particles exert a force on their neighbors and thus act as point

sources. It explains why when a wave passes through an aperture or obstruction, it always spreads to some extents to regions that were not directly exposed to the oncoming waves and can be used to derive useful approximate solutions to the diffraction problem.

In optics, distinction is also made between *near-field* or *Fresnel* and *far-field* or *Fraunhofer* diffraction. In Fresnel diffraction, the point of observation is so close to the aperture that the image formed bears a close resemblance to the aperture, the emergent waves are spherical and intensities received at a given point vary as one travels along the aperture width. For Fraunhofer diffraction on the other hand, the point of observation is so far that the image formed bears almost no resemblance to the actual aperture, the emergent waveforms can be approximated as planar waves with uniform intensities from any point on the aperture width. We deal only with Fraunhofer diffraction in this paper since it encompasses almost all configurations of practical interest in computer graphics.



(a) Diffraction System with N Slits (b) Diagram of a single slit

Fig. 1.

We shall now apply the Huygens-Fresnel principle to derive an expression for the irradiance from an N-slit diffraction grating where N is sufficiently large and the point of observation is sufficiently far from the grating surface. Note that the following derivations apply to gratings with slits, transmissive and reflection gratings. Applying the Huygens-Fresnel principle, each slit or grating edge of a width much less than λ, now acts as a secondary source. The number of slits, N (typically in the thousands per inch of diffraction grating) is tremendously large and their separation is small. Figure 1(a) is an example of such a grating. Only a few slits of the grating are shown in the figure for ease of illustration. We wish to derive a closed form expression for the net contributions of these N slots of a diffraction grating at an arbitrary location in space P. The final closed form

derivation will be approached in a simple two-stage process. First we derive the effect of one of these N slits at the point P, following which we shall consider the net effect of vector addition of several of these slits at the point P.

Figure 1(b) shows a single slit of the diffraction grating. ds is the elemental width of the wave front in the plane of the slit, at a distance s from the center O, which we shall refer to as the origin. The wavelet emitted by the element ds, observed at the point P, will be proportional to the length of ds and inversely proportional to the distance x [Hec87], [Jen57], [Lon67]. The general equation for a spherical wave can be written as:

$$y = \frac{a}{r}\sin(wt - kr) \tag{1}$$

where a is the amplitude at a unit distance from the source and r is the distance of the observation point from the source. Hence, from Figure 1(b), the infinitesimal displacement produced at the point P by the infinitesimal element ds can be written as

$$dy_0 = \frac{ads}{x}\sin(wt - kx) \tag{2}$$

The displacement varies in phase as the position of ds changes by a factor, due to the different path lengths to P_0, which can be expressed as $D = s\sin q$. So, at a given point s below the origin, the contribution will be

$$dy_s = \frac{ads}{x}\sin[wt - kx - ks.\sin(\theta)] \tag{3}$$

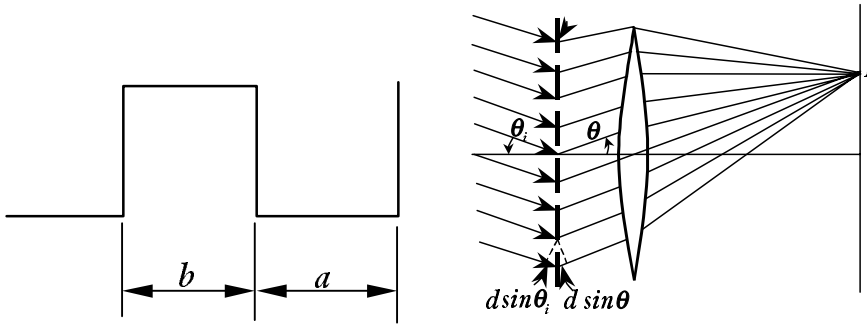
Integrating equation 3 from one edge of the slit to the other, we get

$$y = \frac{ab}{x}\frac{\sin\beta}{\beta}\sin(wt - kx) \tag{4}$$

Hence, the resultant vibration is a simple harmonic one, the amplitude of which varies with position P. Thus, the intensity at the screen due to one slit is then

$$I \approx A_0^2\left(\frac{sin\beta}{\beta}\right)^2 \tag{5}$$

Now, considering a diffraction grating system of N slits, with apertures of width a, and width b of the opaque portion separating apertures (See figure 2(a)). Next, we shall determine the vector sum of several of these single slits at the arbitrary observation point P. The incident light is still at a normal angle of incidence and the phase difference, d, between disturbances from corresponding strips of adjacent apertures is again $\delta = \left(\frac{2\pi}{\lambda}\right)d\sin(\theta)$ where $d = a + b$ specifies the distance between similar points in adjacent apertures as shown in figure 2(a). Expressed as a complex quantity, the phase difference between disturbances from corresponding strips of adjacent apertures is $e^{-i\delta}$. Hence, adding the net contributions of multiple slits of amplitude $\left(\frac{\sin\beta}{\beta}\right)^2$ as determined in the derivation



(a) Diagram showing a and b in adjacent slits

(b) Diagram showing oblique incidence

Fig. 2.

for a single slit and observed at a point P in space, the complex amplitude of the resultant is given by

$$z = \left(\frac{\sin\beta}{\beta}\right)\left[1 + e^{-i\delta} + e^{-2i\delta} .... + e^{-(N-1)i\delta}\right] \tag{6}$$

which gives

$$z = \left(\frac{\sin\beta}{\beta}\right)^2 \frac{\sin^2 N\alpha}{\sin^2\alpha} \text{ where } \alpha = \frac{\delta}{2} = \frac{\pi d \sin\theta}{\lambda} = \frac{\pi(a+b)\sin\theta}{\lambda} \tag{7}$$

Normalizing our result, we get

$$I = I_0 \frac{1}{N^2}\left(\frac{\sin\beta}{\beta}\right)^2 \left(\frac{\sin^2 N\alpha}{\sin^2\alpha}\right) \tag{8}$$

here I_0 is the intensity of the incoming light ray in the $q = 0$ direction. Equation 8 is our final expression for irradiance from an arbitrary N-slit diffraction grating. Finally, we modify our results to take into account oblique incoming and outgoing angles. Consider the following figure 2(b) above. In the case of oblique angles, the general expression for irradiance stays the same. However, the phase difference between disturbances (contributions) from successive slots is given by

$$\delta = \left(\frac{2\pi}{\lambda}(a+b)(sin\theta - \sin\theta_i)\right) \tag{9}$$

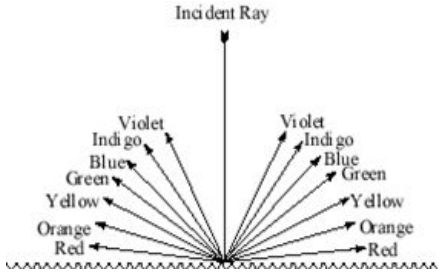
and

$$\beta = \pi b \frac{(\sin\theta - sin\theta_i)}{\lambda} = \qquad \text{and} \qquad \alpha = \pi(a+b)\frac{(\sin\theta - \sin\theta_i)}{\lambda} \tag{10}$$

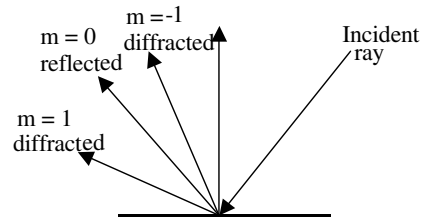
Since $\alpha = \frac{\pi d}{\lambda}(\sin\theta - \sin\theta_i)$

$$d(\sin\theta - sin\theta_i) = m\lambda \tag{11}$$

where $m = 0, \pm 1, \pm 2...$ Equation 11 is widely known as the *grating equation* and gives the locations of maxima.



(a) White light incident perpendicular to the surfaces of a grating and the first order diffraction spectrum on each side



(b) View in the diffraction plane of a monochromatic incident ray and the diffracted rays of first order on each side

Fig. 3.

We can see from equation 11 that different wavelengths (and hence different colors) will peak at different angles with different $\sin q - \sin q_i$ as shown in figure 3(a). Likewise, we can see from figure 3(b) that according to equation 11, different modes peak at different angles.

3 Our Illumination Model

In this section, we shall outline our new illumination models which are based on the Huygens-Fresnel principle, include diffraction and can render iridescent surfaces. We can express our diffraction illumination model as

$$I = Ambient + Diffuse + Diffraction \tag{12}$$

We introduce a new diffraction component in equation 12 to account for both the directional specular and diffraction effects. The ambient and diffuse components in equation 12 are the same as those used in the Phong model [Pho75].The diffraction component in equation 12, $I(\theta)$ is expressed as (see equation 8)

$$I(\theta) = \sum_{\lambda} I_{0,\lambda} \frac{1}{N^2} \left(\frac{\sin\beta}{\beta}\right)^2 \left(\frac{\sin N^2\alpha}{\sin^2\alpha}\right) \tag{13}$$

where equation 13 is a summation of equation 11 over a discrete set of wavelengths and α and β equals the values expressed in equations 10 and $I_{o,\lambda}$ is the light intensity at a summation wavelength, λ. The summation above is over a chosen set of discrete λ, $[\lambda]$ in the visible range. We recall that the visible spectrum has wavelengths in the 380nm to 700nm range. In our illumination model of equation 12, we have replaced the specular term that was used previously in the Phong model by a more general diffraction component that also incorporates the Phong (or Cook-Torrance) specular component. In rendering the model, positions of peak wavelengths for each position are pre-calculated and care taken to include the peak intensity in the rendered image.

4 Rendering Our Models

In this section, we discuss how to use our illumination model to render pictures with diffractive surfaces. A ray tracer is used as our rendering system. However, first, we shall outline issues which need to be taken into consideration before these models can be used.

Thus far, our expressions for our illumination model have not reflected the fact that the grating or diffractive surface may be transformed into new arbitrary 3D positions of the user's choice. A convenient way of incorporating transformations and including the 3D case, is by expressing our illumination model using the half vector. It is interesting to note that in the case where the diffraction grating is coincident with the z-axis, the term $\sin q + \sin q_i$ in our diffraction expression is equal to the z-component of the normalized halfway vector, H. Hence, we can simplify equation 13 by replacing $\sin q - \sin q_i$ with $H.z$. We can thus re-write equations 9 and 10 as

$$\delta = \left(\frac{2\pi}{\lambda}\right) d \times H.z = \left(\frac{2\pi}{\lambda}\right) (a+b) \times H.z \tag{14}$$

and β and α become

$$\beta = \pi b \frac{H.z}{\lambda} = \frac{kb}{2} \times H.z \qquad \text{and} \qquad \alpha = \pi d \frac{H.z}{\lambda} = \frac{k(a+b)}{2} \times H.z \tag{15}$$

This single substitution is very powerful and greatly simplifies our expression and eases manipulation within a rendering system. It is also possible to create patterns using by introducing an arbitrary twist in the grating direction or by creating checkerboard patterns. Color for display in computer graphics is usually specified by relative amounts of a set of primary colors (e.g. Red, Green and Blue or their RGB values) which they contain. Since, the models for iridescence which were described in the preceding sections as well as the accompanying trigonometry, were specified on a wavelength basis, conversion from wavelength to RGB becomes necessary. Furthermore, following conversion, some colors which are readily specified in wavelengths may fall outside the gamut of RGB tuples which the CRT can display. In such a case, these colors need to be systematically

converted or transformed to triples which can be displayed by the CRT. The underlying process of truncation or transformation is known as color clipping. In our ray tracer, we have investigated and implemented three alternative schemes for color clipping. These are clamping, intensity scaling and constant intensity scaling. Hall [3] has a complete discussion of these three scaling techniques.

In a ray tracer, the illumination models are evaluated per pixel and it is sufficient to describe what steps are taken to render a pixel. For each pixel that hits a diffraction surface, the following steps are taken:

1. Determine hit point: determine the first object the eye sees through this pixel while looking into the scene to the ray traced.
2. Compute ambient and diffuse components
3. Build light and eye vectors and compute normalized Half Vector, H
4. Transform grating direction vector, normal vector, twist grating vector
5. Replace $(\sin q + \sin q_i)$ with $H.z$ (half vector component in the grating direction) in equation 13
6. Search for all visible modes and corresponding wavelengths that peak at this angle. If no modes or wavelengths are visible, return diffuse + ambient color. If mode zero is visible, return specular component.
7. Evaluate equation 13 at peak wavelengths.
8. Convert peak wavelengths to RGB colors as described in section 4 and evaluate iridescent color.

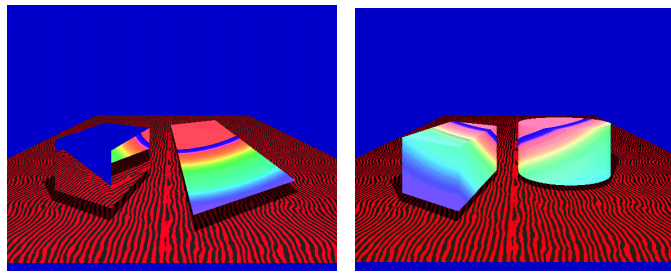
While steps 1-3 above are the same as previous ray tracers, steps 4-8 are related to our diffraction models.

5 Results

Figures 4(a) through 6 are all images incorporating our models and show some iridescent colors. Diffraction gratings with and without an arbitrary twist in the grating direction, have been demonstrated. We have also demonstrated variants of the checkerboard pattern alternating with a diffuse surface, as well as alternating the twist angles. The patterns created were extremely colorful and resembled real life iridescent diffraction patterns. Additionally, we have shown other geometries, such as the cube and cylinder that have been made from these diffractive gratings or patterns. Finally, we have also rendered a CD-ROM surface exhibiting iridescence. Simple animations were also produced to illustrate the color variance as viewer and surface orientations were altered.

6 Conclusion

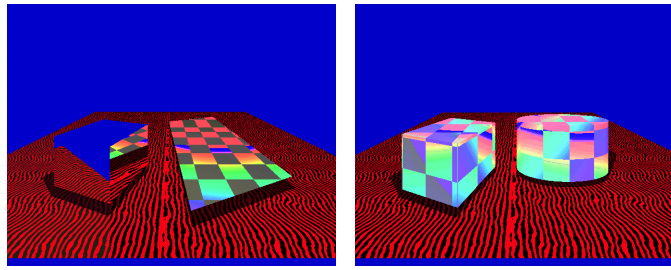
In this paper, we have defined the concept of angle-dependent iridescent coloration of certain materials, due to the optical phenomenon of diffraction. Two earlier attempts by Thorman and Stam, to develop diffraction illumination models were reviewed and their shortcomings clearly stated. We have developed



(a) Diffraction grating and reflective cube on wood textured surface

(b) Diffractive cube and cylinder on wood textured surface

Fig. 4.



(a) Checkerboard diffraction grating on wood textured surface

(b) Checkerboard diffractive cube and cylinder with alternated groove twist angle

Fig. 5.

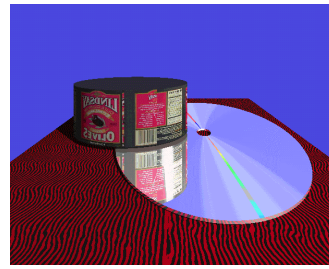


Fig. 6. CD-ROM showing iridescence, and image-mapped can on wood textured surface

diffraction shading models for computer graphics in two distinct phases; first, we developed an optics model, which describes the interaction of incident light with our diffractive surface. Next, we included our optics model in our complete illumination model.

In developing our optics model, we have applied the Huygens-Fresnel principle. In our derivations, we have made assumptions which make sense for computer graphics applications. These include assumptions that incident light is non-polarized, that the grating is several wavelengths away from the point of observation such that emergent waves can be approximated as plane waves (Fraunhofer diffraction).

We have incorporated our optics model into a complete illumination model for computer graphics by adding diffuse and ambient terms, similar to those used in the Phong and Cook-Torrance illumination models. We have rendered these models using a ray tracer and practical issues encountered discussed. Photorealistic scenes with diffractive surfaces, including diffraction gratings, checkerboard patterns and CD-ROMs, have been produced. Possible areas for future research include extending our work to include:

1. Efficient, radiosity-based solution which effectively tracks and renders surface inter-reflections for added realism.
2. Different light sources with different geometries and constituent wavelengths.
3. Three-dimensional gratings such as crystals and gemstones.
4. Diffraction in animals such as snakeskins and butterflies which exhibit iridescence.

References

[Agu02] Agu, E.,Diffraction Shading Models in Computer Graphics, (PhD Dissertation, U. of Massachusetts-Amherst, September 2001).

[Gon94] J. S. Gondek, G. W. Meyer and J. G. Newman, Wavelength Dependent Reflectance Functions, Proc. SIGGRAPH 1994, pp 213-220.

[Hal89] R. Hall, Illumination and Color in Computer Generated Imagery, (Springer-Verlag, 1989).

[Hec87] E. Hecht, Optics, Second Edition, (Addison-Wesley Publishing Co., 1987).

[Jen57] F. A. Jenkins and H. E. White, Fundamentals of Optics, Third Edition, (McGraw-Hill, 1957).

[Lon67] R. S. Longhurst, Geometrical and Physical Optics, Second Edition, John Wiley and Sons, 196.

[Pho75] B. Phong, Illumination for Computer-Generated Pictures, Comm. of ACM, 18(6), pp 311-17, 1975.

[Sta99] J. Stam, Diffraction Shaders Proc. ACM Computer Graphics (SIGGRAPH '99), pp 101-110.

[Str67] G. W. Stroke, Diffraction Gratings pp 426-754, Handbuck Der Physik (Encyclopaedia of Physics), (Springer-Verlag, 1967).

[Tho96] S. Thorman, Diffraction based models for iridescent colors in computer generated imagery (PhD Dissertation, U. of Massachusetts-Amherst, 1996).

A Multiple Depth Buffer Implementation for Radiosity

Roel Martínez[1], László Szirmay-Kalos[2], and Mateu Sbert[1]

[1] Institut d'Informàtica i Aplicacions, Universitat de Girona.
{roel,mateu}@ima.udg.es
http://ima.udg.es/∼roel
[2] Department of Control Engineering and Information Technology.
Technical University of Budapest.
szirmay@iit.bme.hu

Abstract. This paper presents a depth buffer implementation of the global line multipath algorithm for radiosity. The implementation makes use of bundles of parallel lines implemented with OpenGL's depth buffer. For each projection direction the new algorithm uses several pair of depth buffers and exploits coherence between them in order to improve the efficiency. Each depth buffer stores the patch IDs that are projected onto them and a pair of depth buffers with opposite normals are used in order to exchange the energy between patches.

1 Introduction

Local Monte-Carlo approaches sample the domain of integration randomly using a probability density $p(\boldsymbol{x})$, evaluate the integrand $f(\boldsymbol{x})$, and provide the f/p ratio as the primary estimate of the integral. This estimate is accurate if we can find p that precisely mimics f, i.e. it makes f/p as flat as possible. This strategy, which is commonly referred as importance sampling, places more samples where the integrand is large. Since in practice p can be very far from the integrand, the estimator may have high variance. Thus to get an accurate result many independent primary estimators should be used to compute the secondary estimator as their average.

Global Monte-Carlo methods do not rely on the hope to find good sampling density. Instead, they take advantage of the fact that it is usually easy to evaluate function f at a well structured set of sample points $x_1, \ldots, x_n$. The emphasis is on that the simultaneous computation of $f(x_1), \ldots, f(x_n)$ is much cheaper than the individual computation of $f(x_1)$,..., and $f(x_n)$ by a local method, thus in this way we can have much more samples for the same computational effort. The disadvantage of this technique is that finding a probability density that simultaneously mimics the integrand at many points is very difficult, thus practical methods usually use uniform sampling probability. In this way, global methods are implemented using global uniformly distributed lines, in contraposition to "local" lines, generated from sampled points in the scene.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 346–355, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The multipath algorithm [11,12] belongs to the family of global lines algorithms which has seen a further developing in [1,3,14,15]. The objective of this paper is to demonstrate that it is possible and worth to implement the multipath method using acceleration techniques, particularly the OpenGL's depth buffer.

The rest of the paper is organized as follows. In section 2 we present the multipath method with single lines and also with bundles of lines. In section 3 we introduce the depth buffer implementation. In section 4 we show our results and finally present the conclusions.

2 Previous Work

2.1 Multipath Method

Global lines were first used in rendering by Buckalew in [4]. In [10] Sbert showed the intimate relationship of global lines to radiosity and used them to develop a full matrix radiosity method. The multipath algorithm for radiosity [11,12] eliminated the need for computing the form factors explicitly, and was shown in [12] that it could be interpreted as a random walk algorithm. In [11,12] the multipath was implemented with bundles of parallel rays, but the coherence properties have not been exploited. Meanwhile, Neumann [8] presented independently the transillumination algorithm, that has been further developed by [13], [14] and [15] in order to generalize it also for non-diffuse environments. These methods used the painter algorithm to generate global lines in bundles. In [2] the multipath was used in order to compute animated environments.

2.2 Multipath Algorithm with Single Lines

The multipath algorithm, described in [11,12], uses segments of global lines to build random walks that mimic classic random walks with infinite path length. The main differences between multipath and a classic (local) random walk approach are the source probability selection, the simultaneous advance of different paths thanks to global lines, and transporting different logical paths in a single geometrical one. The multipath method is only efficient in "smoothed" scenes, with more or less equilibrated emittance occupying a larger part of the scene. For this reason a *first shot* distributing direct illumination before applying the algorithm is necessary (see also [5] and [16]).

Multipath algorithm works as follows. A predetermined number of random global lines are cast using, for instance, pairs of random points on an enclosing sphere. Each line will produce an intersection list, and the list is traversed taking into account each successive pair of patches. Each patch (if not emitter) stores two quantities. One records the power accumulated, and the other the unshot power. For every pair of patches along the intersection list, the first patch of the pair will transmit its unsent power to the second patch of the pair. So the unshot energy of the first patch is reset to zero, and the two quantities at the second patch, the accumulated and the unsent energy, are incremented. In the

case of a source a third quantity is also kept, the emitted power per line exiting the source. This power is precomputed in the following way: Given the number of lines to cast, the forecast number of lines passing through any source is found. This can be done with Integral Geometry methods [9]. The division of the total source power by this number of forecast lines gives the predicted power of one line. Then, if the first patch of a pair is a source patch, the power transported to the second patch of the pair will also include this predicted power portion.

2.3 Multipath with Bundles of Parallel Lines

In [6] the multipath method has been extended to transfer radiosity with bundles of parallel lines, which takes advantage of the ray-coherence. The bundle of parallel lines is simulated using a general purpose polygon filling algorithm, called the painter algorithm.

The bundle of parallel lines is created as follows (see figure 1). First the scene is wrapped in a sphere. Then a random point is selected on the surface of the sphere, and a plane orthogonal to the sphere is obtained. We call this plane the *projection plane*. The direction of the bundle of parallel lines is defined perpendicular to the plane. The projection plane is associated with XY coordinates, its normal defines the Z axis and is decomposed in n by m pixels.

Every pixel simulates a line, and all patches that are projected in the same pixel are used in order to compute the exchange of energy between them, having sorted according to their z values.

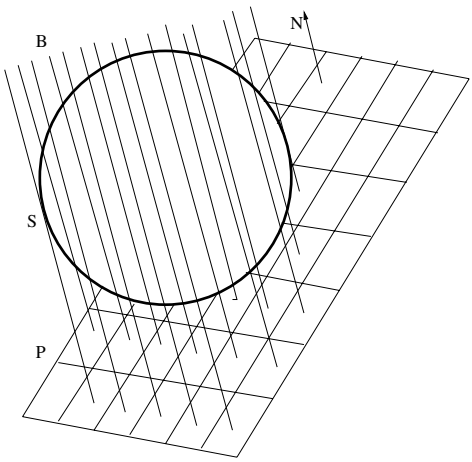


Fig. 1. Bundle of parallel lines, where **S** is the sphere that wraps the scene, **B** is the bundle of lines, **P** is the projection plane, orthogonal to **S** and **N** is the normal to the projection plane

2.4 Representative Projection

In [7] was presented a hardware based OpenGL depth buffer implementation of the multipath algorithm. The implementation was made for a double projection plane for each random direction.

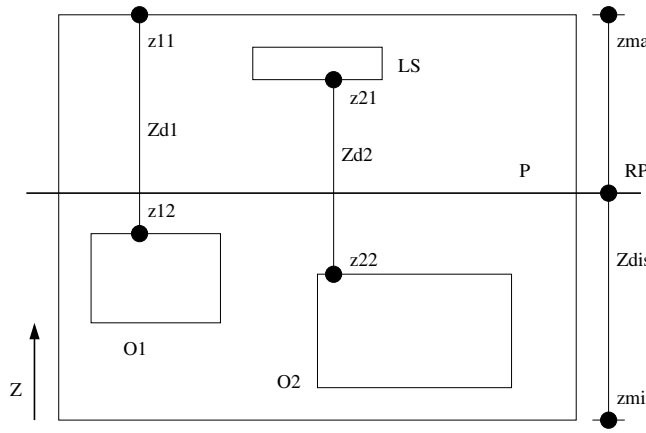


Fig. 2. The probability that the plane P, orthogonal to the Z-axis, crosses between object $O1$ and the ceiling is given by the distance between the object and the ceiling $Zd1$ divided by the maximum distance in the scene $Zdist$. Also, the probability that P crosses between $O2$ and LS is given by $Zd2/Zdist$. The Z-axis is parallel to the scene wall just for the sake of simplicity

Suppose that in a scene (see figure 2) we generate a global random direction. This direction defines the Z-axis of the scene. The Z coordinates of all scene patches are between $zmax$ and $zmin$, which are the maximum and minimum Z values of the scene. Let us generate a uniformly distributed random number z between $zmax$ and $zmin$ that defines a random point RP. A plane P orthogonal to the Z-axis is placed at RP. Defining a window on the clipping plane P to include the projection of all patches, let us run two depth buffers (with opposite viewing direction) with enabling and disabling the two half-spaces on the two sides of the clipping plane P. Reading back the images we have a set of mutually visible pixels which can be used to exchange energy. All mutually visible patches are represented by at least one pixel in the depth buffers. Note that here a pair of patches exchanges energy only with a given probability. Thus for each pair of pixels (that simulates a global line segment) for a correct balance of energy the actual energy exchange has to be divided by this probability. Now, the probability that the plane P crosses between a pair of points is the distance between them divided by the maximum Z distance (see figure 2).

An important advantage is that the OpenGL pipeline scales the Z values to be in [0..1] before writing them into the z-buffer. It means that at the end of

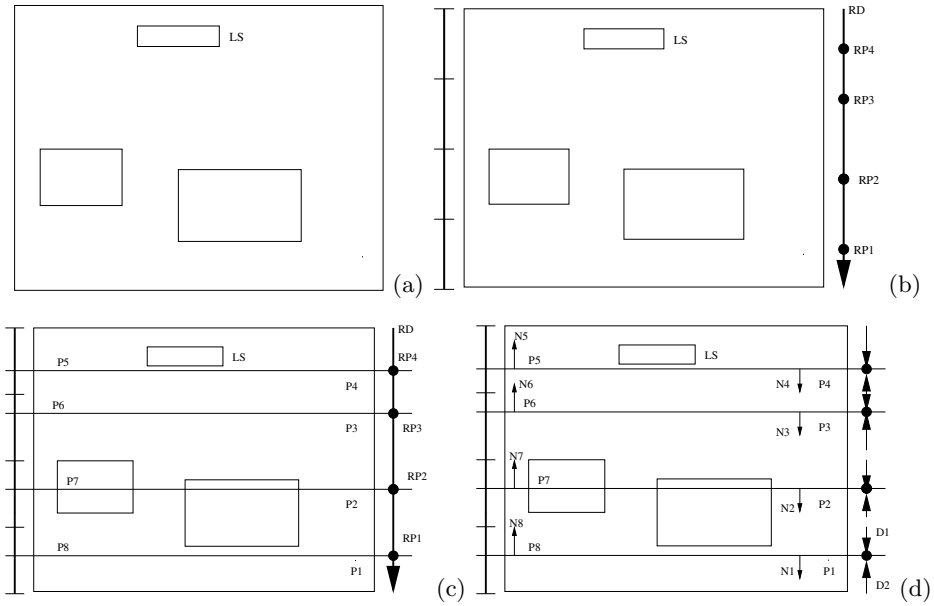


Fig. 3. (a) A simple scene with two objects and a light source (LS). (b) A random direction RD is created. The scene is divided in four intervals and for each interval a random point RP is selected. In this case, for simplicity, RD is parallel to one scene axis. (c) A pair of planes P with two opposite normals are created for each interval incident to RP, and are decomposed into $n \times n$ pixels. (d) Projection directions ($D1$ and $D2$) are defined. $D1$ has the same direction than RD and $D2$ is opposite to $D1$. N represents the normal to the plane

the transformation pipeline Zdist=1 (see figure 2), thus the calculation of the probabilities needs only one addition.

3 A New Algorithm: Multiple Representative Projections

The algorithm presented in [7], explained in section 2.4, uses one double projection plane for each random direction. Here we present a variant using several projection planes for the same random direction. The aim is to exploit coherence between projection planes for each iteration in order to improve the efficiency.

The bundles of parallel lines are obtained as follows: First, a random direction RD is selected. This direction defines the Z-axis. Using this direction all the scene patches are transformed into a new coordinate system and all patches are sorted using the z value of one of their vertices. Second, the minimum and maximum values of Z are computed. With these values the scene is divided in N equal intervals (where N is a user defined parameter, see figure 3(b)). Third, a random point RP between the minimum and maximum values of Z for each interval is selected (see figure 3(b)). The window plane, i.e. the projection plane, is defined

incident to RP and orthogonal to RD (see figure 3(c)), and will be used two times with two opposite viewing directions. This third part is repeated for all the intervals. Fourth, the projection of all scene patches is made in two steps. In the first one, there is a traversal from the minimun to the maximum value of Z. Only the patches that are part of the first interval are projected onto the projection plane using the OpenGL's depth buffer. After the projection, a copy of this projection plane is made. Here it is possible to exploit the coherence between projection planes. The projection plane for the second interval is the sum of the first projected plane plus the projected patches that are between this interval. For the rest of the intervals the same operation is applied. For example, figure 4(a) represents the $P4$ projection plane of the figure 3(d), where $P4 = P1 + P2 + P3$ plus the projected polygons between $RP3$ and $RP4$. In a second step the traversal is made from the maximum to minimum value of Z. Finally, the exchange of power is computed between the corresponding pixels of the two projection planes of each interval (see figure 5). Thus, for example, the first projection plane of the minimum-to-maximum traversal step exchanges energy with the last projection plane of the maximum-to-minimum traversal step.

This is summarized in the following algorithm.

```
begin sendBundle
   Compute a random direction RD (defining the Z-axis)
   Transform the scene (using RD)
   Sort all patches (using the Z values)
   Compute minimum and maximum Z values
   Divided in N intervals the scene
   Create 2*N CBuffers and 2*N ZBuffers
   Clean depthBuffer
   For k=0 to N do
      For m=minimumZ to maximumZ (for this interval) do
         Compute a random point RP
         Create a projection plane (orthogonal to RD, defined at
         RP and nxm size)
         Project patches onto projection plane
      endFor
      copy projection plane into CBuffer k
      copy Z values into ZBuffer k
   endFor
   Clean depthBuffer
   For k=0 to N do
      For m=maximumZ to minimumZ (for this interval) do
         Compute a random point RP
         Create a projection plane (orthogonal to RD, defined at RP
         and nxn size)
         Project patches onto projection plane
      endFor
```

```
    copy projection plane into CBuffer N+k
    copy Z values into ZBuffer N+k
endFor
end
```

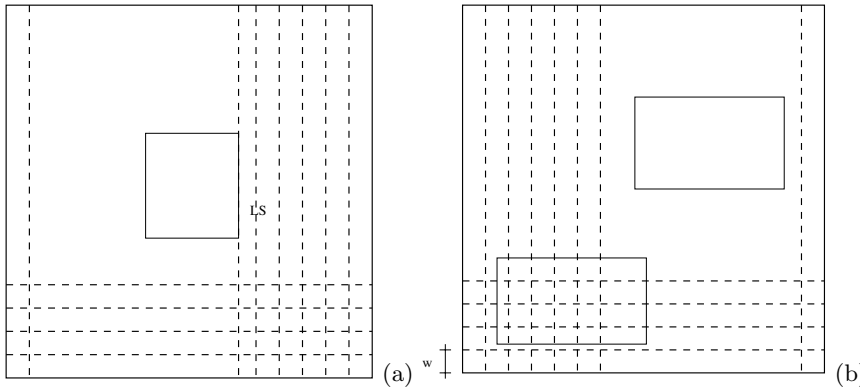


Fig. 4. After all patches (see figure 3(d)) are projected onto the projection planes $P4$ and $P5$, the result is (a) the lower section and (b) the upper section of the projected scene, where w is the pixel width of the projection plane

The discretized projection planes are represented by matrices with $n \times n$ pixels (n is defined by the user) and they store the closest patch IDs that are projected onto them. The two polygons identified by the IDs in the corresponding positions (for the two projection planes, see figure 5) will exchange power. The exchange power function is similar to the multipath single line implementation, explained in section 2.2. The only difference is that now the power is divided by the probability that a plane crosses between two patches. This probability is given by the sum of the two Z values as read out from the z-buffer (see section 3.).

4 Results

We have implemented the new algorithm in C++ using the offscreen MESA libraries and run the program on a Pentium IV 1.6 GHz Linux PC computer. A complete rendering consists of the computation of the first-shot, i.e. the determination of the direct illumination, and the multipath step that computes the indirect illumination. The first shot step was implemented with local lines.

We used two particular test scenes. The "big room" scene (figure 6) consists of 1130 polygons that have been subdivided into 27282 patches. The image was generated with 4 million local lines for the first shot and 100 random directions for the indirect illumination. The time consumed in each step was 86.61 and

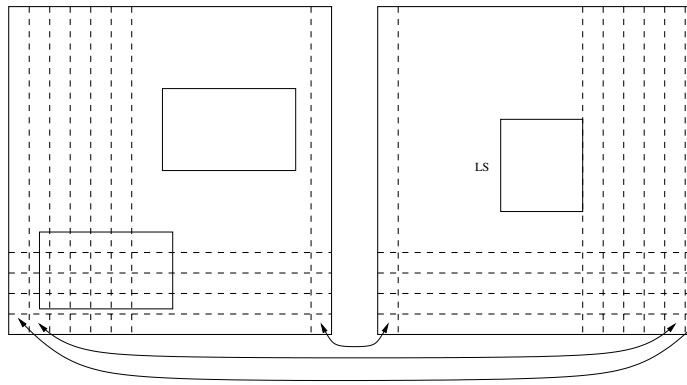


Fig. 5. Exchange of energy between two projection planes. For example, the projection plane $P4$ (see figure 3(d)) exchanges energy with projection plane $P5$

15.16 seconds, respectively. The "office" scene (figure 7) contains 547 polygons decomposed into 26322 patches. Four million local lines were cast in the first step and 100 bundles in the second one. The time consumed was 68.46 and 14.96 seconds, respectively.

The size of the depth buffer matrices is 128×128 pixels for both test scenes. For both executions in every random direction 4 pairs of projection planes were computed.

Finally, a comparison between the respective implementations with and without using coherence of the projection planes showed that the use of coherence reduces the computation time of indirect illumination to near the half.

5 Conclusions and Future Work

In this paper we have presented a OpenGL depth buffer implementation of the multipath algorithm. The new algorithm exploits the coherence of the projection planes for each iteration. The implementation uses global ray bundles to transfer the radiosity. Note that a single bundle may correspond to a lot of bi-directional rays, which can be traced efficiently.

Future work will be addressed to find the optimal number of intervals for each random direction, depending on the scene complexity, so as to obtain the maximum advantages of the projection plane coherence. On the other hand, non-diffuse multipath version or other related global line Monte Carlo methods are also ideal target for using OpenGL depth buffer implementations.

6 Acknowledgements

First and third author are supported in part by grants number TIC2001-2416-C03-01 from the Spanish Ministry of Science and Technology, 2001-SGR-00296

and ACI2002-29 from the Catalan Ministry of Universities, Research and the Information Society. Second author acknowledges financial support from the National Scientific Research Fund of Hungary (OTKA ref. No.: T042735) and the Bólyai Foundation.



Fig. 6. The image of the "big room" scene was obtained with the OpenGL depth buffer implementation. The image was generated with 4 million local rays (86.61 seconds) for the first shot and 100 iteration for the indirect illumination (15.16 seconds). In each iteration were computed 4 pair of projection planes

References

1. Bekaert, Ph.: Hierarchical and Stochastic Algorithms for Radiosity. Ph. D. Dissertation, Department of Computer Science, Katholieke Universitiet Leuven, Leuven, Belgium (1999)
2. Besuievsky, G., Sbert M.: The Multi-Frame Lighting Method: A Monte Carlo Based Solution for Radiosity in Dynamic Environments. Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering) (1996) 185-194
3. Besuievsky, G., Pueyo X.: Making Global Monte Carlo Methods Useful: An Adaptive Approach for Radiosity. Congreso Español de Informatica Gráfica CEIG'97, Barcelona, Spain (1997)
4. Buckalew, C., Fussell, D.: Illumination Networks: Fast Realistic rendering with General Reflectance Functions. Computer Graphics (ACM SIGGRAPH'89 Proceedings) Vol. 23 Num. 3, July (1989) 89-98
5. Castro, F., Martínez R., Sbert M.: Quasi Monte Carlo and extended first shot improvement to the multi-path method for radiosity. Proceedings of Spring Conference on Computer Graphics SCCG'98. Editor. Laszlo Szirmay-Kalos Budmerice, Slovakia, April (1998) 91-102
6. Martínez, R., Szirmay-Kalos, L., Sbert, M.: Adaptive Multipath with Bundles of Parallel Lines. Proceedings of the 3rd International Conference on Visual Computing Visual2000 México D.F. September (2000) 129-136
7. Martínez, R., Szirmay-Kalos, L., Sbert, M.: A Hardware Based Implementation of the Multipath Method. Proceedings of Computer Graphics International 2002 CGI2002. Bradford UK July (2002).
8. Neumann, L.: Monte Carlo Radiosity. Computing, Springer-Verlag (1995). 55, 23-42
9. Santaló, L. A.: Integral Geometry and Geometric Probability. Addison-Wesley New York (1976).



Fig. 7. The image of the "office" scene was obtained with the OpenGL depth buffer implementation. The image was generated with 4 million local rays for the first shot (68.46 seconds) and 100 iteration for the indirect illumination (14.96 seconds). In each iteration were computed 4 pair of projection planes

10. Sbert, M.: An Integral Geometry Based Method for Fast Form Factor Computation. Computer Graphics Forum (Eurographics'93), Vol. 12 Num. 3 Barcelona, Spain September (1993) C409-C420
11. Sbert, M, Pueyo, X., Neumann, L., Purgathofer, W.: Global multipath Monte Carlo algorithms for radiosity. The Visual Computer Vol. 12 Num. 2 (1996) 47-61
12. Sbert, M.: The Use of Global Random Directions to Compute Radiosity. Global Monte Carlo Techniques. PhD. dissertation, Universitat Politècnica de Catalunya, Barcelona, Spain March (1997).
13. Szirmay-Kalos, L., Fóris, T., Neumann, L., Csébfalvi, B.: An analysis to Quasi-Monte Carlo Integration Applied to the Transillumination Radiosity Method. Computer Graphics Forum (Eurographics '97 Proceedings) Vol. 16 Num. 3 (1997) C271-C281
14. Szirmay-Kalos, L., Purgathofer, W.: Global Ray-Bundle Tracing with Hardware Acceleration. Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98) Drettakis G. and Max N. Editors, Springer Wien (1998) 247-258
15. Szirmay-Kalos, L.: Stochastic Iteration for Non-diffuse Global Illumination. Computer Graphics Forum (Proc. Eurographics '99) Vol. 18, Num. 3 Milano September (1999) C233-C244
16. Szirmay-Kalos, L., Sbert, M., Martínez, R., Tobler, R.: Incoming First-Shot for Non-Diffuse Global Illumination. Proceedings of Spring Conference on Computer Graphics SCCG 2000 Budmerice, Slovakia (2000).

Solving Geometric Constraints Invariant Modulo the Similarity Group

Étienne Schramm and Pascal Schreck*

Laboratoire des Sciences de l'Image, de l'Informatique
et de la Télédétection UMR CNRS - Université Louis Pasteur 7005
Pôle API, Boulevard Sébastien Brant
F-67400 Illkirch - France
schramm@dea-dess-info.u-strasbg.fr, schreck@dpt-info.u-strasbg.fr

Abstract. In the area of Computer Aided Design (CAD), the geometric constraint systems considered are invariant under the direct isometry group. We show that the invariance under the direct similarity group also leads to interesting problems even in the CAD domain. This paper describes a method to solve such problems in the case of angle constraints involving three points. This method relies on extensions of euclidean notions described in earlier work and can be used in combination with other solvers.

1 Introduction

The problem of solving geometric constraints in CAD consists in yielding a figure which meets some metric requirements, usually specified under graphical form (See Fig. 1.). The result is often a graphical figure which is an ε-approximated numerical solution. So, the problem can be solved with usual iterative numerical methods like the Newton-Raphson's method or the continuation method [11, 6]. But, for some reasons, mainly stability and completeness, constructive methods are sometimes preferred. In these methods, the numerical values are abstracted and an history of the construction process is memorized. With our terminology, we say that the dimensionned sketch is transformed into a literal symbolic statement which is solved in two steps. The first step consists in a *symbolic* resolution and yields a construction program (or construction plan) which is a list of elementary operations to perform in order to produce the solutions, and the second step yields graphical figures through numerical interpretation of the construction plan.

One of the features of the CAD constraint systems is their invariance under the group of the direct isometries, also called direct Euclidean group. This property was exploited by various researchers in order to decompose a geometric constraint system into smaller ones if it is possible (see for instance [12, 16, 2, 4] for 2D-problems and [3, 6, 9] for 3D-problems). In [5] we have explained that

* To whom correspondence should be addressed

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 356–365, 2003.
© Springer-Verlag Berlin Heidelberg 2003

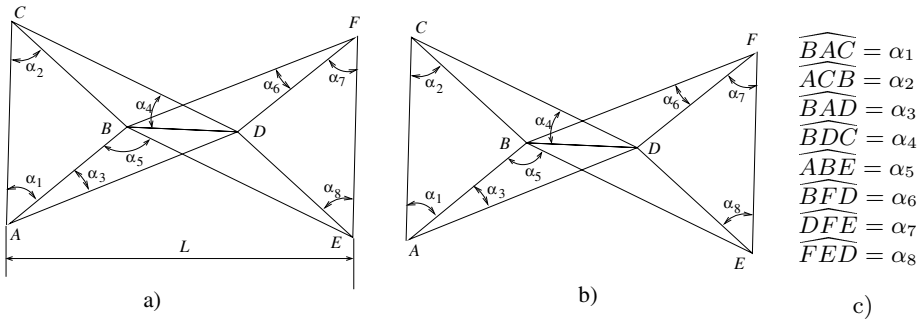


Fig. 1. Dimensionned sketches a) invariant under direct isometry group, b)using only angle constaints c) list of the angle constraints

the fundamental notions underlying these methods are the notions of reference, border and assembly process. Recall that a geometric type is a reference type for the direct Euclidean group if and only if it acts simply transitively on the objects of this type. The border of a solved sub-system within a constraint system consists in all the metric informations one can derive from the solved objects. When two sub-systems $\mathcal{S}_1$ and $\mathcal{S}_2$ are locally solved, for instance, $\mathcal{S}_1$ is solved by fixing reference r_1 and $\mathcal{S}_2$ is solved by fixing reference r_2, it is possible to assemble the two solutions if the unknowns shared by $\mathcal{S}_1$ and $\mathcal{S}_2$ determine more than a reference.

In this paper, we show that this framework can be extended to the similarity group. Note that this approach can help to solve difficult problems in 2D geometry constructions. For instance, the sketch in Fig. 1.a) is not constructively solved by the methods described in [12, 16, 2, 10], but releasing the distance constraint gives a problem invariant under the similarity group which is constructively solvable. Then using an appropriate similarity, one can find all the solutions for the sketch in 1.a). We explain how the notions of reference, border and assembly are transformed in order to take into account the new transformation group and how to solve problems like the one in Fig. 1.b).

The rest of this paper is organized as follows. Section 2 shows how to solve an example and gives some basic definitions. Section 3 explains how our framework can be translated into data structures, namely constraint schemes and S-clusters, in order to perform constraint propagation and assembly of local solutions. Section 4 explains how Chasles relation is used and sets out final data structures. Section 5 concludes.

2 First Definitions and Examples

Let **E** denote the Euclidean plane, the similarity group $S(2)$ is the set of transformations of **E** which preserve the size of angles. Recall that each similarity can be written as the composite of an isometry and a dilatation about O. $S^+(2)$

is the subgroup of the direct similarities, *i.e* the similarities which preserve the orientation. We have the first definition:

Definition 1. *A constraint* $\mathcal{C}(x_1, \ldots x_n)$ *is* invariant under the direct similarity group, in short S^+-invariant, *if and only if for any direct similarity* φ*:*

$$\mathcal{C}(x_1, \ldots x_n) \Leftrightarrow \mathcal{C}(\varphi(x_1), \ldots \varphi(x_n))$$

The usual S^+-invariant constraints concern the size of oriented angles and the ratio of distances. More precisely, in this framework, the following constraints are considered:

- $\angle(\overrightarrow{MN}, \overrightarrow{PQ}) = \alpha$ where α is a parameter;
- $\frac{MN}{PQ} = k$ where k is a parameter.

For the sake of simplicity, this paper is focused on systems containing only angle constraints under the form $\angle(\overrightarrow{PM}, \overrightarrow{PN}) = \alpha$, denoted more simply by $\widehat{MPN} = \alpha$. But, note that the behaviour of distance ratio constraints is very similar to the angle constraints. The sketch given in Fig. 1 shows graphically an example of a system of angle constraints. Let us solve"manually" this example using the loci method. Let $\mathcal{S}$ be the constraint system given in Fig. 1.c) where the unknowns are A, B, C, D, E and F. Note that, this system generally admits solutions.

First, since $\mathcal{S}$ is invariant under $S^+(2)$, there is an infinity of solutions for $\mathcal{S}$. But, each solution can be obtained from a finite number of them by using a similarity. More precisely, if we fix two points, we get a well constrained system, say $\mathcal{S}_1$, whose solutions are particular solutions of $\mathcal{S}$. Let $\mathcal{F}_1 = \{f_1, ...f_n\}$ be the set of solutions of $\mathcal{S}_1$; a figure f is solution for $\mathcal{S}$ if and only if there are a figure f_{i_0} in $\mathcal{F}_1$ and a similarity φ such that $f = \varphi(f_{i_0})$. We say that: the type *point* $\times$ *point* is a reference type. More generally, we have the definition:

Definition 2. *A type* $\mathcal{T}$ *is a* reference type for the direct similarity group*, in short an* S^+-reference type*, if and only if for any pair of geometric objects* (o_1, o_2)*, each of type* $\mathcal{T}$*, there is one, and only one, similarity* φ *such that* $\varphi(o_1) = o_2$*.*

Let us fix points A and C with the coordinates $(0, 0)$ and $(1, 0)$, then the lines (CB) and (AB) are constructible using angle values α_1 and α_2. So, point B is defined by the intersection of lines (CB) and (AB). Next, we can deduce line (AD) and an arc of circle defined by points B, C and angle α_4 (also known as the subtended arc (B, C, α_4)). So, point D is also constructible as the intersection of line (AD) and arc (B, C, α_4). No more points are deducible, but the problem is not yet completely solved: we only have a partial particular solution, we say a *local sub-solution*. Let us denote $ref_1 = \{A = (0, 0), C = (1, 0)\}$ the reference constraint used to fix points A and C and $\{A_1, C_1, B_1, D_1\}$ the local sub-solution obtained within reference ref_1.

Let us try now to solve $\mathcal{S}$ by fixing other points, for instance, points D and F, with coordinates $(0, 0)$ and $(0, 1)$. We have $D = A_1$, but it is, of course,

only a coincidence: to prevent any ambiguity, we call ref_2 this new reference constraint and we suffix the unknowns by 2. So, D_2 and F_2 are given and we can construct successively line (F_2E_2), subtended arc (D_2, F_2, α_8), point E_2, line (F_2B_2). Now, the angle value $\widehat{ABD}$ does not depend on the choice of any reference: so for any solution of $\mathcal{S}$, we have the same value for $\widehat{ABD}$. Hence, we have $\widehat{ABD} = \widehat{A_1B_1D_1} = \widehat{A_2B_2D_2}$. So knowing $\widehat{A_2B_2E_2} = \alpha_5$, we have by Chasles relation: $\alpha_9 = \widehat{D_2B_2E_2} = \widehat{A_1B_1D_1} - \alpha_5$. We can then construct subtended arc (D_2, E_2, α_9) and, finally, point B_2. We have used here the fact that any S^+-invariant property of a local solution is true for all the solutions: we call *the border of a local sub-solution* $\mathcal{F}$, the set of all the S^+-invariant constraints one can built with $\mathcal{F}$.

Definition 3. *If* $\mathcal{F} = \{M_1^1, M_1^2, \ldots M_1^k\}$ *is a local sub-solution of a system* $\mathcal{S}$*, the* border of $\mathcal{F}$ *in* $\mathcal{S}$ *is the set of all the constraints under the form* $f(M^i, ...M^j) = f(M_1^i, ...M_1^j)$ *for any function* f stable *under* $S^+(2)$.

At this point, we have two local sub-solutions: $\{A_1, C_1, B_1, D_1\}$ within reference ref_1 and $\{D_2, F_2, E_2, B_2\}$ within reference ref_2. Now, for every similarity φ, $\{\varphi(D_2), \varphi(F_2), \varphi(E_2), \varphi(B_2)\}$ is still a local sub-solution. Then, applying the direct similarity moving B_2 into B_1 and D_2 into D_1, we obtain a local sub-solution $\{D_1, F_1, E_1, B_1\}$ *within reference* ref_1. So, $\{A_1, C_1, B_1, D_1\, F_1, E_1\}$ is a particular solution for the whole system $\mathcal{S}$ within reference ref_1. We call such a process, consisting in determining the good similarity and applying it on a partial solution, the *assembly of the two local sub-solutions.*

3 Constraint Scheme

Let's now explain more formally the method we used to resolve constraint systems like the one given in example. First of all, we use a different terminology to clearly distinguish the geometric problem and the computational problem (with algorithms on data structures).

3.1 Constraint Scheme

So, we translate the notion of constraint system into a *valuated* and oriented hypergraph which we call *a constraint scheme.* We have:

Definition 4. *Given a set* V *of point symbols, a constraint scheme is a 3-tuple* $C =< V, A, v >$*, where* A *is a set of 3-tuples* (P_1, P_2, P_3)*, with* $P_1, P_2, P_3 \in V$ *and different one from another, and* $v : A \to \mathbb{R}$ *is a function which gives the value of the angle constraints.*

The set V of vertices corresponds to the unknown points, and each hyperedge (P_1, P_2, P_3) valuated by $v(P_1, P_2, P_3) = \alpha$ corresponds to the constraint $\widehat{P_1P_2P_3} = \alpha$. P_2 is called the *apex* of the constraint, while P_1 and P_3 are the

points of the base of the constraint. As we said before, we only deal with angle constraints between 3 points, so each hyperedge has degree 3.

A solution of a constraint system is a *figure* which associates the name of the unknown points with either numeric values, in the case of numeric resolution, or symbolic terms, in the case of symbolic resolution. In our combinatorial framework, a solution of a constraint scheme C can be viewed as a particular constraint scheme (V, A'), where $\forall\ P_1, P_2, P_3 \in V, (P_1, P_2, P_3) \in A'$. Indeed, if one knows the angle values between all points, one can construct all points after fixing two of them. We call such a complete hypergraph *S-cluster*. Conversely, given a fixed reference r defined by two points, a set of point symbols associated with coordinates related to reference r define an S-cluster because all the angle values are computable. The most elementary S-clusters only contain two points and no hyperedge –these are *trivial* S-clusters– while S-clusters with three points contain three hyperedges and correspond to triangles where one knows the three angle values. This notion is an extension of the classical notion of rigid cluster [2], or CD-set [15], replacing rigidity modulo the Euclidean group by rigidity modulo $S^+(2)$.

S-clusters are obviously not implemented by complete hypergraphs where most of the relationships between points are redundant. In fact, from a purely combinatorial point of view, S-cluster are only sets of vertices. Adding geometry, it is sufficient to represent them by sets of point symbols associated

- either with numeric coordinates within a fixed reference,
- or with symbolic terms specifiying a way to construct them.

In our terminology, the first case denotes *numeric clusters* since the second case denotes *symbolic clusters*. Each of these notions corresponds exactly to the notion of local sub-solution mentionned in section 2. The purpose of a solver is then to find an S-cluster containing all the vertices of V.

3.2 Constraint Propagation

Given a constraint scheme, a first way to build an S-cluster is to use constraint propagation. The constraint propagation method we use is an extension to hypergraphs of classical methods [1, 2]. The main idea is to decrease the degree of freedom of a point using the following fact: given an hyperedge, if two of its vertices are known, the degree of freedom of the third point is decreased by 1. Indeed, there are two cases in the dealing of the constraint $\widehat{ABC} = \alpha$

- if points A and B are known, then C is on the (oriented) line passing through B and making an angle of size α with line (BA). A similar construction occurs if B and C are known.
- it A and C are known, then point B is on the subtended arc (A, C, α)

So, knowing four points (not necessarily different) and two angle constraints linking a fifth point to them, one can construct it as the intersection of two loci among those described above. A first version of a propagation algorithm can be

Algorithm constraint propagation
Input - constraint scheme G which will be modified,
- two vertices P_1 and P_2 whose coordinates are fixed to (0,0) and (1,0)
Output - cluster C

- $C = \{ P_1, P_2\}$
- **while** G contains a vertex Q linked to C by hyperedges $h_1 \neq h_2$
  - compute the coordinates of Q within reference (P_1, P_2) as the intersection of the loci involved by h_1 and h_2
  - $C = C \cup \{Q\}$
  - delete h_1 and h_2 from G
- **return** C

Table 1. Pseudo-code for the propagation algorithm

deduced straightforwardly, as described in Table 1, but we have to add some comments. It is well known that loci intersections can yield more than one point (see, for instance, [4, 8, 13]). There are different ways to handle this difficulty: one can compare the current solution with the sketch [8] or use *multi-functions* and interpretation trees like in [7]. Another problem arises when the constraint system degenerates, for example, if two lines become parallel: this can be treated at the symbolic level like described in [14]. Since it is not the main subject of this paper, we do not consider here these difficulties. Furthermore, we chose to delete the used hyperedge to avoid to re-construct a part of a yet constructed S-cluster: we will see below that, thanks to our angle management, this does not remove any information. It is easy to see that we have:

Theorem 1. *Any S-cluster yielded by the algorithm* `constraint propagation` *from a constraint scheme G corresponds to a local sub-solution of the constraint system underlying G, according to the correctness of coordinates computation.*

3.3 Assembly of Two S-clusters

Let us go back to geometry and describe now the relationship between local sub-solutions. Let $\mathcal{S}$ be a constraint system and two local sub-solutions $\mathcal{F}_1 = \{M_1^1, \ldots, M_1^k\}$, within reference ref_1, and $\mathcal{F}_2 = \{P_2^1, \ldots, P_2^n\}$, within reference ref_2, of $\mathcal{S}$. That is: $\mathcal{F}_1$ and $\mathcal{F}_2$ are solutions for the constraint system $\mathcal{S}_1 = \mathcal{S}' \cup ref_1$ and $\mathcal{S}_2 = \mathcal{S}'' \cup ref_2$ with $\mathcal{S}' \subset \mathcal{S}$ and $\mathcal{S}'' \subset \mathcal{S}$. Suppose that $S = S' \cup S''$, and that the shared unknowns between S' and S'' define a reference r. These unknowns are solved within ref_1 to give objects which are part of $\mathcal{F}_1$. Let r_1 be this instance of r. We have similarly an instance r_2 of r which is a part of $\mathcal{F}_2$. By definition, there is only one direct similarity φ such that $\varphi(r_1) = r_2$. With these notations, we have:

Theorem 2. *The figure*

$$\mathcal{F} = \{P_2^1, \ldots, P_2^n\} \cup (\{\varphi(M_1^1), \ldots, \varphi(M_1^k)\} - r_2)$$

Algorithm assembly
Input - clusters C_1 and C_2 to be assembled
- two vertices P and Q shared by C_1 and C_2 that is $P_1 \in C_1$, $Q_1 \in C_1$, $P_2 \in C_2$ and $Q_2 \in C_2$
Output - cluster C_2

- compute $\varphi \in S^+(2)$ such that $\varphi(P_1) = P_2$ and $\varphi(Q_1) = Q_2$
- **for** M_1 **in** C_1 such that $M_1 \neq P_1$ and $M_1 \neq Q_1$ do
  - compute $M_2 = \varphi(M_1$
  - $C_2 = C_2 \cup \{M_2\}$
- **return** C_2

Table 2. Pseudo-code for the assembly of two clusters

is a solution of $\mathcal{S} \cup ref_2$.

Moreover, if we combine that way all the solutions of $\mathcal{S}_1$ *with all the solutions of* $\mathcal{S}_2$*, we obtain all the solutions of* $\mathcal{S} \cup ref_2$*. Consequently, we have all the solutions of* $\mathcal{S}$ *letting act* $S^+(2)$ *on the solutions of* $\mathcal{S} \cup ref_2$.

So, our solver must be able first to detect any pair of sub-solutions having a common reference and, second, perform the assembly of the two sub-solutions. Actually, with the S-cluster terminology, we have to detect if two S-clusters have two common points and then merging the corresponding S-clusters. The first point will be detailled below when our data structures will be clarified. The second point simply consists in computing the images by φ of all points in the moved cluster, exept the shared ones, and then making the union of the two sets like described in Table 2.

4 Chasles Relation and *S*-cluster

The algorithm we gave, works well on a large class of problems, but it cannot solve the constraint system given in Fig. 1. Indeed, there are some useful angle constraints that are not directly accessible: they must be computed using Chasles relation (that is: $\angle(\overrightarrow{MN}, \overrightarrow{PQ}) + \angle(\overrightarrow{PQ}, \overrightarrow{M'N'}) = \angle(\overrightarrow{MN}, \overrightarrow{M'N'})$) and the angle values given by the S-clusters computed before (remember that *all* the angle values "inside" an S-cluster are known). So, we must deal with *computed angle constraints* as well as with user constraints.

An efficient way to handle equivalence relations consists in using a forest as a data structure: each tree describes an equivalence class, each node in a tree being an element of the class. This can be used in our case: the Chasles relation applied to the angle constraints describes an equivalence relation which, in turn, can be described by a forest of *labeled trees of clusters*. Figure 2a) shows a simplified view of the forest coming from example 1: all the S-clusters are reduced to two points. As an example, the angle value $\widehat{CBE}$ can be deduced from the path going from CB to BE: we have $\widehat{CBE} = -\alpha_2 - \alpha_1 + \pi - \alpha_5$.

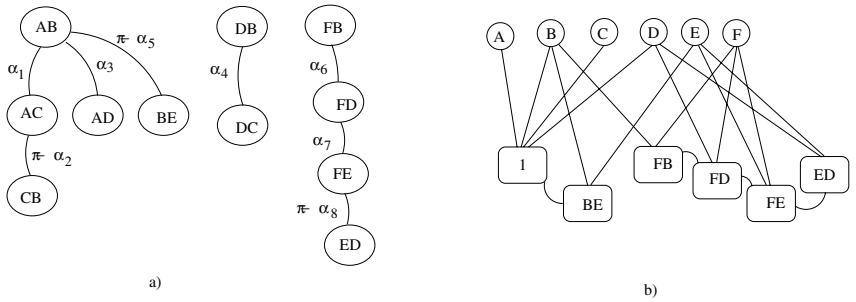


Fig. 2. Forest: a) simplified view; b) bipartite forest

Since a point can be owned by several S-clusters, it is more convenient to represent the incidence relation between points and S-clusters giving a bipartite forest. Figure 2b) shows the bipartite forest obtained in example 1 after the construction of local sub-solution within ref_1.

With these data structures, the propagation constraint algorithm given in previous section is now based on the search of points linked by an angle constraint to a given cluster C_1. Two cases may occur: either the apex of the angle constraint is in S-cluster C_1 (case a) in Fig. 3), or the apex is *not* in C_1(case b) in Fig. 3). When a point is found to be linked to an S-cluster C_1 with two different angle

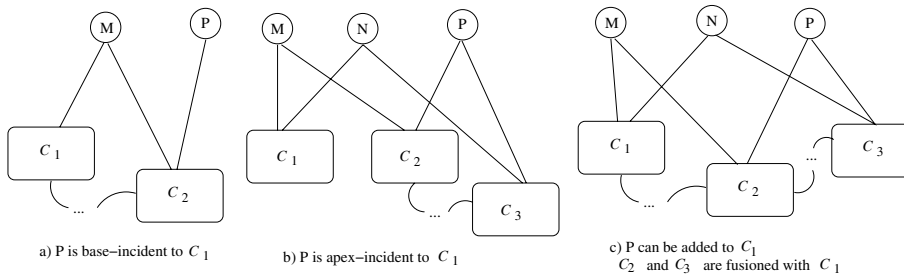


Fig. 3. Angle constraint incidence

constraints, using the algorithm given in Table 3, it is added to C_1.

We should then delete some angle constraints. But, since the auxiliary S-clusters, for instance C_2 and C_3 in the case described in Fig. 3 c), share two points with C_1, one can *immediately* assemble them. At the same time, the angle relation is updated, as decribed below, in order to reflect the new situation about the angle constraints.

Let us be more precise about the assembly process. First the discovery of a pair of points shared by two S-clusters is easily done using our data structures. The assembly of two S-clusters C_1 and C_2 is done like described in Table 2,

Algorithm search of a 2-angle incident point
Input S-cluster C_1
Output point P with a pertinent value or not_a_point

- $L_b = \emptyset$
- L_s=list of points shared by C_1 and an other S-cluster in the same tree
- found = false
- P=not_a_point
- **for** $M \in L_s$ **and while** (**not** found) do
  for $C \in$ list of clusters containing M and in the same tree than C_1
  and while (**not** found) do
  if $C \cap L_b = \emptyset$ **then** $L_b = C \cup L_b - \{M\}$
  else • choose P in $C \cap L_b$;
  • found = true
- **if** (found) **then return** P
  else do the same search for the apex-incident points starting with L_b
- **return** P

Table 3. Algorithm for finding points linked to C_1 by two different angle constraints

and the subtrees sons of C_1 are grafted as sons of cluster C_2: the angle values associated with the edges associated with C_1 are re-computed according to the reference of C_2. One important point during the assembly of two S-clusters is the removing of some edges of the tree. If there is an edge between C_1 and C_2, it has no sense to keep it, because all angle values are known in an S-cluster, by definition, so this edge must be deleted. In the same way, if, atfer the assembly, there are two or more edges between C_1 and another S-cluster, all but one can be removed. These situations occur after an immediate assembly, and so, in order to avoid redundancy, we absolutely have to do the immediate assembly every time we can.

This way, these data structures permit to efficiently implement algorithms given in the previous sections tacking Chasles relation into account. However, the naive search of the apex-incident point have a $O(n^3)$ complexity in the worst case and it must be improved.

5 Conclusion

Even if the actual challenge in geometric constraint solving lies in the 3D geometric constraints, there is a lot of interesting problems in 2D. In particular, this paper describes how to solve a large class of 2D geometric constraint systems using their invariance under the similarity group.

We explained here a simple limited framework with angle constraints, but even in this case, our method can solve difficult CAD problems, that are not decomposable with other methods. Moreover, this framework will be extended easily in order to take into account distance ratio constraints.

Our method can be used in combination with other methods, for instance within a multiagent architecture. But, a more interesting study concerns the exploitation of different invariance groups in the resolution of CAD problems: indeed, it seems to us that the decomposition of a geometric constraint system can be improved if the solver is able to identify some parts of the system and their greatest invariance group.

References

[1] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Proceedings of the Compugraphics Conference*, pages 83–92, 1993.

[2] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, 1995.

[3] B. Brüderlin. Constructing three-dimensional geometric objects defined by constraints. In *Interactive 3D Graphics*, pages 111–129, October 1986.

[4] J.-F. Dufourd, P. Mathis, and P. Schreck. Formal resolution of geometrical constraint systems by assembling. *Proceedings of the 4th ACM Solid Modeling conf.*, pages 271–284, 1997.

[5] J.-F. Dufourd, P. Mathis, and P. Schreck. Geometric construction by assembling solved subfigures. *Artificial Intelligence Journal*, 99(1):73–119, 1998.

[6] Cassiano B. Durand. *Symbolic and Numerical Techniques for Constraint Solving.* PhD thesis, Purdue University, 1998.

[7] C. Essert-Villard, P. Schreck, and F. Dufourd, J. Sketch-based pruning of a solution space within a formal geometric constraint solver. *Artificial Intelligence*, 124:139–159, 2000.

[8] I. Fudos and C. M Hoffmann. Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry and Applications*, page ???, 1994.

[9] X.-S. Gao, C. M. Hoffmann, and W.-Q. Yang. Solving spatial basic geometric constraint configurations with locus intersection. In *Proceedings of the ACM Solid Modeling Conference (Saarbrucken)*, pages 95–104, 2002.

[10] C. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition plans for feometric constraint problems, part ii : New algorithms. *J. Symbolic Computation*, 31:409–427, 2001.

[11] H. Lamure and D. Michelucci. Solving constraints by homotopy. In *Proceedings of the ACM-Sigraph Solid Modeling Conference*, pages 134–145. ACM Press, 1995.

[12] J. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the 1th ACM Symposium of Solid Modeling and CAD/CAM Applications*, pages 397–407. ACM Press, 1991.

[13] E. Schramm and P. Schreck. A case study in geometric constructions. In *Proceedings of the 2002 International Conference on Computational Science (Amsterdam), Part II*, pages 200–209. LNCS 2329, Springer, 2002.

[14] P. Schreck. Robustness in cad geometric construction. In *Proceedings of the fifth International Conference on Information Visualisation, IV2001 (London)*, pages 111–116. IEEE, 2001.

[15] G. Sunde. A CAD system with declarative specification of shape. In *Proceedings of the IFIP WG 5.2 on Geometric Modeling,* Rensselaerville, 1986.

[16] G. Sunde. Specification of shape by dimensions and other geometric constraints. In *Proceedings of the Eurographics Workshop on Intelligent CAD systems,* Noordwisjkerout, 1987.

Triangular Bézier Surfaces of Minimal Area*

A. Arnal[1], A. Lluch[1], and J. Monterde[2]

[1] Dep. de Matemàtiques,Universitat Jaume I
Castelló, Spain
parnal@mat.uji.es,lluch@mat.uji.es
[2] Dep. de Geometria i Topologia, Universitat de València,
Burjassot (València), Spain
monterde@uv.es

Abstract. We study some methods of obtaining approximations to surfaces of minimal area with prescribed border using triangular Bézier patches. Some methods deduced from a variational principle are proposed and compared with some masks.

1 Introduction

In this note we address the problem of finding the triangular Bézier patch minimizing the area, and saving then material costs, of the corresponding triangular Bézier surface with prescribed border. As it is well known, the border of a triangular Bézier surface is determined by the border control points. So, the problem can be reformulated as follows: Given the exterior points of a triangular control net, find out the inner ones in such a way that the resulting triangular Bézier surface had minimal area among all the triangular Bézier surfaces with the same border. Let us call this problem as the *triangular Bézier-Plateau problem.*

The theory of minimal surfaces shows that in order to prove the existence of minimal surfaces, one can replace the area functional, a highly non linear functional, by another one having the same extremals. The common substitute is the Dirichlet functional, so called in the mathematical literature, also called the stretch functional in the CAGD literature (see [6]).

The advantage of the use of the Dirichlet functional is that the determination of extremals becomes just a linear problem. Moreover, if a triangular Bézier chart is harmonic and isothermal it is extremal both of the area and Dirichlet functionals. At this point, we can compute the extremal for cubical triangular Bézier surfaces and to use the solution as a mask for obtaining approximations in higher degrees, or we can try to obtain the extremals of the Dirichlet functional in higher degrees directly.

The use of masks is due to the fact that the system of linear equations to solve is easier with their use, it is a sparse system, than the system deduced from the Dirichlet equations, which has no null coefficients.

* This work has been partially supported by Spanish Ministry of Sci. and Tech. BFM2002-00770 and Bancaixa-Universitat Jaume I, P1-1A2002-110.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 366–375, 2003.
© Springer-Verlag Berlin Heidelberg 2003

There is a difference with the same problem for rectangular Bézier surfaces (which has been treated in [5]). For usual Bézier surfaces, there is a very clear association between rectangular control nets and charts of the Bézier surface. In the triangular case, things are not so easy. The most direct association between triangular control nets and charts of triangular Bézier surfaces is the one obtained after the substitution of one of the barycentric coordinates by an expression depending of the other two having in mind the relation $u+v+w=1$. For example, if $\overrightarrow{y}(u,v,w)$ is the chart in barycentric coordinates, then we shall work with the chart $\overrightarrow{x}(u,v) = \overrightarrow{y}(u,v,1-u-v)$. Moreover, if we suppose now that (u,v) are cartesian coordinates, or equivalently, that the triangle used to define the barycentric coordinates is the non equilateral one whose vertices are $(0,0),(1,0)$ and $(0,1)$, then this association has a serious drawback: the breakdown of the symmetry. Even if the control net is symmetric ($P_{\sigma(i)\sigma(j)\sigma(k)} = P_{ijk}$ for any permutation σ), the Bézier surface does not preserve the symmetry.

Nevertheless we have followed this non symmetric approach due to the following facts:

1. Our approximations, although the use of non symmetric methods, seem to be at least as good as than other results deduced from symmetric methods.
2. We have checked that when the control net verifies some condition at the three corner points (isothermality) then our method shows, in general, a significative improvement.
3. For degree 3, there is a well known polynomial minimal surface, the Enneper's surface. It is possible to show that fixing the border control points as the ones of an arbitrary triangular piece of the Enneper surface, our asymmetric mask gives always exactly the inner control point. Moreover this is no longer true for any symmetric mask, and even it can be shown that there are pieces of the Enneper surface for which the inner control point cannot be obtained by applying a symmetric mask to the exterior control points.
4. Symmetric masks are deduced after some arguments on the control net, but not on the Bézier surface. Our methods are based directly on the Bézier surface because what we want is to minimize some functional directly related with the surface.

When isothermality at the three corner points is not satisfied, then the Dirichlet extremal considered as an approximation to the area extremal, presents an intrinsic error due to the method. In the last section we propose an improvement of the approximation based on geometric principles which maintain the use of linear systems. Now, the surfaces we obtain are of lesser area than the surfaces obtained by the other methods for the same border configurations.

2 Notation, Definitions, and Preliminary Results

2.1 Triangular Bézier Surfaces

Consider a triangle with vertices A, B, C and a fourth point P, all in $\mathbb{R}^3$. Then, it is always possible to write P as a barycentric combination of A, B, C

$$P = uA + vB + wC \quad \text{requiring that} \quad u+v+w=1.$$

The coefficients $\mathbf{u} = (u, v, w)$ are called *barycentric coordinates* of P respect to A, B, C. To build a triangular Bézier surface of degree n we have to repeat the barycentric interpolation analogously as we repeat the bilinear interpolation in the Casteljau Algorithm for constructing a Bézier surface. The control net for a triangular Bézier surface of degree n consists of $\frac{(n+1)(n+2)}{2}$ points arranged in a triangular grid. If $I = (i, j, k)$ each point of the triangular control net will be denoted by P_I. We will use also the following notation $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ and $|I| = i + j + k$.

We can express a triangular surface in terms of trivariate Bernstein polynomials: if $|I| \neq n$, then $B_I^n(\mathbf{u}) = 0$, else, i.e, if $|I| = n$ then

$$B_I^n(\mathbf{u}) = \binom{n}{I} u^i\, v^j\, w^k = \frac{n!}{i!j!k!}\, u^i\, v^j\, w^k.$$

These Bernstein polynomials, although they look trivariate, they are not, since $u + v + w = 1$. We will denote by $\mathcal{R}$ the region $\mathcal{R} = \{\mathbf{u} = (u, v, w)/u + v + w = 1 \text{ and } u, v, w \geq 0\}$ and by $\mathcal{T}$ the region $\mathcal{T} = \{(u, v) \in \mathbb{R}^2 : 0 \leq u, 0 \leq v, u+v \leq 1\}$.

Definition 1. *Given a triangular control net in* $\mathbb{R}^3$, $\mathcal{P} = \{P_I\}_{|I|=n}$, *the triangular Bézier surface of degree* n *associated to* $\mathcal{P}$, $\overrightarrow{x} : \mathcal{T} \longrightarrow \mathbb{R}^3$ *is given by:*

$$\overrightarrow{x}(\mathbf{u}) = \sum_{|I|=n} P_I B_I^n(\mathbf{u}).$$

A surface S is minimal if its mean curvature vanishes. Equivalently, S is a minimal surface iff for each point $p \in S$ one can chose a neighborhood, U_p, which has minimal area among other patches V having the same boundary as U_p.

A chart $\overrightarrow{x} : U \longrightarrow S$ of a surface is said to be isothermal if $E = G, F = 0$, being E, F, G the coefficients of the first fundamental form associated to the chart $\overrightarrow{x}$.

Proposition 1. *If a chart,* $\overrightarrow{x} : U \to S$, *of a surface,* S, *is isothermal, then* $\overrightarrow{x}(U)$ *is minimal if and only if the chart is harmonic, i.e.,* $\Delta \overrightarrow{x} = 0$.

3 The Dirichlet Functional Results

To solve the triangular Bézier-Plateau problem we have to try to minimize the functional area among all the triangular Bézier Surfaces with prescribed border determined by the exterior control points. Nevertheless, due to its high non linearity, the problem of minimizing the area functional is hard to manage with, so we shall work instead with the Dirichlet functional:

$$D(\mathcal{P}) = \frac{1}{2} \int_{\mathcal{T}} (\|\overrightarrow{x_u}\|^2 + \|\overrightarrow{x_v}\|^2) du\, dv.$$

There are two reasons for doing such a substitution: the first one is given by the following fact relating the area and the Dirichlet functional:

$$(E\,G\,-\,F^2)^{\frac{1}{2}} \leq (E\,G)^{\frac{1}{2}} \leq (E\,+\,G)/2. \tag{1}$$

Therefore, for any triangular control net $\mathcal{P}$, $A(\mathcal{P}) \leq D(\mathcal{P})$. Moreover, equality can occur only if $E = G$ and $F = 0$, i.e.: for isothermal charts.

The second one is related with the Euler-Lagrange equation associated to the Dirichlet functional defined not on control nets, but on charts

$$\overrightarrow{x} \to \frac{1}{2}\int_{\mathcal{T}} (\|\overrightarrow{x_u}\|^2 + \|\overrightarrow{x_v}\|^2) du\, dv.$$

This equation is just $\Delta\overrightarrow{x} = 0$. Therefore, if the extremal of the Dirichlet functional is an isothermal chart, it is automatically a harmonic chart, and then the surface is minimal. Nevertheless, we are not working with charts. We are working instead with triangular control nets. So, our aim is to find the minimum of the real function $\mathcal{P} \to \mathcal{D}(\overrightarrow{x}_{\mathcal{P}})$, being $\overrightarrow{x}_{\mathcal{P}}$ the triangular Bézier chart associated to the control net $\mathcal{P}$.

Proposition 2. *A triangular control net, $\mathcal{P} = \{P_I\}_{|I|=n}$, is an extremal of the Dirichlet functional with prescribed border if and only if:*

$$0 = \sum_{|I|=n} \frac{\binom{n-1}{I}}{\binom{2n-2}{I+I_0}}(a_1 + a_2 + 2a_3 - b_{13} - b_{23})\; P_I$$

for all $|I_0 = (i_0, j_0, k_0)| = n$ with $i_0, j_0, k_0 > 0$ and where:

$$a_1 = \begin{cases} 0 & i = 0, \\ \frac{i_0 i}{(i+i_0)(i+i_0-1)} & i > 0 \end{cases} \quad a_2 = \begin{cases} 0 & j = 0, \\ \frac{j_0 j}{(j+j_0)(j+j_0-1)} & j > 0 \end{cases} \quad a_3 = \begin{cases} 0 & k = 0, \\ \frac{k_0 k}{(k+k_0)(k+k_0-1)} & k > 0 \end{cases}$$

$$b_{12} = \frac{i_0 j + j_0 i}{(i+i_0)(j+j_0)} \qquad b_{13} = \frac{i_0 k + k_0 i}{(i+i_0)(k+k_0)} \qquad b_{23} = \frac{j_0 k + k_0 j}{(j+j_0)(k+k_0)}$$

In particular we give the general result for the case $n = 3$.

Proposition 3. *A triangular Bézier surface of degree 3 is an extremal of the Dirichlet functional with prescribed border if and only if*

$$P_{111} = \frac{1}{4}\,(2\,P_{003} - P_{021} + P_{030} + P_{120} - P_{201} + P_{210} + P_{300}).$$

4 Triangular Permanence Patches Related with the Bézier-Plateau Problem

Farin and Hansford define in [2] the triangular permanence patches as those triangular patches that satisfy the *permanence principle*: given a triangle $\mathcal{T}$ in the domain U of a triangular Coons patch, the three boundaries of this subpatch will map to three curves on the Coons patch, then the triangular Coons patch of those three boundaries is the original triangular Coons patch. This permanence principle can be established by using a mask like:

$$\begin{array}{ccccc} & \alpha & \beta & \beta & \alpha \\ & \beta & \star & \beta & \\ & \beta & \beta & & \\ & \alpha & & & \end{array}$$

with $3\alpha + 6\beta = 1$ (i.e.: $\alpha = \frac{1-6\beta}{3}$). Let us denote this mask by M_α.

It can be found in [2] that the mask M_0 is the discrete form of the Laplacian operator when the control net is considered as a discretization of the Bézier surface. Such a mask is used in the cited reference to obtain control nets resembling minimal surfaces that fit between given boundary polygons.

Another main masks are: $M_{\frac{1}{9}}$ that can be deduced by asking the quadrilaterals associated to the interior edges of the triangular patch to be as close as possible to parallelograms and mask $M_{\frac{1}{3}}$ that is the dual of M_0 in the sense that for $\alpha = \frac{1}{3}$ we have $\beta = 0$.

From the condition obtained in Proposition 3 we can try to generate, given the exterior control points the whole triangular net by solving a linear system where the equations are:

$$\begin{aligned} 4P_{i,j,k} = {} & 2\,P_{i+2,j-1,k-1} - P_{i,j-1,k+1} - P_{i,j+1,k-1} + P_{i-1,j+2,k-1} \\ & + P_{i-1,j+1,k} + P_{i-1,j,k+1} + P_{i-1,j-1,k+2}, \end{aligned}$$

being $P_{i,j,k}$ a inner control point. This equation can be expressed as the following mask, which will be called *the Dirichlet mask*:

$$\frac{1}{4} \times \begin{array}{ccccc} & 1 & & 1 & 1 & & 1 \\ & & -1 & & \star & -1 & \\ & & & 0 & 0 & & \\ & & & & 2 & & \end{array}$$

As we can see, the Dirichlet mask is not a mask like Farin-Hansford's mask because it is not symmetric. The asymmetry of the Dirichlet mask is due to the fact that the triangle on which we define the Bernstein polynomials is not an equilateral triangle, (see [7]). Applying a symmetrization process to the Dirichlet mask we obtain one of the masks worked in [2], that with $\alpha = \frac{1}{3}$.

5 Comparison between Masks

As it is said in [3], the natural question: *there is or not a better mask*, has a negative answer. It depends on the boundary conditions. In this section we will show some examples with simple boundary curves.

Case $\boldsymbol{n = 3}$. Let us start the comparison by studying some examples in the cubic case. We fix the three boundary curves with its control points and we construct the triangular Bézier surface computing the inner control point using the masks M_α and the Dirichlet mask. We have chosen some examples with

their border control points: along the border of a piece of the Enneper's surface in the first example; along two straight lines and a circle of radius 1 in NI1; along three circles of radius 1 in NI2, border Is is built in such a way that at the corner points any associated chart would be isothermal and finally border HNI is such that the isothermality conditions at its corners are far of being fulfilled. The following figures show the borders and the triangular Bézier surfaces constructed by means of the Dirichlet mask.

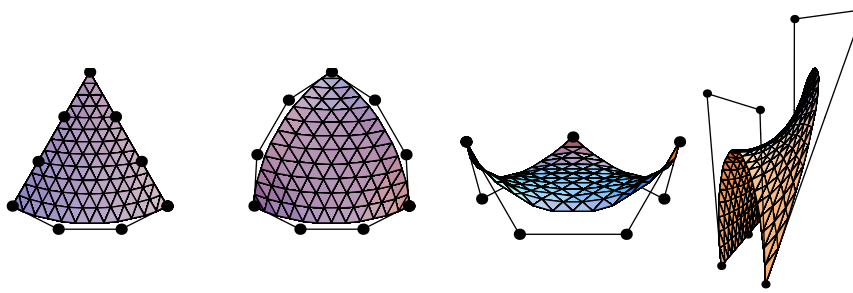


Fig.I: Surfaces NI1, NI2, Is and HNI by its Dirichlet Extremal.

In table I the areas of the corresponding triangular Bézier Surfaces are shown:

Method	Enneper	NI1	NI2	Is	HNI
M_0	4.67858	0.99685	1.21350	2.99046	13.22692
$M_{\frac{1}{9}}$	4.67835	0.99631	1.20844	2.88558	12.66618
$M_{\frac{1}{3}}$	4.67899	0.99563	1.20275	2.76656	11.67948
Dirichlet mask	4.67778	0.99793	1.20277	2.76957	12.22934
Dirichlet Correction	4.67778	0.99546	1.20216	2.75167	11.36520

Table I: Comparison between different masks for degree 3.

As we will introduce later the method that we have called Dirichlet correction, let us first analyze the results for the M_α masks and the Dirichlet mask. We can find that for these cubical examples the lesser areas are obtained by $M_{\frac{1}{3}}$ mask with the exception of the Enneper case that we will study in a later section.

Case $\boldsymbol{n = 10}$. Let us see in this section how things change with more degrees of freedom. The following examples for the case $n = 10$ are similar to the cubical examples NI2, Is and HNI. In NI2 we have chosen equally spaced border control points along the circles described before. At the other cases the choice of the borders has been done with the same configuration than before but with a slight modification in order to assure the isothermality at the corners in Is and the non isothermality in HNI. Table II shows the areas of the corresponding triangular Bézier surfaces by using the masks M_α, the Dirichlet mask and the area of the Dirichlet extremal, that is, the area of the triangular Bézier surface which inner control points are obtained by applying the Dirichlet equations in Prop. 2.

Method	NI2	Is	HNI
M_0 mask	1.34247	3.54592	12.61296
$M_{\frac{1}{9}}$ mask	1.34009	3.49978	12.53044
$M_{\frac{1}{3}}$ mask	1.33864	3.47307	12.47569
Dirichlet mask	1.33961	3.43799	12.68629
Dirichlet extremal	1.33963	3.43659	12.68513
Dirichlet Correction	1.33623	3.41091	12.42494
Second step	1.33625	3.37410	12.25581

Table II: Different masks and the Dirichlet extremal areas for $n = 10$.

In the NI2 and the HNI cases the best area is the one obtained using the $M_{\frac{1}{3}}$ mask, but now when we have the isothermality at the corners, case Is, the Dirichlet extremal is the one that give us the lesser area, even the use of the Dirichlet mask represents a significative improvement. An explanation of why the Dirichlet extremal has less area in Is will be given in section 7.

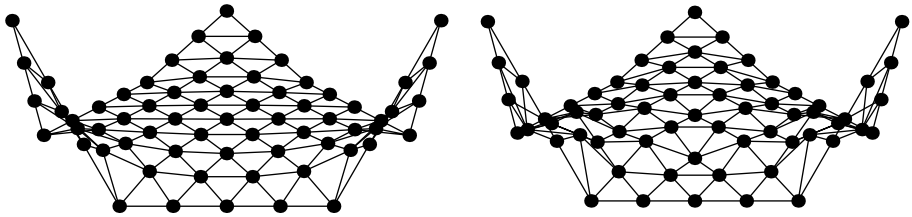


Figure II: Left M_0 and right Dirichlet mask control nets for Is surface.

Note the non regular shape of the control net of the right hand figure, the one obtained using the Dirichlet mask for degree 10, in comparison with the left hand figure, the one obtained by using the mask M_0, that is the mask for the discrete form of the Laplacian operator on the control net. The control net is not regular, but the associated Bézier surface is a better approximation to the minimal surface. Recall that we are looking for triangular Bézier surface minimizing some functional related with the surface, and not for triangular control nets minimizing some functional related with the net. The same fact also happens for rectangular Bézier surfaces.

6 The Enneper's Surface as a Testing Model

The first non trivial example of minimal surface with polynomial coordinate functions is the Enneper's surface (see [3] or [1] for some plots of this surface), $\overrightarrow{x} : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ defined by

$$\overrightarrow{x}(u,v) := (u - \frac{u^3}{3} + uv^2, v - \frac{v^3}{3} + vu^2, u^2 - v^2). \tag{2}$$

Therefore, it can be used to test the masks we have used. Moreover, as the chart (2) is isothermal, then it is an extremal no just of the area functional, but also of the Dirichlet functional. This means that, if we take a triangular piece of the Enneper's surface, we determine its control net, and we look for the extremal of the Dirichlet functional with that border control points, then the inner control point P_{111} is always given by the formula given in Proposition 3.

Nevertheless, there are cases where all symmetric masks fail to reobtain the inner control points. For example, let us consider the chart

$$\overrightarrow{y}(u,v) = \overrightarrow{x}(u+1,v), \qquad (u,v) \in T.$$

It is again an isothermal and harmonic chart, so it is a chart of a minimal triangular surface and the area of this triangular Bézier surface is 4.67778. The control net is

$$\begin{array}{ccccccc} (\frac{2}{3},0,1) & & (\frac{2}{3},\frac{2}{3},1) & & (1,\frac{4}{3},\frac{2}{3}) & & (\frac{5}{3},\frac{5}{3},0) \\ & (\frac{2}{3},0,\frac{5}{3}) & & (\frac{2}{3},1,\frac{5}{3}) & & (\frac{4}{3},2,\frac{4}{3}) & \\ & & (\frac{1}{3},0,\frac{8}{3}) & & (\frac{1}{3},\frac{5}{3},\frac{8}{3}) & & \\ & & & (-\frac{2}{3},0,4) & & & \end{array}$$

It is easy to check that this control net verifies the formula in Proposition 3. So, the inner control point P_{111} can be reobtained using the corresponding asymmetric mask.

Nevertheless, for a symmetric mask, the computation of the inner control points gives

$$P^{\alpha}_{111} = (\frac{13-9\alpha}{18}, \frac{17-21\alpha}{18}, \frac{5}{3}),$$

and there is no value for α such that $P_{111} = (\frac{2}{3}, 1, \frac{5}{3})$.

Moreover, the minimum of the area of the associated triangular Bézier surface with inner control point P^{α}_{111} is attained at $\alpha = 0.12833$ and its value is 4.67834. A possible explanation of the fact that in this case a non symmetric method gives better results that any symmetric method is the following: Let us recall first that the definition of triangular Bézier patches heavily depends on the triangle defining the barycentric coordinates. In this case, the coordinates u, v in the chart of the Enneper's surface are rectangular coordinates. After polarization, coordinates u, v, w are the barycentric coordinates with respect to triangle $\mathcal{T}$ which is not an equilateral triangle, and this fact breaks down the symmetry between the barycentric coordinates.

7 Correction of the Dirichlet Extremal

The obtainment of an approximation of the minimal Bézier surface according to the previous method has a serious drawback: the first fundamental form at the

corners of any triangular Bézier surface with prescribed border is determined by the border control points. For example, at the point $\overrightarrow{x}(0,0)$ the three coefficients of the first fundamental form are determined by the control points $P_{0,0,n}, P_{0,1,n-1}$ and $P_{1,0,n-1}$. Therefore, since the three points are border control points, the coefficients E, F and G at $\overrightarrow{x}(0,0)$ of any triangular Bézier chart with the same border will be always the same, even for the Dirichlet extremal, no matter which are the inner control points.

Let us recall that the Dirichlet method is based in the fact that the substitution of the area functional by the Dirichlet one will cause a negligible error. Moreover, both functional agree for isothermal charts. But if the configuration of the border control points is such that at the corner points the chart is always non isothermal, then the inequalities in (1) are strict. The non isothermality at corners points will produce an error when substituting the area functional by the Dirichlet one. At different points from the corner points, the configuration of the Dirichlet extremal tends to the isothermality of the chart. But at the corners points, isothermality or not is fixed from the border control points and it can not be modified. This is why the Dirichlet extremal does not improve the results of the harmonic mask in some cases.

We will propose along this section a method that will obtain, from the Dirichlet extremal as a first approximation to the minimal Bézier surface, a new and better approximation trying to avoid this problem and maintaining the fact that the new approximation is computed thanks to a system of linear equations.

Let us recall the following fact about the Dirichlet functional. As we have mentioned before, the Euler-Lagrange equation of the Dirichlet functional defined on the set of all differentiable charts with prescribed border is

$$\Delta \overrightarrow{x} = 0,$$

where Δ is the usual Laplacian operator. This equation is related to minimal surfaces thanks to Proposition 1. But there is another main result that does not mention the isothermality condition.

Proposition 4. *A chart $\overrightarrow{x}$ is minimal iff $\Delta^g \overrightarrow{x} = 0$ where g represents the first fundamental form of $\overrightarrow{x}$ and Δ^g is the associated Laplacian operator: for a function f:*

$$\Delta^g f = \left(\frac{f_u G - f_v F}{\sqrt{EG-F^2}}\right)_u + \left(\frac{-f_u F + f_v G}{\sqrt{EG-F^2}}\right)_v.$$

It is easy to check that, for a given metric, g, with coefficients E, F and G, the equation $\Delta^g \overrightarrow{x} = 0$ is the Euler-Lagrange equation of the functional

$$\begin{aligned}\mathcal{D}^g(\overrightarrow{x}) &= \int_{\mathcal{T}} \left(\frac{||\overrightarrow{x}_u||^2 G - 2<\overrightarrow{x}_u, \overrightarrow{x}_v> F + ||\overrightarrow{x}_v||^2 E}{\sqrt{EG-F^2}}\right) dudv \\ &= \int_{\mathcal{T}} g^{-1}(d\overrightarrow{x}, d\overrightarrow{x})\mu_g,\end{aligned}$$

where $\mu_g = \sqrt{EG-F^2}dudv$ is the metric volume element.

Note that for a given g, the extremal of the functional $\mathcal{D}^g$ is a chart that can be computed thanks to a linear system. Therefore, the correction of the Dirichlet

method is the following: let $\overrightarrow{x}_0$ be the Dirichlet extremal and let g_0 be its first fundamental form. The new approximation is the extremal of the functional $\mathcal{D}^{g_0}$, this is, using the Dirichlet extremal as the fixed metric. Note that the functional $\overrightarrow{x} \to \mathcal{D}^{g_0}(\overrightarrow{x})$ is quadratic in $\overrightarrow{x}$. Therefore the extremal equations are linear.

Proposition 5. *A triangular control net, $\mathcal{P} = \{P_I\}_{|I|=n}$, is an extremal of the functional $\mathcal{D}^{g_0}$ with prescribed border if and only if:*

$$\begin{aligned} 0 = \sum_{|I|=n} \frac{\binom{n-1}{I}}{\binom{2n-2}{I+I_0}} \Big(& \int_{\mathcal{T}} \frac{G_0}{\mu_{g_0}} a_1 \, B^{2n-2}_{I_0+I-2e_1} + \int_{\mathcal{T}} \frac{E_0}{\mu_{g_0}} a_2 \, B^{2n-2}_{I_0+I-2e_2} \\ & + \int_{\mathcal{T}} \frac{G_0 - 2F_0 + E_0}{\mu_{g_0}} a_3 \, B^{2n-2}_{I_0+I-2e_3} - \int_{\mathcal{T}} \frac{G_0 - F_0}{\mu_{g_0}} b_{13} \, B^{2n-2}_{I_0+I-e_1-e_3} \\ & - \int_{\mathcal{T}} \frac{F_0}{\mu_{g_0}} b_{12} \, B^{2n-2}_{I_0+I-e_1-e_2} - \int_{\mathcal{T}} \frac{E_0 - F_0}{\mu_{g_0}} b_{23} \, B^{2n-2}_{I_0+I-e_2-e_3} \Big) \; P_I \end{aligned}$$

for all $|I_0 = (i_0, j_0, k_0)| = n$ with $i_0, j_0, k_0 > 0$ and where a_s and b_{rt} were defined in Proposition 2.

The formulas obtained in Prop. 5 give us a system of linear equations for the interior points of the triangular net given its border. Now if we have a look at the last row in Table I and Table II, we can see that this method improves the results obtained through all the other methods, and moreover we get this improvement for all the examples, even when we deal with non isothermal charts.

Finally we have gone one step forward, if $\overrightarrow{x}_1$ is the Dirichlet extremal of the functional $\mathcal{D}^{g_0}$ and g_1 is its first fundamental form the new approximation is the extremal of the functional $\mathcal{D}^{g_1}$. The results obtained from this last method are shown in the last row at Table II, and from them we conjecture that, specially for highly non isothermal charts, the improvement given by the correction method can be even enhanced by repeating the process.

References

[1] C. Cosín, J. Monterde, *Bézier surfaces of minimal area*, Proceedings of the Int. Conf. of Computational Science, ICCS'2002, Amsterdam, eds. Sloot, Kenneth Tan, Dongarra, Hoekstra, LNCS 2330, vol II, pages 72–81, Springer-Verlag, (2002).

[2] G. Farin, D. Hansford, *Discrete Coons patches*, CAGD 16, 691–700 (1999).

[3] J. Gallier, *Curves and surfaces in Geometric Modeling*, Morgan Kaufmann publishers, S.Francisco, California, (2000).

[4] J.Hoscheck and D.Lasser, *Fundamentals of Computer Aided Geometric Design*, A.K.Peters, Wellesley,(1993).

[5] J. Monterde, *Bézier surfaces of minimal area: The Dirichlet approach*, preprint.

[6] R. Veltkamp, W. Wesselink, *Variational Modeling of triangular Bézier Surfaces*, pr.

Projection Filling Based on Contour Structural Points

Donggang Yu, Haizhen Dong, Wei Lai and Yun Yang

School of Information Technology, Swinburne University of Technology, PO Box 218, Hawthorn, VIC 3122, Australia
{dyu, hdong, wlai, yyang}@it.swin.edu.au

Abstract. This paper introduces a new method, called the projection filling, to fill any regions based on contour structure points. Four types of the structure points are detected based on the structure pattern models of chain codes of contours. The projection filling proceeds between each pair of the neighbouring structural points (filling section) parallelly. The new filling algorithm can fill any regions including some contours which can not be filled by traditional filling algorithms, such as the "seed" and the "even-odd" algorithms. The projection filling method can be used in image analysis, recognition and computer graphic**s.**

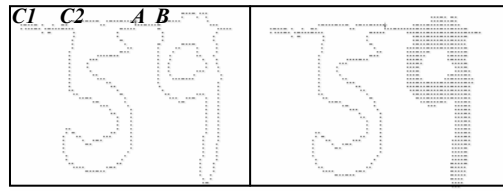
1 Introduction

In image processing and computer graphics, it is often required to fill regions of an image object [1-3]. Several methods have been developed in the past to solve the problem [1-9]. The regions enclosed by such simple contours (e.g. rectangle and polygon) are relatively easy to be filled. The regions enclosed by more complex contours are difficult to be filled. As J. D. FOLEY etc and J. K. WU [8-9] pointed out that filling result is ambiguous for some complicated contour regions. When we did some research on separation and recognition of hand written digits, we found some contour regions cannot be filled correctly by these methods, "seed" and "even-odd" algorithms. This can be seen in the following examples. When we use those traditional filling algorithms such as the "seed" and "even-odd" algorithms [1, 3, 4, 9] to fill regions in Fig. 1(1) and Fig. 2(1), there are some problems. If the link between two outer contours of an image is a line, the part of the outer contour can be missed out of filling by the "seed" algorithm. For instance, if the "seed" is on the left sides (or the right side), it can not pass from point *A* to *B* (or from point *B* to *A*) to enter the another part of the outer contour (see Fig. 1(1) and Fig. 2(1)). Suppose that the "seed" is chosen as the top-left point of contours. The incorrect filling results are shown in Fig. 1(2) and Fig. 2(2) by using the 'seed" algorithm. The "even-odd" filling algorithm cannot process well for expected regions to be filled in Fig. 1(1) and Fig. 2(1) and due to some special lines such as lines (*C1 C2*) in Fig. 1(1) and Fig. 2(1). Therefore no correct filling can be obtained in these contours. The improved edge fill method [2] can only fill the contours of standard alpha-numeric character images but it is not efficient for the complicated contour regions. Other filling method uses poly-

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2667, pp. 376-386, 2003.
© Springer-Verlag Berlin Heidelberg 2003

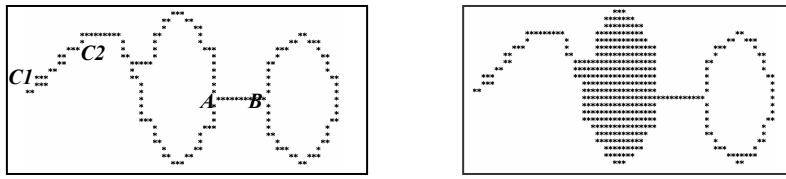
gons as contour line information [7, 8], but faults appear at line conjunction points in the object [2].

In this paper, we develop a new filling method, project filling, to fill any regions, which can fill all above regions which cannot be filled by other algorithms.



(1) (2)

Fig. 1: Example 1 of incorrect filling.



(1) (2)

Fig. 2: Example 2 of no correct filling.

In Section 2 how to detect structural points is described. In Section 3 how to determine filling sections based on the structural points is introduced, and the filling rules in each filling region are studied. Finally, conclusion is given.

2 Structural Points Detection

Suppose P is a binary image, S is an object with pixel value 1, $\overline{S}$ is the background with pixel value 0 and S_k is the set of pixels of contour k in the image. The Freeman code (Code 0 – Code 7) is shown in Fig. 3(1). If b_i $\{0 \le b_i \le 7\}$ is the next contour pixel, then the index number i is the direction chain code of the pixel p. For pixel $p \in P$, we call $NB_p = (b_7 b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)$ the 8-neighborhood byte of p as shown in Fig. 3(2) [10]. For pixel $p \in P$, if p is a contour pixel, then one neighbouring pixel in NB_p must be the next contour pixel in the sequence. If $\overline{S}$ is 8-connected, then both S and S_k are four-connected. If $\overline{S}$ is 4-connected, then S and S_k are 8-connected. In this paper we assume that S_k is 8-connected.

In this paper, the basic contour following rule is that S is kept on the left of the contours. For convenience of describing the required structural points, the direction of the chain code of inter contour is transformed to keep S on the right of the inter con-

tour of an object. The starting point of a contour is the upper-left corner of the contour.

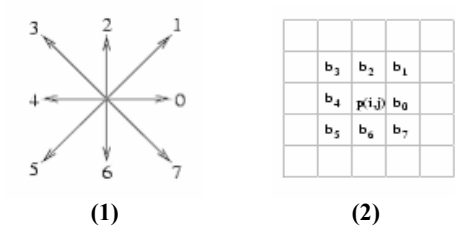


Fig. 3: Freeman code and neighbouring bytes.

2.1 Four Types of Structural Points on Outer Contours

Based on the above basic contour following rules, we can determine four types of structural points on outer contours based on the pattern models shown in Figs. 5-6.

(1) Structural point "^", convex point in Code 4.

These structural points on the outer contour are the convex point in direction chain code 4, which are determined based on all possible pattern models in Fig. 4(a-f). The pattern model of Fig. 54a) can be described as follows:

(1.1) The starting point is the *i*-th point and its chain code is code 1, and the direction chain code of the (*i*+1)-th point is only code 0, 4, or 5.

(1.2) From the (*i*+1)-th point, there may be n points whose chain code is direction chain code 0. The direction chain code of the (*i*+*n*+1)-th point is direction chain code 4 (*n*=0,1...).

(1.3) The terminal point is the *j*-th point, its code of the (*j*-1)-th point is determined by the number of points (*k*) whose code is code 4 based on the contour following rules. If (*k*=0), the code of the terminal point is code 5. If (*k*=1) the code of the terminal point is code 5 or 6. If (*k*>1) the code of the terminal point is code 5, 6 or 7.

Other five models in Fig. 4(b-f) can similarly be described. The convex point in direction chain code 4 on the outer contour is represented by character "^".

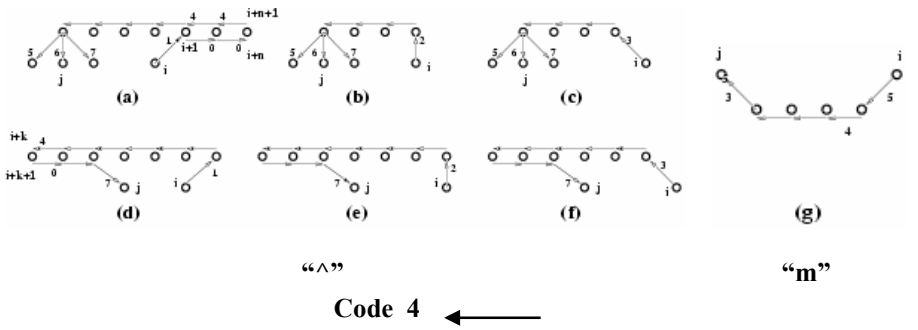


Fig. 4: Pattern models of structural points of the outer contour in Code 4.

(2) Structural point "m", the concave point in the Code 4.

This structural point is the concave point of the direction chain code 4 on the outer contour. It is represented by character "m". Based on the contour following rules there is only one pattern model for structural point "m" shown in Fig. 4(g).

(3) Structural point "v", convex point in the Code 0.

These structural points on the outer contour are determined based on all possible pattern models shown in Fig. 5(a-f). It is represented by character "v".

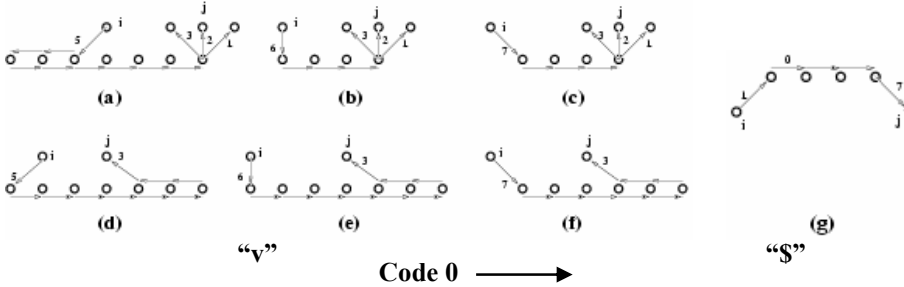


Fig. 5: Pattern models of structural points of outer contour in the Code 0.

(4) Structural point "$", the concave point in Code 0.

The fourth type of structural point is the concave point of direction chain code 0 on the outer contour. It is represented by character "$". Based on the contour following rules there is only one pattern model for structure point"$" shown in Fig. 5(g).

2.2 Four Types of Structural Points on Inter Contour

Four types of structural points on inter contours are based on the basic contour following rules and the pattern models shown in Figs. 7-8.

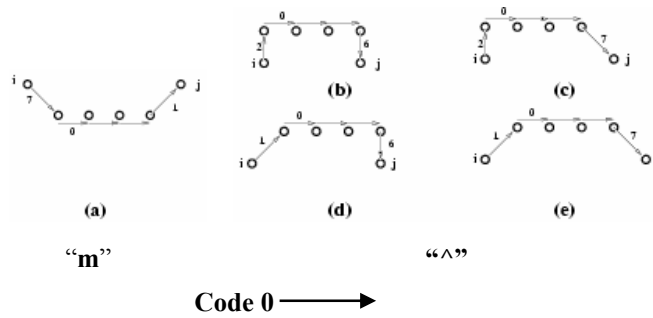


Fig. 6: Pattern models of structural points of inter contour in Code 0.

The convex and concave points in direction chain code 0 on the inter contour are represented by characters "^" and "m" which are determined based on the pattern models in Figs. 6(b-e) and (a) respectively. The convex and concave points in direc-

tion chain code 4 on the inter contour are represented by characters "v" and "$" which are determined based on the pattern models in Figs. 7(b-e) and (a).

In image analysis of binary images and computer graphics, the properties of contours (being outer contour or inter contour), the direction chain codes of contours are given.

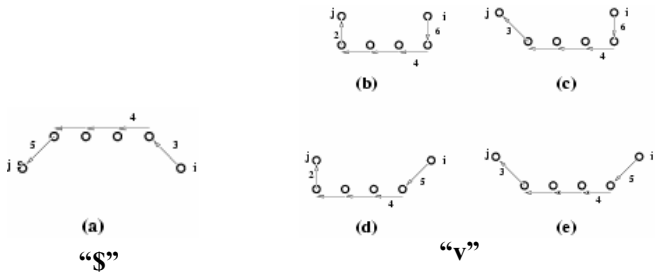


Fig. 7: Pattern models of structural points of in Contour in Code 0.

Therefore, the structural points of these given contours can be detected based on the above pattern models. For example, there is one outer contour (represented by O1), and there is one inter contour (represented by I1) in Fig. 8. Their structural points can be extracted based on the above models of structural points, and are shown in Fig. 8. There are two points "^", one point "m", two point "v" and one point "$" in both outer contour (O1) and inter contour (I1) in Fig. 8. In Fig. 9 there are one outer contour and four inter contours, and there are one point "^", one point "$" and two points "v" in the outer contour (O1) and there are one point "^" and one point "v" in four inter contour (I1, I2, I3 and I4) respectively.

3 The Projection Filling Algorithm

In this section, the projection filling algorithm is described. Firstly, we determine the filling sections of region contours. The filling section consists of the contour points between two special neighbouring structural points.

3.1 Finding Filling Sections

A filling section is detected based on the following contour properties:

(1) It is clear that the number of points "^" and "$" is equal to that of points "m" and "v" in each contour. Therefore the total number of structural points is even. Let it be *c*.

(2) In each contour the first structural point is structural point "^" and the second point of the order is point "v" or "m".

(3) For each structural point "^" or "$", its followed structural points are structural points "v" or "m" in a series of structural points of a contour.

(4) The filling section consists of contour points between two neighbouring structural points in a series of structural points of a contour. In the filling section the starting point is structural points "^" or "$". It is clear that the number of the filling section of a region contour is *c*/2.

(5) The filling sections of a contour are determined in the direction of the contour following.

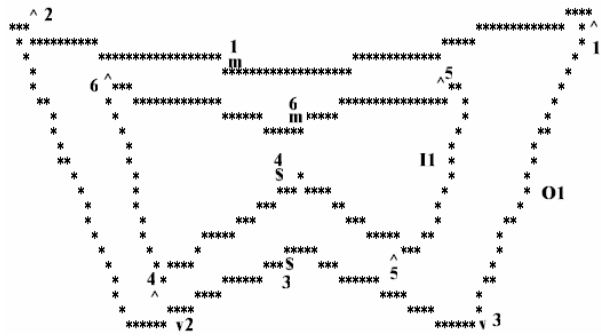


Fig. 8: Example 1 of structural points, filling sections and projection filling.

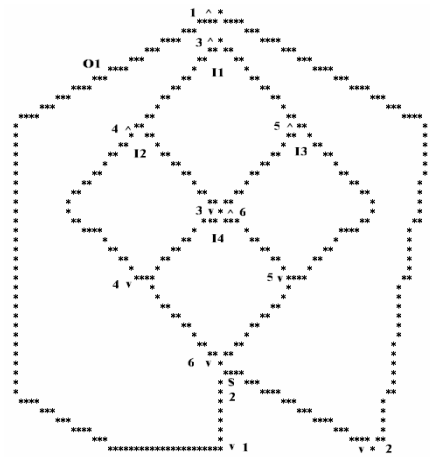


Fig. 9: Example 2 of structural points, filling sections and projection filling.

For example, the number of spoints "^" and "$" of the outer contour in Fig. 8 is equal to 3 (two points "^" and one point "$"), and the number of the points "v" and "m" in Fig. 8 is equal to 3 (two "v" points and one "m" point. The series of the structural points of this outer contour is: "^"->"m"->"^"->"v"->" $ "->"v" in the direction of the contour following. Therefore three filling sections of the outer contour can be found based on the above definition of the filling section. Three filling sections are "^"->"v", "$"->"v" and "^"->"m" respectively.

In the first filling section the starting point is structural point "^", and the terminal point is structural point "m".
In Fig. 9 there are two filling sections, "^"->"v" and "$"->"v", in one outer contour (O1), and there is one filling section, "^"->"v", in each inter contour (I1, I2, I3 and I4) respectively based on our algorithm.

3.2 Projection Filling

Projection filling is proceeding from left to right in the same row. Let the filling section include *n* points represented as follows:

$$X_k=\{x_0,x_1\ldots x_i,\ldots x_{n-1}\} \tag{1}$$

$$Y_k=\{y_0,y_1\ldots y_i,\ldots y_{n-1}\} \tag{2}$$

where X_k and Y_k are the x and y coordinate sets of the k-th point in the filling section respectively. It is clear that the first point of the filling section is structural points "^" or "$" and its x and y coordinates are x_0 and y_0, and the last point of the filling section is structural points "m" or "v" and its x and y coordinates are x_{n-1} and y_{n-1} respectively. All points in a filling section are starting projection filling point.

Procedures of outer contour projection filling:
Projection filling of outer contour is proceeding in the filling sections found in each outer contour.

(I) Projection filling of the first point (x_0, y_0) in filling sections of outer contours.

The first point (x_0, y_0) in each filling section is structural point "^" or "$". The filling is the horizontal projection from the left to the right.
Rule 1: If the first point in a filling section is point "^", no horizontal projection point is made.
Rule 2: If the first point in a filling section is "$", there are three cases as following:

- **(1)** If point "$" only belongs to one contour point, then:
  - **(1.1)** One horizontal projection filling point (x_0+1, y_0) is made.
  - **(1.2)** Check if filling point (x_0+1, y_0) intersects with any point of other contours in the object image. If it is true, stop filling. Otherwise, make the next horizontal projection filling point (x_0+2, y_0)
  - **(1.3)** Similar to Step (1.2) the filling proceeds successively until no new projection filling point is made.
- **(2)** If point"$" belongs to more than one contour and next starting project filling point (x_1, y_1) in the filling section only belongs to one contour, then:
  - **(2.1)** One horizontal projection filling point (x_0+1, y_0) is made.
  - **(2.2)** The procedure is the same as above Steps (1.2) and (1.3).
- **(3)** If point ``$" belongs to more than one contour and next starting project point (x1, y1) in the filling section belongs to more than one contour, then no horizontal projection filling point (x0+1, y0) is made.

(II) Projection filling of middle points (x_k, y_k) (k = 2, …n-2) in filling sections of outer contours.

Rule 3: Starting projection filling points (x_k, y_k) (k = 2, …n-2) are the middle points in a filling section. For each point (x_k, y_k) of these points there are three cases of projection filling as following:

(1) If this point (x_k, y_k) only belongs to one contour, then:
- **(1.1)** One horizontal projection filling point (x_k+1, y_k) is made.
- **(1.2)** Check if filling point (x_k+1, y_k) intersects with any point of other contours in the object image. If it is true, stop filling. Otherwise, make the next horizontal projection filling point (x_k+2, y_k).
- **(1.3)** Similar to Step (1.2) the filling proceeds successively until no new projection filling point is made.

(2) If this point belongs to more than one contour and next starting project filling point (x_{k+1}, y_{k+1}) in the filling section only belongs to one contour point, then:
- **(2.1)** One horizontal projection filling point (x_k+1, y_k) is made.
- **(2.2)** The procedure is the same as above Steps (1.2) and (1.3).

(3) If this point belongs to more than one contour and next starting project point (x_{k+1}, y_{k+1}) in the filling section belongs to more than one contour, then no horizontal projection filling point (x_k+1, y_k) is made.

(III) Projection filling of the last point (x_{n-1}, y_{n-1}) in filling sections of outer contours.

The last point (x_{n-1}, y_{n-1}) in each filling section is structural points "m" or "v". The filling procedure of the last point can be described as follows:

Rule 4: If the last point (x_{n-1}, y_{n-1}) is point "v", no horizontal projection point is made.

Rule 5: If the last point (x_{n-1}, y_{n-1}) in a filling section is "m", there are three cases as following:

(1) If point "m" only belongs to one contour, then:
- **(1.1)** One horizontal projection filling point (x_{n-1}+1, y_{n-1}) is made.
- **(1.2)** Check if filling point (x_{n-1}+1, y_{n-1}) intersects with any point of other contours in the object image. If it is true, stop filling. Otherwise, make the next horizontal projection filling point (x_{n-1}+2, y_{n-1}).
- **(1.3)** Similar to Step (1.2) the filling proceeds successively until no new projection filling point is made.

(2) If point "m" belongs to more than one contour, then no horizontal projection filling point (x_{n-1}+1, y_{n-1}) is made.

Similar to the above project filling rules of outer contours, all rules of projection filling of inter contour can be found.

Based on the above filling algorithm, we can fill any region contours. For Fig. 8 its filling process is shown in Fig. 10. For the first filling section "^"->"v" of the outer contour its filling result is shown in Fig. 10(1), and for the first point, "^", in the filling section no projection filling point is made based on Rule 1, for the middle points projection filling points are made based on Rule 3, and for the last point, "v", no projection point is made based on Rule 4.

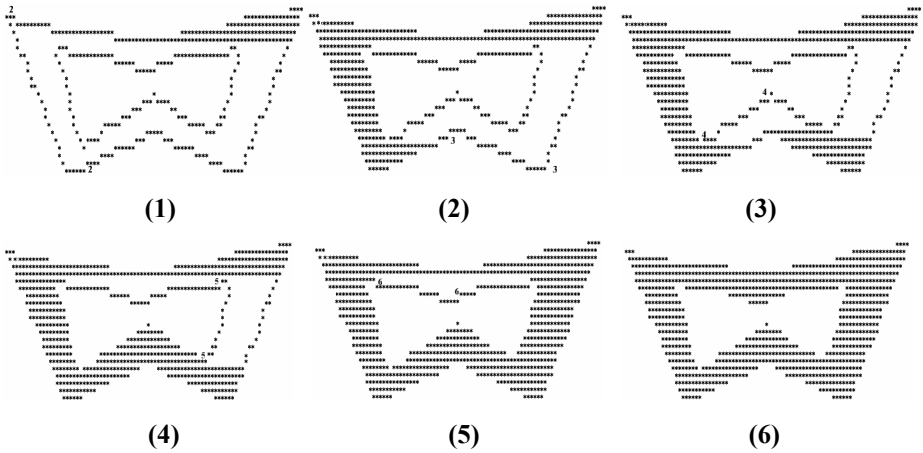


Fig. 10: Example 1 of projection filling.

For all filling sections of inter contours of Figs. 8-9, the projection filling can be proceeded based on the related filling rules of inter contours. The filling result of each step is shown in Figs. 10-11.

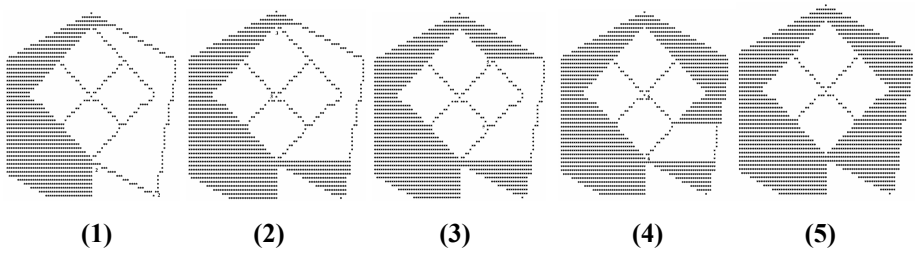


Fig. 11: Example 2 of projection filling.

It is necessary to point out that the filling of each point in all filling regions is independent, therefore the filling can proceed parallelly.

4 Conclusion

A new and efficient filling algorithm is developed. Structural points of each contour are detected, filling sections are determined, and the rules of projection filling are described. The method can be used to fill all types of contour correctly. The existing methods such as the "seed" algorithm and "even-odd" algorithm cannot fill some types of contour which shown in Fig. 1(1), Fig. 2(1) and Fig. 9, but our method can do it correctly. The filling process and result by our method are shown in Figs. 11-13 (only part of filling processes and results are being shown). Our algorithm is efficient and correct because the structural points are used and the morphological feature of structural points is considered in the filling procedures. Our algorithm has been used

for the recognition of handwritten characters, separation of touching handwritten strings and the reconstruction of broken handwritten digits with good results.

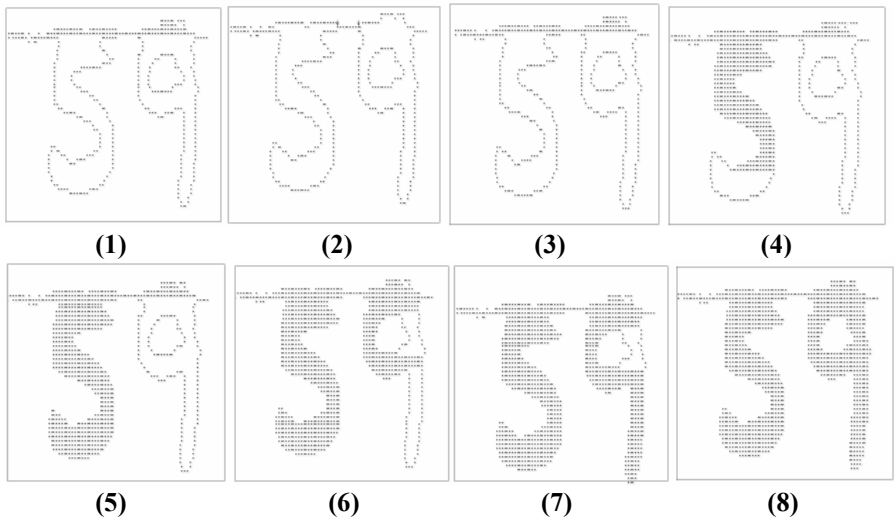


Fig. 12: Example 3 of projection filling.

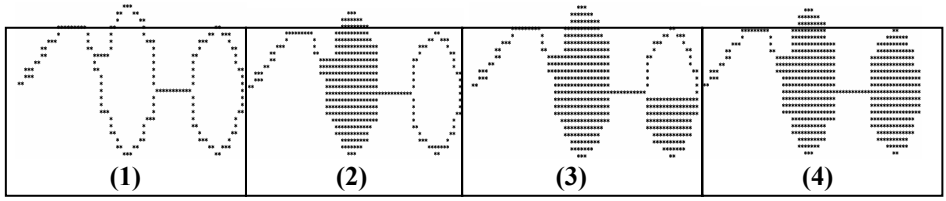


Fig. 13: Example 4 of projection filling.

Acknowledgement- This work is partly supported by an Australian Research Council SPIRT grant (C00107573).

References

1. T. PAVLIDIS: Algorithms for Graphics and Image Processing, Computer Science Press Rockville (1982).
2. K. NAKASHIMA, M. KOGA and K. MARUKAWA: A Contour Method for Alpha-numeric Character Image Generation, *Proc. Of the second international conference on document analysis and recognition* (1993) 722-725.
3. S. V. BURTSEV and Y. P. KUZMIN: An efficient Flood-filling Algorithm, *Comput. and Graphics*, **17, (5)** (1993) 549-561.

4. U. SHANI: Filling Regions in Binary Raster Images: A Graph-Theoretic Approach, *Proc. of SIGRAPH'80*, **14, (1)** (1980) 321-327.
5. T. PAVLIDIS: Filling Algorithms for Raster Graphics, *Comput. Graphics and Image process*, **10, (2)** (1979) 126-141.
6. T. PAVLIDIS: Contour Filling in Rater Graphics, *Pro. of SIGRAPH'81*, **15, (3)** (1981) 29-36.
7. R. D. MERILL: Representation of Contours and Region for Efficient ComputerSearch, *Commun. ACM*, **16** (1973) 69-82.
8. J. D. FOLEY, A. V. DAM, S. K. FEINER and J. F. HUGHES: *Computer Graphics*, Adisson-Wesley Publish Company (1 995).
9. J. K. Wu:Analysis of Digital Images, PEOPLE'S POSTS TELECOMMUNICATIONS PUBLISHING HOUSE (1989).
10. C. WANG, Y. QI, D. YU and S. Xu: A fast algorithm for boundary tracing of binary image with neighbourhood coding, *Proc. international conference on signal processing*, **2** (1990) 1083-1085.

A Graph Based Algorithm for Intersection of Subdivision Surfaces

S. Lanquetin, S. Foufou, H. Kheddouci and M. Neveu

LE2I, FRE CNRS 2309, UFR des Sciences et Techniques, Universit de Bourgogne,
BP 47870, 21078 DIJON Cedex, France
{slanquet, sfoufou, kheddouc, mneveu}@u-bourgogne.fr

Abstract. Computing surface intersections is a fundamental problem in geometric modeling. Any boolean operation can be seen as an intersection calculation followed by a selection of the parts necessary for building the surface of the resulting object. A robust and efficient algorithm to compute intersection on subdivision surfaces (surfaces generated by the Loop scheme) is proposed here. This algorithm relies on the concept of a bipartite graph which allows the reduction of the number of faces intersection tests. Intersection computations are accelerated by the use of the bipartite graph and the neighborhood of intersecting faces at a given level of subdivision to deduce intersecting faces at the following levels of subdivision.

1 Introduction

Computing intersection curves between two surfaces is not a new problem. Lots of works have been done on this subject. Parametric surfaces intersection methods can be classified into five main categories [5], [13], [1]: analytical methods [7], [13] which are preferred when one surface is expressed in the implicit form, and the other in the parametric form. Lattice evaluation methods, which reduce the surface/surface intersection problem to a set of curve/surface intersection problems [3]. Marching methods, which require at least one point on the intersection curve, called the starting point, to generate a sequence of points on the intersection using the local geometry of intersecting surfaces [2]. Subdivision methods use the face splitting of surfaces to find polygonal approximation of intersection curves. The reliability of these methods depends on the chosen level of subdivision and tools used to map out the subdivision tree. Mixed methods combine several ideas from the previous techniques.

Subdivision surfaces are particularly used to create characters or other complex forms in animation movies like Geri s Game [8], Final Fantasy or Monsters Inc. They are also used in modelers like LightWave 3D. Few works deal with the intersection of subdivision surfaces. We can note that of O Brien and Manocha [12] about this very subject, the various methods proposed by Linsen to join two nets [10], the trimming algorithm described by Litke in [11] and the contribution of Bierman on boolean operations applied to multiresolution surfaces [4].

For subdivision surfaces, the na ve algorithm which consists in intersecting each face of one surface with all the faces of the other surface is inefficient and time con-

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 387-396, 2003.
© Springer-Verlag Berlin Heidelberg 2003

suming. Indeed, let S_1 and S_2 be two surfaces respectively with 694 and 128 faces corresponding to the bunny and the tube in Figure 9. The number of faces increases in every subdivision step. For instance, in the case of Loop scheme, the number of faces becomes 4 times this of the previous level. At the third level of refinement, there will be 44 416 faces for the first surface and 8192 faces for the second one. Testing face intersection directly (the naïve algorithm) results on 6 144 000 000 comparisons. This is a very long process that increases considerably for the following levels of subdivision. It is of the utmost importance that any efficient intersection algorithm should reduce the number of face intersection tests. To this end, we propose an intersection computation algorithm based on a bipartite graph [6]. The use of this graph allows us to organise the intersecting pairs of faces to accelerate the comparison steps. First intersection information are collected at the coarser level; second, these information are used to deduce intersection at finer levels for speeding up the process.

As the tools and computation methods used in this work are general, they can be applied to any subdivision surface scheme. We chose Loop subdivision surfaces in our implementation. Indeed, Loop subdivision surfaces are defined by triangular faces, which are widely used in geometric modeling due to the fact that they can be generated by modelers, 3D scanners or in any polygonal mesh triangulation. Moreover robust simplification algorithms exist for these meshes.

This paper is organized as follows. In section 2, we briefly introduce subdivision surfaces, in particular the Loop scheme, then boolean operations are described. Section 3 is devoted to the presentation of the proposed algorithm. Section 4 shows and discusses some experimental results.

2 Background

2.1 Subdivision Surfaces



Fig. 1. On the left, the initial face. On the right, the four sub-faces.

Subdivision surfaces are defined by a coarse mesh and a set of refinement rules. Applying these rules generates a sequence of finer and finer meshes $M_1, \ldots, M_i$ of control points which converges to a smooth surface M_∞ called *limit surface*. There are two kinds of subdivision schemes: *interpolating* and *approximating*. Loop scheme is an approximating scheme based on a face split and it produces a quadric box spline as a limit surface. This scheme is composed of two steps. First, every initial triangular face is split into four as shown in Figure 1.

Then, initial face vertices, called *even* vertices and new inserted vertices, called *odd* vertices are displaced according to the masks described in Figure 2 [2]. The (a) sub-

figure represents the interior even vertex mask where k denotes the vertex valence and β is chosen to be:

$$\beta = 3/16 \ if \ k = 3 \ and \ 1/k\left(5/8 - \left(3/8 + 1/4\cos\left(2\pi/k\right)\right)^2\right) \ if \ k > 3$$

The (b) sub-figure represents the crease and boundary even vertex mask. The (c) sub-figure illustrates the interior odd vertex mask. The (d) sub-figure shows the crease and boundary odd vertex mask.

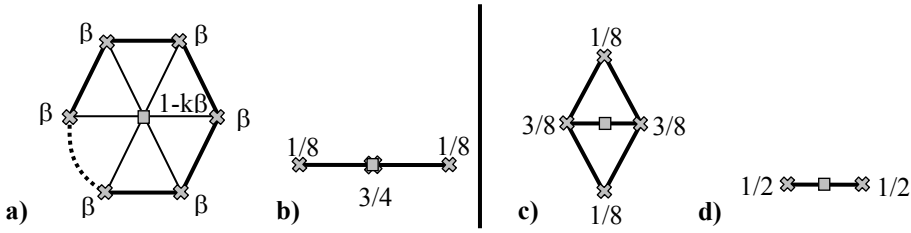


Fig. 2. Loop masks where ⊠ represent old vertices and □ the new position respectively of an *even* vertex (left: (a), (b)) and of an *odd* vertex (right: (c), (d)).

2.2 Boolean Operations

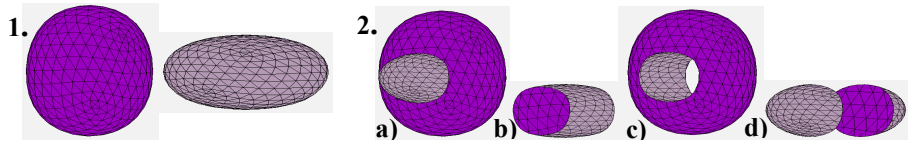


Fig. 3. Boolean operations on two simple objects (1) represented by two subdivision surfaces. 2. Left: Two objects to be combined. Right: a) Union. b) Intersection. c), d) Differences.

The increasing use of subdivision surfaces calls for the development of new, specific modeling tools or the adoption of existing tools associated with classical surfaces. The computation of boolean operations, for instance, is fundamental for the construction of complex objects from simpler ones. A CSG object is generated by the combination of several boolean operations (intersection, union, difference) on elementary primitives [9]. Primitives may be simple forms (cube, cylinder, torus...) or more complex forms built using a set of simpler primitives or generated by more complicated surfaces.

In a general way, a boolean operation proceeds in two steps:

- Computing intersection curves between the surfaces of the two primitives implied in the operation: some intersection points are found, classified and connected to approximate the intersection curve.
- Keeping the primitive parts necessary to construct the surface of the resulting object according to the considered boolean operation.

Computing intersections on subdivision surfaces helps for a better use of these surfaces in shape design. Efficient solutions are to be found for this crucial question.

3 The Intersection Algorithm

The objective of our algorithm is to reduce the number of intersection tests at an advanced level of subdivision. For two meshes $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, this algorithm can be decomposed into 4 steps: determining the pairs of faces in a position of intersection (f_j, g_k), $f_j \in \boldsymbol{S}_1$ and $g_k \in \boldsymbol{S}_2$, stocking the intersection information in a bipartite graph, recovering the neighborhood of each intersecting faces and then testing the intersection of subsets of faces connected by an edge in the graph.

3.1 Recovery of Intersecting Faces

The aim of this step is to classify the pairs of faces of the two subdivision surfaces into intersecting and non intersecting sets. At the initial level this is achieved with the accelerated naïve algorithm (see Figure 4). The naïve algorithm consists in testing the intersection between all pairs of faces. The degree of complexity of this naïve algorithm is high but at the initial level, the meshes of subdivision surfaces are relatively coarse. However it can be accelerated using a space partition and axes aligned bounding box tests. Let I_i be the face set of surface S_i involved in the intersection.

$I_i = \{F_k \mid F_k \in S_i \;\&\; F_k \cap S_{3-i} \neq \varnothing\}$, $i = 1$ or 2.

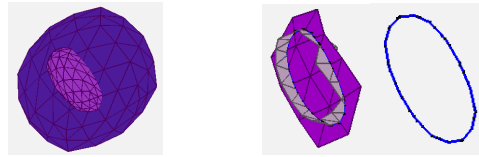


Fig. 4. Intersection at the initial level is computed using an accelerated naïve algorithm. It consists in computing the intersection between all pairs of faces formed by taking a face of each surface. Left: Intersection curve. Right: Intersecting faces.

3.2 Construction of the Bipartite Graph

Bipartite Graph. A bipartite graph G is given by $G(I_1, I_2, E)$ [6], where I_1 and I_2 are vertex sets and E is a edge set such that each edge of E has one extremity in I_1 and the other in I_2. In our case, we represent the intersecting faces by a bipartite graph as follows:

- Vertices of I_1 and I_2 represent respectively the intersecting faces of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ (a graph vertex is a face or a set of faces).
- The edge (i_1, i_2) where $i_1 \in I_1$ and $i_2 \in I_2$ are respectively a vertex of I_1 and I_2, means that i_1 intersects i_2.
- No edge can connect two vertices of the same vertex sets I_1 or I_2.

Example. Fig. 5 illustrates how to construct the graph in a straightforward example.

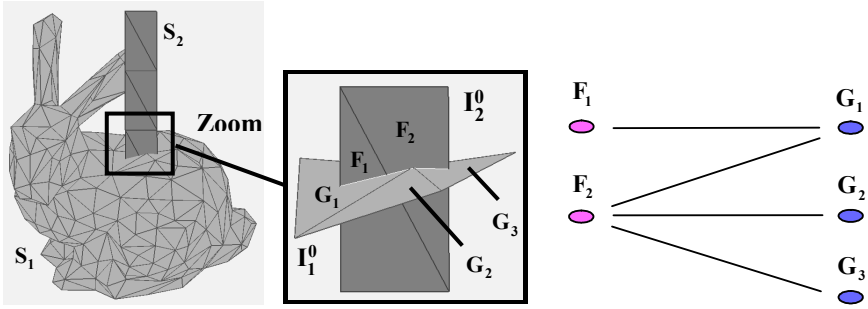


Fig. 5. Construction of the graph. Left: a zoom on the intersection area. The intersecting faces of the first surface are noted F_1 and F_2 and those of the second surface are noted G_1, G_2 and G_3. We only show the intersecting faces because the others are not linked by edges. Right: the bipartite graph of intersection. The nodes of the left column represent the faces of the first surface and those of the right column represent the faces of the second one. Intersecting faces are connected by edges.

3.3 Recovery of the Neighborhood

Neighborhoods of F_i and G_k are then added to the graph.

Neighborhood. Notions of 1-neighborhood and 2-neighborhood of a face F are specified in this section. They are noted $\mathcal{V}_1(F)$ and $\mathcal{V}_2(F)$ at the considered subdivision level and $\mathcal{W}_1(F)$ and $\mathcal{W}_2(F)$ at the next level (Figure 6). In the following, we note:

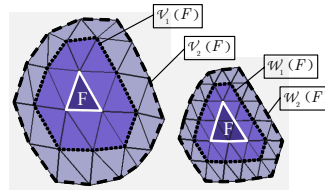


Fig. 6. The 1-neighborhood and the 2-neighborhood

$S(F)$: the list of vertices of a face (or a set of faces) F.

$Loop(F)$: the Loop subdivision of a face (or a set of faces) F.

The 1-neighborhood $\mathcal{V}_1(F)$ of a face F contains this face and all faces adjacent to face F by a vertex: $\mathcal{V}_1(F) = \{F\} \cup \{F_i \mid \exists s \in S(F) \cap S(F_i)\}$.

The 2-neighborhood $\mathcal{V}_2(F)$ of a face F is the 1-neighborhood of $\mathcal{V}_1(F)$:

$\mathcal{V}_2(F) = \{\mathcal{V}_1(F_i) \mid F_i \in \mathcal{V}_1(F)\}$.

The face set obtained by a subdivision of the 1-neighborhood of F is noted $\mathcal{W}_2(F)$: $\mathcal{W}_2(F) = \{F_i \mid F_i \in Loop(\mathcal{V}_1(F))\}$

$\mathcal{W}_1(F)$ is a face subset of $\mathcal{W}_2(F)$ restricted to the faces which have a common vertex with one of the sub-faces resulting from the subdivision of F :

$$\mathcal{W}_1(F) = \left\{ F_i \mid F_i \in Loop\left(\mathcal{V}_1(F)\right) \text{ and } \exists s \in S(Loop(F)) \cap S\left(F_i\right) \right\}$$

Example. Fig. 6 illustrates the terms of 1-neighborhood and 2-neighborhood of F. Left, the 1-neighborhood $\mathcal{V}_1(F)$ of a face F contains this face and all faces adjacent to face F by a vertex; it is delimited by dotted lines. The 2-neighborhood $\mathcal{V}_2(F)$ of a face F contains this face and all faces adjacent to $\mathcal{V}_1(F)$ by a vertex ($\mathcal{V}_2(F)$ is the 1- neighborhood of $\mathcal{V}_1(F)$), it is delimited by a line of dashes. Right, the face set $\mathcal{W}_2(F)$ obtained by a subdivision of the 1-neighborhood $\mathcal{V}_1(F)$ is delimited by a line of dashes. $\mathcal{W}_1(F)$ is a face subset of $\mathcal{W}_2(F)$ restricted to the faces which have a common vertex with one of the sub-faces resulting from the subdivision of F, it is delimited by dotted lines.

The bipartite graph of intersection is computed by this neighborhood information. Figure 7 shows the addition of the subdivided 1-neighborhood to all vertices for the graph of the example in Figure 5.

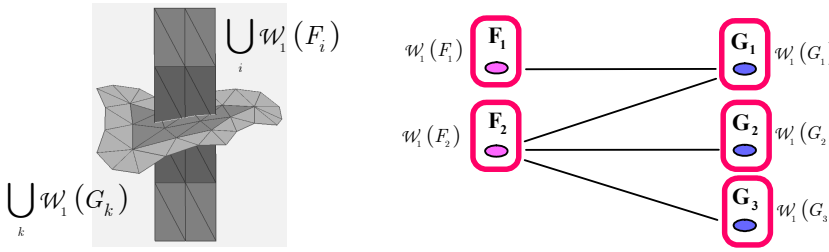


Fig. 7. The bipartite graph of Figure 5 right is now completed by adding the subdivided 1-neighborhood of all faces.

We choose to conserve only the subdivided 1-neighborhood because it fits the majority of cases. We have not yet succeeded in giving a mathematical condition which justifies this choice and imposes a larger neighborhood for particular cases.

3.4 Computation of Face/Face Intersection

The faces intersection computation is now reduced to a subset of pairs of faces. Indeed, the bipartite graph of intersection associates to each intersecting face F_i of I_1, a set G_i of I_2 intersecting faces, so that one needs only calculate the intersections between the elements of $\mathcal{W}_1\left(F_i\right)$ and the elements of $\mathcal{W}_1\left(G_k\right)$ connected by an edge.

To summarize, at each level the input of this algorithm is a bipartite graph $G^0\left(I_1^0, I_2^0, E^0\right)$ where I_1^0 and I_2^0 are the vertex sets of the graph (intersecting faces) and E^0 the edge set (intersecting pairs of faces). The output is the intersection curves with intersecting faces. So the process can easily be iterated because knowing these

faces allows to construct a new bipartite graph $G^1\left(I_1^1, I_2^1, E^1\right)$ where I_1^1 and I_2^1 are the vertex sets (intersecting faces of $\mathcal{W}_1\left(F_i\right)$ and $\mathcal{W}_1\left(G_k\right)$) and E^1 the edge set (intersecting pairs of faces between these two parts).

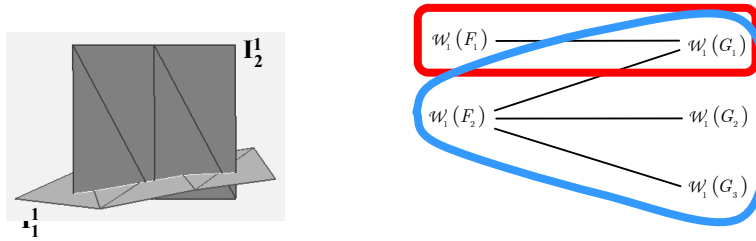


Fig. 8. The intersection (example of Figure 5) is computed between the faces of the sets $\mathcal{W}_1\left(F_1\right)$ and $\mathcal{W}_1\left(G_1\right)$ because they are connected by an edge. Moreover intersections of sets $\mathcal{W}_1\left(F_2\right)$ and $\mathcal{W}_1\left(G_1\right)$, $\mathcal{W}_1\left(G_2\right)$, $\mathcal{W}_1\left(G_3\right)$ are also computed.

The algorithm operates in 4 consecutive steps in the following way:

1. A naive algorithm is used to determine the pairs of intersecting faces between two initial surfaces.
2. From these faces, the bipartite graph is built.
3. For each node of the graph, the 1-neighborhoods of faces F_i of the first object and faces G_k of the second object are computed; they are respectively referred to as $\mathcal{V}_1\left(F_i\right)$ and $\mathcal{V}_1\left(G_k\right)$. $\mathcal{V}_1\left(F_i\right)$, $F_i \in I_1$ and $\mathcal{V}_1\left(G_k\right)$, $G_k \in I_2$ are then partially subdivided so as to preserve only the 1-neighborhoods of the subdivision of F_i and G_k. These 1-neighborhoods are designated by $\mathcal{W}_1\left(F_i\right)$ and $\mathcal{W}_1\left(G_k\right)$.
4. The naïve algorithm is again used to obtain intersection curves and intersecting faces between elements of $\mathcal{W}_1\left(F_i\right)$ and $\mathcal{W}_1\left(G_k\right)$ connected by an edge.

4 Experimental Results

4.1 Case 1: Normal Intersections

This example corresponds to a current case of intersection. Figure 9 shows two surfaces in a position of intersection (left) and the first step of our algorithm (right) i.e. intersection curves and intersecting faces obtained by applying the accelerated naïve algorithm. Figure 10 illustrates the following steps obtained using only the naïve algorithm, a neighborhood algorithm similar to O'Brien's [12] or the bipartite graph algorithm. At each level of subdivision, O'Brien only keeps the intersecting faces and

their neighborhood. Notice that the quality is the same with the three algorithms but the complexity of our algorithm is much lower than the two others (Table 1). Time of computation is also better for our algorithm, but time reduction is less interesting than the reduction in the number of tests, this is due to the management of the graph.

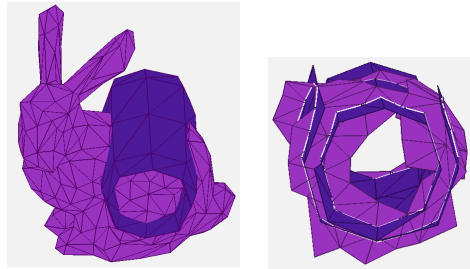


Fig. 9. Left, our algorithm is applied on the bunny (694 faces) and a tube (128 faces). Right, the intersection curves are determined at the initial level by the application of the accelerated naïve algorithm.

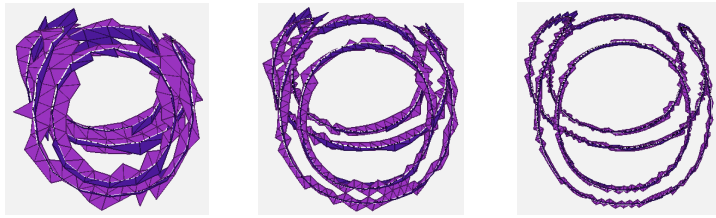
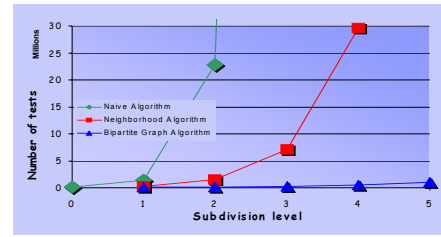


Fig. 10. The computation of the intersection curves with our algorithm at the next subdivision levels is represented from left to right.

Table 1. Qualitative results. Comparison of the results obtained in applying the naïve algorithm, the "Neighbor" algorithm and the "Graph" algorithm on the example of the Figure 9. We compare respectively the number of intersection tests and the computing time in seconds for each algorithm.

	Naive		Neighbor		Graph	
	Tests	Time	Tes	Time	Tests	Time
	9 E4	0,15				
1	1 E6	5	2 E5	1	8 E4	1
2	2 E7	114	1 E6	7	1 E5	5
3	4 E8	2175	7 E6	41	2 E5	31
4			3 E7	260	5 E5	214



4.2 Case 2: Wrong Intersections

Operations of subdivision can change the intersection status of two meshes from one subdivision level to another: two non intersecting meshes can become intersecting after one or two step of subdivision, on the other hand two intersecting meshes can

become non intersecting after subdivision. These cases are referred to as wrong intersections. Figure 11 shows an example of wrong intersection.

Our algorithm works well on these cases as shown in Figures 11 and 12.

Let us consider the two meshes of Figure 11 (left), and analyse their intersection where the egg-like mesh is put inside the pyramid-like one. At the first level, there is no intersection because the egg is completely contained inside the pyramid without any overlapping between the faces of both surfaces. At the second level of subdivision (Figure 11, right), the pyramidal polygon is flattered enough to overlap with the subdivided egg.

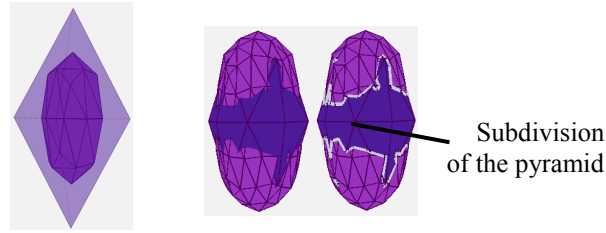


Fig. 11. An example of "wrong" intersection. The two initial meshes do not intersect and the refined meshes intersect. At the initial level, one mesh is inside the other, so there is not intersection and after one step of subdivision the two meshes intersect. As there are no intersections at level k, the naïve algorithm is applied at level k+1 ensuring to test all possible intersecting faces.

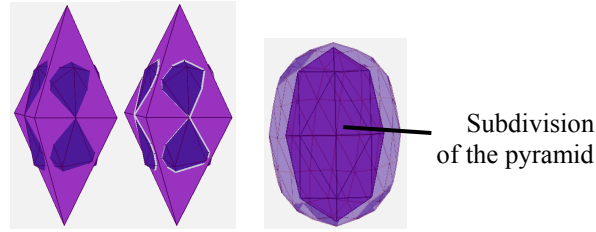


Fig. 12. An other example of "wrong" intersection. The two initial meshes intersect at the initial level and the refined meshes do not intersect. The pyramid-like mesh is inside the egg-like mesh.

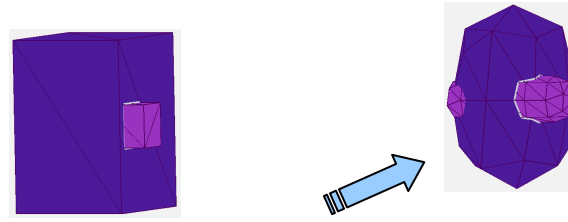


Fig. 13. At the initial level of subdivision, there is only one intersection but at the next level there are several intersections and for instance the intersection pointed to by the arrow is not detected.

4.3 Example 3

Some more complicated cases of wrong intersections are still not covered by our algorithm, Figure 13 shows an example of such cases where our algorithm is not able to detect all intersections.

5 Conclusion

In this paper, we described a robust and efficient algorithm for the computation of intersection curves between two objects modeled by subdivision surfaces. The algorithm is structured in two independent and complementary modules; the intersection module and subdivision module. The first one is dedicated to computations related to the intersection (faces bounding boxes, intersections), the role of the second module is the management of subdivision operations, computation of faces neighborhood and construction of the bipartite graph. Subdivision scheme and intersection algorithm are independent from one another, it is then possible to switch to another subdivision scheme without modifying the intersection computation. The integration of this algorithm into boolean operation framework is the next step of our work.

References

1 Abdel-Malek, K., and Yeh, H. J. "Determining Intersection Curves between Surfaces of Two Solids." *Computer Aided Design*, 28, pp. 539-549, 1996.
2 Bajaj, C. L., Hoffmann, C. M., Hopcroft, J. E., and Lynch, R. E. "Tracing Surface Intersections." *Computer Aided Geometric Design*, 5, pp. 285-307, 1988.
3 Barnhill, R. E., Farin, G., Jordan, M., and Piper, B. R. "Surface / Surface Intersection." *Computer Aided Geometric Design*, 4, pp. 3-16, 1987.
4 Biermann, H., Kristjansson, D., and Zorin, D. "Approximate Boolean Operations on Free-Form Solids." in *CAGD 2000*, Oslo, Norway, 2000.
5 Boender, E. "A Survey of Intersection Algorithms for Curved Surfaces." *Computer & Graphics*, 15, pp. 99-115, 1991.
6 Bondy, A., and Murty, U. S. R. (1976), *Graph Theory with Apllications*, ed. Inc. American Elsevier Publishing Co., New York.
7 Chandru, V., and Kochar, B. S., "Geometric Modeling: Algorithms and New Trends." in *Chapter Analytic Techniques for Geometric Intersection Problems*, PA, Philadelphia: SIAM, pp. 305-318,1987.
8 DeRose, T., Kaas, M., and Truong, T. "Subdivision Surfaces in Character Animation." in *SIGGRAPH Proceedings*, pp. 85 - 94, 1998.
9 Krishnan, S., Narkhede, A., and Manocha, D. "Boole: A System to Compute Boolean Combinations of Sculptured Solids." Technical Report, Department of Computer Science, University of North California, 1994.
10 Linsen, L. "Netbased Modelling." in *SCCG 2000*, Slovakia, pp. 259-266, 2000.
11 Litke, N., Levin, A., and Schrđer, P. "Trimming for Subdivision Surfaces." Technical Report, Caltech, 2000.
12 O'Brien, D. A., and Manocha, D. "Calculating Intersection Curve Approximations for Subdivision Surfaces." 2000. http://www.cs.unc.edu/õbrien/courses/comp258/project.html.
13 Patrikalakis, N. M. "Surface-to-Surface Intersections." *IEEE Computer Graphics & Applications*, 13, pp. 89-95, 1993.

Implicit Polynomial Support Optimized for Sparseness*

Ioannis Z. Emiris[1] and Ilias S. Kotsireas[2]

[1] Department of Informatics & Telecommunications, National University of Athens, Panepistimiopolis, 15771 Greece, emiris@di.uoa.gr, and INRIA Sophia-Antipolis, France

[2] Department of Computing, Wilfrid Laurier University, 75, University Avenue West, Waterloo N2L 3C5, ON, Canada, ikotsire@wlu.ca, and Ontario Research Centre for Computer Algebra, University of Western Ontario, London N6A 5B7, ON, Canada, Ilias.Kotsireas@orcca.on.ca

Abstract. We propose the use of various tools from algebraic geometry, with an emphasis on toric (or sparse) elimination theory, in order to predict the support of the implicit equation of a parametric hypersurface. The problem of implicitization lies at the heart of several algorithms in geometric modeling and computer-aided design, two of which (based on interpolation) are immediately improved by our contribution. We believe that other methods of implicitization shall be able to benefit from our work. More specifically, we use information on the support of the toric resultant, and degree bounds, formulated in terms of the mixed volume of Newton polytopes. The computed support of the implicit equation depends on the sparseness of the parametric expressions and is much tighter than the one predicted by degree arguments. Our Maple implementation illustrates many cases in which we obtain the exact support. In addition, it is possible to specify certain coefficients of the implicit equation.

1 Introduction

In this paper we apply several tools from algebraic geometry, with an emphasis on toric (or sparse) elimination theory, in order to predict the support of the implicit equation of a parametric hypersurface. The problem of switching from a rational parametric representation to an implicit, or algebraic, representation of a curve, surface, or hypersurface lies at the heart of several algorithms in computer-aided design and geometric modeling, cf. e.g. [1, 3, 6, 11, 16, 18].

* This work was initiated when the first author was visiting the Ontario Research Centre for Computer Algebra, London, Canada. The first author is supported partially by FET Open European Project IST-2001-35512 (GAIA-II), and Project 70/4/6452 of the Research Committee of the National University of Athens, Greece. The second author is partially supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 397–406, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In particular we use, on the one hand, bounds on the total degree of the implicit expression, as well as bounds on its degree in each variable. For tightness, we formulate these bounds in terms of Newton polytopes and mixed volumes, which exploit the structure in the parametric expressions and depend only on their nonzero terms. On the other hand, we exploit information on the support of the toric (or sparse) resultant by considering the extreme monomials as described in [15, 22]. The output support scales with the sparseness of the parametric polynomials and is much tighter than the one predicted by degree arguments. In many cases, we obtain the exact support of the implicit equation, as seen by applying our Maple program. Moreover, it is possible to specify certain coefficients in this equation.

Our motivation comes mainly from two implicitization algorithms based on interpolation. The first one (see [6]) treats parametric families of curves, surfaces and hypersurfaces. The method has a very wide range of applicability, can handle base points, and works both symbolically and numerically, depending on the way one performs the integrations; cf. section 6 for an illustration. It may be improved as follows: The method looks for an implicit equation of a particular degree at a time. This implies that any information on the degree of the implicit equation (such as upper bounds) may accelerate execution. More importantly, the method constructs a symmetric singular square matrix and computes a basis of its nullspace. The dimension of this matrix equals the number of possible monomials in the implicit equation, which is in principle $\binom{m+n}{m}$, where the number of parametric equations is n and the algorithm seeks an implicit equation of degree m. The examples show (cf. the table below) that we succeed to constrain the monomials that will appear in the implicit equation, hence diminishing dramatically the size of the matrices. One last improvement concerns the block-Hankel structure of the matrix, but this goes beyond the scope of the current paper. Hankel-like structural properties of implicitization matrices are established in [17].

Our second motivation are algorithms based on perturbed resultant matrices, which yield the implicit equation even in the presence of base points, e.g. [9, 18]. The problem reduces to sparse interpolation, which is substantially accelerated when we can accurately predict the output support. The table below shows the results of the algorithm on the examples of section 7. We refer to our method as IPSOS.

Problem	**Input Degree**	**Degree of Implicit Eq.**	**General # monomials**	**# monomials from IPSOS**
Unit Circle	2	2	6	3 (optimal)
Descartes Folium	3	3	10	3 (optimal)
Buchberger	1,2	4	35	2 (optimal)
Busé	3	5	56	4 (optimal)
Bilinear	1,1	2	10	9 (optimal)

This paper is structured as follows: In section 2 we introduce toric elimination for predicting the support of the implicit by exploiting the input sparseness. In

section 3 we summarize algebraic geometry arguments that allow us to predict the degree of the implicit equation. We may extend these arguments to bound the degrees of the implicit equation in each variable separately. Section 4 describes the algorithm and applies it to parametric families of (hyper)surfaces. In section 5 we sketch our Maple implementation and its interoperability with other public-domain C/C++ stand-alone programs that we employ. Section 6 presents examples of implicitization of curves and surfaces; the computed support turns out to be optimal in all cases thanks to their sparseness. Finally, section 7 provides ideas for future work.

A preliminary version of our results has appeared as [13].

2 Toric Elimination Theory

This section overviews our algebraic tools, from the theory of toric (or sparse) elimination. For details see [8].

Let $A_i \subset \mathbb{Z}^n$ and consider polynomials $f_i(x) = \sum_{a \in A_i} c_{ia}\ x^a$, in n variables $x = (x_1, \ldots, x_n)$, with indeterminates $c_{ia} \neq 0$. Then A_i is the *support* of polynomial f_i and its *Newton polytope* $Q_i \subset \mathbb{R}^n$ is the convex hull of A_i.

Theorem 1. *The mixed volume $MV(Q_1, \ldots, Q_n)$ of the $Q_i \subset \mathbb{R}^n$ corresponding to polynomials $f_1, \ldots, f_n \in K[x^{\pm 1}]$ bounds the degree of the toric variety of the well-constrained system of polynomials $f_1, \ldots, f_n$.*

Mixed volume generalizes Bézout's classical bound on the number of common roots for a well-constrained system in the sense that it reduces to Bézout's bound for dense polynomials but is in general tighter. Consider an *over-constrained* system of polynomials $f_0, \ldots, f_n \in K[x^{\pm 1}]$, with supports $A_0, \ldots, A_n \subset \mathbb{Z}^n$.

Definition 1. *The* toric (or sparse) resultant $R(f_0, \ldots, f_n)$ *is a polynomial in $\mathbb{Z}[c_{ia}]$, homogeneous in the coefficients of each f_i, with degree equal to $MV_{-i} := MV(Q_0, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n)$. The resultant vanishes after a specialization of the coefficients iff the specialized system of $f_0, \ldots, f_n$ has a solution in the toric variety associated to the Q_i.*

Certain works, including [15, 22], have studied the Newton polytope of R. In particular, [15] describes the hyperplanes defining the facets of the Newton polytope of R. But this study is not extended to toric resultants in general dimension over arbitrary input supports.

Instead, we shall compute the vertices of this Newton polytope, i.e. the *extreme monomials* in R, cf. [22] and [15, Sect. 8.3]. Consider any collection $\{A_i\}_{i \in I}$ of supports, $I \subset \{0, \ldots, n\}$. Its rank, denoted $\mathrm{rk}(I)$, is the rank of the affine lattice generated by the $\sum_{i \in I} A_i$. A collection $\{A_i\}_{i \in I}$ is *essential* iff $\mathrm{rk}(I) = |I| - 1$ and $\mathrm{rk}(J) \geq |J|$ for every proper subset $J \subset I$.

The initial form $init_\omega(f)$ of a multivariate polynomial f in k variables, with respect to some functional $\omega : \mathbb{Z}^k \to \mathbb{R}$, is the sum of all terms in f which maximize the inner product of ω by the corresponding exponent vector. When

$k = |A_0| + \cdots + |A_n|$, then ω defines a *lifting* function on the input system, by lifting every support point $a \in A_i$ to $(a, \omega(a)) \in \mathbb{Z}^n \times \mathbb{R}$. The lifted supports $\widehat{Q}_i \subset \mathbb{R}^{n+1}$ have Minkowski sum $\widehat{Q} = \widehat{Q}_1 + \cdots + \widehat{Q}_n$. Its lower hull projects bijectively, along the last coordinate, to $Q = Q_1 + \cdots + Q_n \subset \mathbb{R}^n$. The lower hull facets then correspond to maximal *cells* of an induced *coherent mixed decomposition* of Q.

If ω is sufficiently generic, then this decomposition is *tight*; in the sequel, we assume our mixed decompositions are both coherent and tight and denote it by Δ_ω. Then, maximal cells of the form $F = F_0 + \cdots + F_n$, where $\dim F_j = 1$ for all $j \in \{0, \ldots, n\}$ except for one value, are called *mixed*. It is clear that the $(n+1)$st summand must have dimension 0; if this is the i-th summand, then the cell is said to be of type i or i-mixed. The sum of volumes of all i-mixed cells equals the partial mixed volume MV_{-i}, for any $i \in \{0, \ldots, n\}$. The corresponding coefficient in f_i is denoted by c_{iF_i}; its monomial is x^{F_i} and $F_i \in A_i$.

Theorem 2 ([22]). *Suppose that $\{A_0, \ldots, A_n\}$ is essential. Then the initial form of the toric resultant R with respect to a generic ω equals the monomial $init_\omega(R) = \prod_{i=0}^n \prod_F c_{iF_i}^{vol(F)}$, where $vol(\cdot)$ denotes ordinary Euclidean volume and the second product is over all mixed cells of type i of the tight mixed coherent decomposition Δ_ω.*

It is clear that a bijective correspondence exists between the extreme monomials and the configurations of the mixed cells of the A_i. So, it suffices to compute all distinct mixed-cell configurations, as discussed in [19, 20].

Another (simpler) means of reducing the number of relevant mixed decompositions is by restricting attention to those with a specific number of cells. This number is usually straightforward to compute in small dimensions (e.g. when $n = 2, 3$) and reduces drastically the set of mixed decompositions. For instance, when studying the implicitization of a biquadratic surface, the total number of mixed decompositions is 19728, whereas those with 8 cells is 62.

In certain special cases, we can be more specific about the Newton polytope of the toric resultant. First, its dimension equals $k-2n-1$ [15, 22]. Certain corollaries follow: For essential support families, a 1-dimensional Newton polytope of R is possible iff all polynomials are binomials. The only resultant polytope of dimension 2 is the triangle; in this case the support cardinalities must be 2 and 3. For dimension 3, the possible polytopes are the tetrahedron, the square-based pyramid, and polytope $N_{2,2}$ given in [22]; the support cardinalities are respectively $2, 2$ and 3.

It is known that the coefficients of all extreme monomials are in $\{-1, 1\}$ [14, 4, 22]. Sturmfels [22] also specifies, for the extreme monomials, a way to compute the precise coefficients. But this requires computing several coherent mixed decompositions, and goes beyond the scope of the present paper.

The so-called *Cayley trick* introduces a new point set $C := \{(z, a_{0j}, 1) \,:\, a_{0j} \in A_0\} \cup \{(e_i, a_{ij}, 1) \,:\, i = 1, \ldots, n,\ a_{ij} \in A_i\} \subset \mathbb{Z}^{2n+1}$, where $z = (0, \ldots, 0) \in \mathbb{N}^n$ is the zero vector and $e_i = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{N}^n$ has a unit at the i-th position and $n-1$ zeroes.

Theorem 3. [15, 20, 22] *The problem of computing all mixed decompositions of* $A_0, \ldots, A_n \subset \mathbb{Z}^n$ *is equivalent to computing all regular triangulations of the set* C *defined above, which contains* $k_0 + \cdots + k_n$ *points, where* $k_i = |A_i|$.

Efficient algorithms (and implementations) exist for computing all regular triangulations of a point set [21]. Regular are those triangulations that can be obtained by projection of a lifted triangulation.

3 Degree Bounds for the Implicit Equation

In this section we sketch some techniques to bound the total degree of the implicit equation, given the rational parametric equations. In addition, we adapt the approach to bound the degree in each variable separately. Current work for formalizing the rough account below is beyond the scope of this paper.

For the sake of simplicity, we shall describe our arguments in affine space instead of projective space, but this is no lack of generality. We motivate the discussion using a plane curve:

$$x_0 = P_0(t_1)/Q(t_1), \quad x_1 = P_1(t_1)/Q(t_1) \tag{1}$$

where $P_0(t_1), P_1(t_1), Q(t_1)$ are univariate polynomials in t_1. When we intersect a plane curve with a generic straight line, we obtain generically a number of points equal to the total degree of the implicit equation. The term "generic" here means that the line is sufficiently random, in particular, it cannot be tangent to the curve. Then, we substitute x_0 and x_1 by (1), clear out denominators and compute the degree of the resulting equation in t_1. This will be the total degree of the implicit equation.

In general, we intersect the parametric hypersurface with n generic linear equations in $x_0, \ldots, x_n$, where n denotes the number of parameters. The parametric expressions are substituted in these n equations to yield a non-linear system of n polynomials in $t_1, \ldots, t_n$. The degree of the (toric) variety defined by this system is bounded by the corresponding mixed volume, which thus bounds the total degree of the implicit equation. Remark that the support of each polynomial in t is the union of the supports of the $x_i Q(t) - P_i(t)$, seen as polynomials in t. Therefore all equations have the same Newton polytope, hence the mixed volume equals $n!$ multiplied by the volume of this Newton polytope.

To compute the degree of the implicit equation in x_0, intersect the curve with a straight line $x_1 = K_1$, where K_1 is a generic constant. Then, x_1 is substituted by its parametric expression in (1). The degree of the resulting equation in t_1 bounds the degree of the implicit equation in x_0. In general, to bound the degree of the implicit equation in some x_j, $j \in \{0, \ldots, n\}$, intersect the implicit equation by the equations $x_i = K_i$ for $j \neq i \in \{0, \ldots, n\}$ and replace all x_i by their parametric expressions. The resulting system of n equations in $t_1, \ldots, t_n$ is well-constrained, and its mixed volume bounds the implicit degree in x_i. This is simply the mixed volume of the polynomials $x_i Q(t) - P_i(t)$, $i \neq j$, seen as polynomials in the parameters t.

4 The IPSOS Algorithm

We describe the overall algorithm which estimates the support of the implicit equation of a rational parametric (hyper)surface. Given the parametric expressions $x_i = P_i(t)/Q(t)$, for $i = 0, \ldots, n$, we regard them as polynomials $f_i = x_i Q(t) - P_i(t)$ in the parameters $t = (t_1, \ldots, t_n)$. Then, the implicitization problem is equivalent to eliminating the parameters t; the implicit equation equals the resultant of the f_i, provided there are no base point and that the parametrization is one-to-one (i.e. proper). If the latter condition is violated, then the resultant gives us a multiple of the implicit equation. For simplicity, we may assume the given parametrization is one-to-one.

Our toric elimination tools shall be applied to the polynomials f_i, where we ignore the specific values of the coefficients. This is an interesting feature of the algorithm, namely that it considers the monomials in the parametric equations but not their actual coefficients. This shows that the algorithm is suitable for use as a preprocessing off-line step in CAGD computations, where one needs to compute thousands of examples with the same support structure in real time. This handles the implicitization of (multiparametric) families of (hyper)surfaces, indexed by one or more parameters. This technique is called *generic implicitization*.

Of course, the generic resultant coefficients are eventually specialized to functions of the x_i. Then, any bounds on the implicit degree in the x_i are applied, in order to reduce the support set which is output.

`INPUT:` The rational parametric equations of a hypersurface $x_0 = P_0(t)/Q(t), \ldots, x_n = P_n(t)/Q(t)$, where $t = (t_1, \ldots, t_n)$, and $\gcd(P_i(t), Q(t)) = 1$ for $i = 0, \ldots, n$.

`OUTPUT:` A superset of the monomials in the support of the implicit equation of the input, and the coefficients of certain (at least one and at best all) of the corresponding terms.

1. Define the polynomials $f_i = x_i Q(t) - P_i(t), i = 0, \ldots, n$ and look at them as polynomials in t: $f_i = \sum_{\alpha_{ij} \in A_i} c_{ij} t^{\alpha_{ij}}$, where $A_i \subset \mathbb{Z}^n$.
2. Apply the Cayley trick and compute all regular triangulations of the corresponding point set C, which yield all mixed subdivisions of $A_0 + \cdots + A_n$.
3. Obtain the extreme monomials of the Newton polytope of the resultant from the mixed subdivisions. Then compute a superset of the support of the resultant.
4. Transform the support, which is a set of monomials of the form $\prod c_{ij}^{e_{ij}}$, to a set of monomials in the $x_0, \ldots, x_n$.
5. Use the implicit degree bound to eliminate any of the monomials that cannot appear in the implicit equation, i.e. whose degree is higher than the total implicit degree.

Step 2 yields as by-product all partial mixed volumes MV_{-i} for $i = 0, \ldots, n$, and hence the implicit degree separately in the x_i variables.

Step 3 may be analyzed into certain substeps, in different ways. Given the vertices of a polytope in dimension $k = k_0 + \cdots + k_n$, there are algorithms for computing all integer points in its interior. Some efficient implementations though require that the polytope be described in terms of its facets, so we may have to produce these facets from the set of extreme points.

Notice that in the last step, we may also use implicit degree bounds in each separate variable x_i. But these bounds should already be taken into account when the algorithm computes the toric resultant support. The latter claim follows from toric resultant theory and the homogeneities of the toric resultant as a polynomial in the coefficients of the f_i.

5 Implementation of the Algorithm

A preliminary implementation of the algorithm in Maple 8 is available upon request from the authors. The name of the package is IPSOS. Besides Maple 8 functions, it makes use of certain Linux/Unix commands as well as publicly available software for Linux/Unix. The following programs were actually used during the development stages of the algorithm and the experiments.

• The C Program PORTA [5] is a collection of routines for analyzing polytopes, switching between their facet and vertex presentations, as well as for enumerating all integral points inside a polytope (though the latter does not seem to be a fast algorithm).
• The Maple V program PUNTOS [10] allows us to compute regular triangulations of point sets. Its applicability is limited by reasons of efficiency as well as by the fact that it cannot handle well very small examples. Its advantage is its portability since it is developed in Maple. A more efficient alternative is TOPCOM.
• The C++ program TOPCOM version 0.11.1 [21] computes all regular triangulations of a point set. An important feature is the possibility to specify the number of simplices in the triangulations of interest. We also experimented with symmetries, which do not seem to accelerate execution.
• The C program Mixvol, which is an implementation of the incremental mixed volume algorithm from [12].

Current work is improving the software as follows: First, recall that a bijective correspondence exists between the extreme monomials and the mixed-cell configurations, which is studied in [20]. Unfortunately, we were unable to find an implementation of this work, hence it is not yet used in our software. Another possibility of improvement is in enumerating integral points inside polytopes of any dimension. For this, we are experimenting with different software.

6 Examples

6.1 Unit Circle

Suppose that we are given the rational parameterization $x = (t^2-1)/(t^2+1)$, $y = 2t/(t^2+1)$. First we write the equations as polynomials in t. The Cayley trick yields C as the columns of the matrix $\begin{pmatrix} 0\,0\,1\,1\,1 \\ 0\,2\,0\,1\,2 \\ 1\,1\,1\,1\,1 \end{pmatrix}$. The corresponding regular triangulations are $\{\{1,2,3\}, \{2,3,5\}\}$, $\{\{1,2,5\}, \{1,3,5\}\}$, $\{\{1,2,3\}, \{2,3,4\}, \{2,4,5\}\}$, $\{\{1,3,4\}, \{1,2,4\}, \{2,4,5\}\}$, $\{\{1,3,4\}, \{1,4,5\}, \{1,2,5\}\}$. They yield the candidate monomials $\{1, y^2, x^2, y^2\,x, y^2\,x^2\}$. Using the degree bound $d = 2$ we obtain the monomials $\{1, y^2, x^2\}$. This support is exact. The implicitization method in [6] employs, in principle, a 6×6 symmetric singular matrix. Using the information from our algorithm, we reduce it to the matrix $M = \begin{bmatrix} 1 & x^2 & y^2 \\ x^2 & x^4 & x^2y^2 \\ y^2 & x^2y^2 & y^4 \end{bmatrix}$. We finally obtain the implicit equation $-1+x^2+y^2 = 0$.

6.2 Folium of Descartes

Consider $x = 3t^2/(t^3+1)$, $y = 3t/(t^3+1)$ [7]. The candidate monomials are $\{y^3, x^3, x^3\,y^3, x\,y, y^2\,x^2\}$. After applying the degree bound $d = 3$ we obtain the monomials $\{y^3, x^3, x\,y\}$, which is optimal. Using the implicitization method of [6], this example requires, in principle, a 10×10 matrix. Our algorithm yields $\begin{bmatrix} x^2y^2 & x^4y & xy^4 \\ x^4y & x^6 & x^3y^3 \\ xy^4 & x^3y^3 & y^6 \end{bmatrix}$ and arrives at the implicit equation $x^3 + y^3 - 3\,x\,y = 0$.

6.3 Example from [2]

The parametrization is $x = rt$, $y = rt^2$, $z = r^2$. The implicitization of [6] employs, *a priori*, a 35×35 matrix. Using the support information we can use a 2×2 matrix. The support computed by IPSOS is optimal, even without applying any degree bound. The implicit equation is $x^4 - y^2z = 0$.

6.4 Example from [3]

The parametrization is $x = s^2/(s^3+t^3)$, $y = s^3/(s^3+t^3)$, $z = t^2/(s^3+t^3)$. The Cayley trick yields the point set C corresponding to the columns of the matrix below, where the j-th column corresponds to the j-th monomial in the following sequence of 8 monomials from the 3 input polynomials:

$$\begin{bmatrix} 0\,0\,0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,1\,1 \\ 0\,2\,3\,0\,3\,0\,0\,3 \\ 3\,0\,0\,3\,0\,2\,3\,0 \\ 1\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix}, \qquad [[-x,1,-x],[-y,1-y],[1,-z,-z]]\,.$$

Our algorithm finally yields the monomials $\{x^3y, x^3, x^3y^2, y^2z^3\}$. To finish the implicitization by the method of [6] we construct the 4×4 matrix

$$\begin{bmatrix} x^6y^2 & x^6y & x^6y^3 & x^3y^3z^3 \\ x^6y & x^6 & x^6y^2 & x^3y^2z^3 \\ x^6y^3 & x^6y^2 & x^6y^4 & x^3y^4z^3 \\ x^3y^3z^3 & x^3y^2z^3 & x^3y^4z^3 & y^4z^6 \end{bmatrix}.$$

After substitution and integration, the matrix has a nullspace spanned by $[-2, 1, 1, -1]$. This gives the implicit equation: $-2\,x^3y + x^3 + x^3y^2 - y^2z^3 = 0$.

7 Future Work

A well-known *bicubic surface* example (whose implicit equation is computed with the special method [16]) represents a significant challenge for IPSOS:

$x = 3\ t\ (t-1)^2 + (s-1)^3 + 3\ s$,

$y = 3\ s\ (s-1)^2 + t^3 + 3\ t$,

$z = -3s(s^2-5s+5)t^3 - 3(s^3+6s^2-9s+1)t^2 + t(6s^3+9s^2-18s+3) - 3s(s-1)$.

After the Cayley trick, we obtain the following point set $C \subset \mathbb{Z}^5$:

```
[0, 0, 0, 0, 1], [0, 0, 0, 1, 1], [0, 0, 0, 2, 1], [0, 0, 0, 3, 1],
[0, 0, 1, 0, 1], [0, 0, 2, 0, 1], [0, 0, 3, 0, 1], [1, 0, 0, 0, 1],
[1, 0, 0, 1, 1], [1, 0, 0, 3, 1], [1, 0, 1, 0, 1], [1, 0, 2, 0, 1],
[1, 0, 3, 0, 1], [0, 1, 0, 0, 1], [0, 1, 0, 1, 1], [0, 1, 0, 2, 1],
[0, 1, 1, 0, 1], [0, 1, 1, 1, 1], [0, 1, 1, 2, 1], [0, 1, 1, 3, 1],
[0, 1, 2, 0, 1], [0, 1, 2, 1, 1], [0, 1, 2, 2, 1], [0, 1, 2, 3, 1],
[0, 1, 3, 1, 1], [0, 1, 3, 2, 1], [0, 1, 3, 3, 1].
```

There are 737129 regular triangulations (by TOPCOM). The last one is:

```
{2,3,4,7,13},{3,4,5,7,13},{3,5,6,7,13},{3,6,9,13,14},{6,9,12,13,14},
{3,6,9,14,15},{6,9,12,14,15},{6,12,13,14,16},{6,12,14,15,16},
{6,12,15,16,17},{3,6,9,15,18},{6,9,12,15,18},{6,12,15,17,18},
{3,9,15,18,19},{3,6,9,18,19},{6,9,12,18,19},{6,12,16,17,20},
{6,12,17,18,20},{3,6,9,19,23},{6,9,12,19,23},{6,12,19,22,23},
{6,12,22,23,24},{6,12,23,24,25},{3,6,9,23,26},{6,9,12,23,26},
{6,12,23,25,26},{0,2,4,7,13},{3,6,7,9,13},{6,12,18,19,22},
{6,12,18,20,24},{6,7,9,12,13},{6,12,18,22,24}.
```

The size of the file is 383 MBytes. This underlines the fact that we should not compute all regular triangulations but only the mixed-cell configurations.

References

[1] C.L. Bajaj and S. Evans. Splines and geometric modeling. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 833–850. CRC Press, Boca Raton, Florida, 1997.

[2] B. Buchberger. Applications of Gröbner bases in non-linear computational geometry. In J.R. Rice, editor, *Mathematical Aspects of Scientific Software*, volume 14 of *IMA Volumes in Math. & Appl.*, pages 59–87. Springer, 1988.

[3] L. Busé. Residual resultant over the projective plane and the implicitization problem. In *ISSAC (London ON, Canada)*, pages 48–55. ACM, 2001.

[4] J.F. Canny and I.Z. Emiris. A subdivision-based algorithm for the sparse resultant. *J. ACM*, 47(3):417–451, May 2000.

[5] T. Christof and A. Löbel. PORTA, 1997. www.zib.de/Optimization/Software/.

[6] R.M. Corless, M.W. Giesbrecht, I.S. Kotsireas, and S.M. Watt. Numerical implicitization of parametric hypersurfaces with linear algebra. In *Artificial intelligence and symbolic computation (Madrid, 2000)*, pages 174–183. Springer, Berlin, 2001.

[7] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms*. Springer-Verlag, New York, second edition, 1997. An introduction to computational algebraic geometry and commutative algebra.

[8] D. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*. Springer-Verlag, New York, 1998.

[9] C. D'Andrea and I.Z. Emiris. Computing sparse projection operators. In *Symbolic computation: solving equations in algebra, geometry, and engineering (South Hadley, MA, 2000)*, pages 121–139. Amer. Math. Soc., Providence, RI, 2001.

[10] J.A. De Loera. PUNTOS, http://www.math.ucdavis.edu/ deloera/, 2000.

[11] T. Dokken. Approximate implicitization. In *Mathematical methods for curves and surfaces (Oslo 2000)*, Innov. Appl. Math., pages 81–102. Vanderbilt Univ. Press, Nashville, 2001.

[12] I.Z. Emiris and J.F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symbolic Computation*, 20(2):117–149, 1995.

[13] I.Z. Emiris and I.S. Kotsireas. On the support of the implicit equation of rational parametric hypersurfaces. Technical Report TR-02-01, ORCCA, Canada, 2002.

[14] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. Discriminants of polynomials in several variables and triangulations of Newton polytopes. *Leningrad Math. J.*, 2(3):449–505, 1991. (Translated from *Algebra i Analiz* 2, 1990, pp. 1–62).

[15] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, resultants, and multidimensional determinants*. Birkhäuser, Boston, 1994.

[16] L. Gonzalez-Vega. Implicitization of parametric curves and surfaces by using multidimensional Newton formulae. *J. Symbolic Comput.*, 23(2-3):137–151, 1997. Parametric algebraic curves and applications (Albuquerque, NM, 1995).

[17] I.S. Kotsireas and E. Lau. Implicitization of polynomial curves. In *Proc. ASCM*, 2003. To appear.

[18] D. Manocha and J.F. Canny. Algorithm for implicitizing rational parametric surfaces. *Comput. Aided Geom. Design*, 9(1):25–50, 1992.

[19] T. Michiels and R. Cools. Decomposing the secondary Cayley polytope. *Discrete Comput. Geom.*, 23(3):367–380, 2000.

[20] T. Michiels and J. Verschelde. Enumerating regular mixed-cell configurations. *Discrete Comput. Geom.*, 21(4):569–579, 1999.

[21] J. Rambau. TOPCOM, 2001. www.zib.de/rambau/TOPCOM.

[22] B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Combin.*, 3(2):207–236, 1994.

"CONTOUR" Algorithm for Finding and Visualizing Flat Sections of 3D-Objects

D.V. Mogilenskikh

Russian Federal Nuclear Center – the All-Russian Scientific and Research Institute of Technical Physics named after academician E. I. Zababakhin
(RFNC-VNIITF)
456770, Snezhinsk, Chelyabinsk Region, Russia, P.O. Box 245
d.v.mogilenskikh@vniitf.ru

Abstract. One of the visualization problems implies finding boundaries and insides of flat sections of 3D objects specified analytically. This problem is to be solved to analyze the results and to control the geometry specification. On the one hand, the analytic definition of profiles for "not simple" 3D-objects in general case is actually unsolvable problem; on the other hand, it is necessary to apply virtual designing of 3D-objects in the field of numerical simulation.
The numerical algorithm "CONTOUR" is proposed for finding the connected section contour of 3D-objects, which are specified analytically in the form of second-order planes and rotation connected by multiple-theoretic operations. "CONTOUR" algorithm suggests using topologically parallel scanning for generating the empirical values, and then, considering them in the process of isoline generation, the polygonal section contour is constructed.

1 Introduction

The problem of creation of optimal numerical algorithms of finding (localization) and describing of boundaries and inside of flat sections of arbitrary 3D objects specified analytically is urgent. The urgent nature lies in the fact that analytical determination of section contours for "not simple" objects in a general case is actually unsolvable. From the other hand, there is the exigency to virtually construct objects with complex geometry in the filed of numerical simulation. For that reason the solution for finding and visualizing flat sections of 3D-objects is actual.

The given paper proposes the numerical algorithm CONTOUR for localization and description of section connected contour for 3D objects of a specific class. There are different ways and languages to describe essentially 3D geometry. One of the most common technique is analytical specification of primitives, which are connected by multiple-theoretic operations. In RFNC-VNIITF to describe complex geometry of objects the approaches are applied stated in papers [1-3]. The approach is based on application of second order surfaces (SOS) as primitives, which are connected by multiple-theoretic operations. With SOS one can describe geometry of various complexity. The set of SOSs connected by multiple-theoretic operations is called in the given paper as "Logical Block of Primitives" (LBP).

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 407-417, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Specific necessity of adequate visualization of analytically specified 3D objects can be seen in various areas of numerical simulation. For example, the problem of specifying and verifying of accuracy of complex geometries is urgent for any software package of Monte-Carlo technique (MCT). In RFNC-VNIITF the package of executable codes for simulation of particles MCT "PRIZMA" has been developed and intensely utilized and intensely runs with problems with diverse geometry [4, 5]. Another example of the necessity is generation of 3D difference grids.

The aim of creation of algorithm CONTOUR: To develop the numerical algorithm for localization and description of connected contour of arbitrary flat section of 3D object specified by LBP.

When visualizing the flat section of the objects specified by set-theoretical method, speed and visualization quality is especially critical. It is due to the fact that under the conventional approach to visualization of such data as in the algorithm of scanning rays (ASR Section 6), the possibility of optimal application of state-of-the-art graphical software and hardware means is restricted. However, at the conventional approach the visualization speed may be increased at the expense of enhancing a processor efficiency or parallel calculations. The problems:

1. It is practically impossible to obtain the connected section contours of complex objects in the form of oriented polygons by ray scanning.
2. It is extremely difficult in the general case to describe analytically the contours of flat sections of LBP.
3. The solution of the problem on accurate and rapid determination of sizes of the object and its location in space is complicated.

2 One Method of Specification of Arbitrary Geometry of a 3D Object

We present in brief one of the methods of description of essentially 3D geometry stated in papers [5, 7]. The method is based on the set-theoretical ground [6]. We mark a 3D object as: G^*. The surface of the object G^* is described by the SOS set: $\{F_1, F_2, \ldots, F_n\}$. The general SOS equation:

$$A_{xx}x^2 + A_{yy}y^2 + A_{zz}z^2 + A_{xy}xy + A_{xz}xz + A_{yz}yz + A_x x + A_y y + A_z z + A = 0.$$

This information is not enough for unique description of connected topology of the surface of the 3D object G^*. The geometry of the object G^* surface is specified uniquely [6], if: $\forall A \in R^3 \Rightarrow A \in G^* \vee A \notin G^*$, that is the space R^3 is uniquely divided into the two subspaces: $R^3 = G^* \cup G^* \backslash R^3$. The unique nature of description of the object G^* surface is obtained by the set-theoretical Boolean operations with the set $\{F_1, F_2, \ldots, F_n\}$. We mark the Boolean operation as: $\otimes$. The operation $\otimes$ means the following operations: $\cup$ - union, $\cap$ - intersection, \ - addition, $()$ - brackets. In the result the surface of the object G^* is specified uniquely: $G^* = F_1 \otimes F_2 \otimes \ldots \otimes F_n$. (1)

We formalize the fact of the arbitrary point $A(x_a, y_a, z_a) \in R^3$ belonging to the object G^*. It is important to note that the main thing that determines the SOS selection for description of objects geometry is the class of numerical simulation problems. **Fixed sign feature:** the surface $F(X, Y, Z) = 0$ is called of fixed sign, if $\forall A(x_a, y_a, z_a) \in R^3 \Rightarrow F(x_a, y_a, z_a) = 0 \vee F(x_a, y_a, z_a) < 0 \vee F(x_a, y_a, z_a) > 0$, then $F(X, Y, Z)$ is called of fixed sign.

We introduce the notion of the logical index $IND(F_i)$, which characterizes the position of the point A relative to each surface from the set $\{F_1, F_2, \ldots, F_n\}$. According to the set theory the index may have the values: 1 – true or 0 - false. According to the feature of fixed sign we shall consider, if: $F_i(x_a, y_a, z_a) = 0 \vee F_i(x_a, y_a, z_a) < 0$, then $IND(F_i) = 0$, otherwise $IND(F_i) = 1$.

The formalization of the fact of belonging of the point $A(x_a, y_a, z_a)$ from R^3 to the object G^*: $A \in G^*$, if $IND(G^*) = IND(F_1) \otimes IND(F_2) \otimes \ldots \otimes IND(F_n) = 1$, otherwise $A \notin G^*$.

3 Problem Statement

Let the finite system of the 3D objects be specified: $G = \{ G_i \}$, where $i \in [1, m]$. Each G_i object is specified as LBP (1). Let the arbitrary section plane (SP) P in R^3 be specified: $A_p x + B_p y + C_p z + D_p = 0$. (2)

Problem: to determine the section contour of objects from G with P surface and to visualize the obtained sections, it means – to uniquely describe the section contour and visualize it in R^2, in particular:

- To draw the section boundaries;
- To fill the section inside.

Assumption: not limiting the generality we will consider one LBP from G system, we mark it as G^*.

All the specified objects are determined in the basic Cartesian coordinates (BCS). The virtual screen, at which an image would be formed, belongs to P.

4 Coordinate Systems

With P plane the screen coordinate system (SCS) is connected. The implementation of mechanisms of SP specification, its movement and rotations may be different. That influences the technique of calculation of mutual location pf BCS and SCS. The equation (2) for P does not uniquely determine SCS, some additional conditions are required. It is always possible to determine the normalized matrix T of coordinates transformation from BCS into SCS and, correspondingly, the matrix T^{-1} of the reverse transformation from SCS into BCS. Figure 1 presents the compliance scheme of coordinate systems used by the author. We mark the coordinates of SCS starting in BCS: X', Y', Z'.

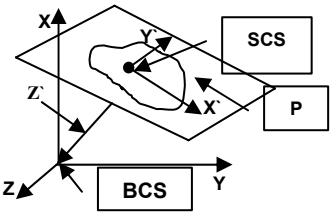


Fig. 1. SCS and BCS compliance scheme

Note 1: in SCS $\forall A \in P \Rightarrow z \equiv C \equiv Const$.
The note, obvious at first sight, has key significance for applying the algorithms of isolines localization in R^2 for localization and description of LBP section contour.

5 Sizes of Object Section

It is required to state briefly one of the important subtasks – the determination of describing sizes of potential section. For successful localization of section contours in SP it is necessary to know the approximate sizes of an object section. The complexity of the problem lies in complexity of its solution in automatic mode. The four variants of solution are proposed:

1. Empirical way of sizes specification – the researcher specifies the approximate sizes based on his knowledge.
2. The automatic method lies in coarse scanning of proposed object location.
3. Determination empirically or with coarse scanning of "good" inner point of an object. Then from this point the scanning rays are put out and the boundary points are determined, which compose the approximate sizes of the object.
4. Combination of all the methods.

We assume that in some way the section sizes are determined and reduced to SCS: ***X_max*** and ***X_min*** – sizes along ***X'*** axis, ***Y_max*** and ***Y_min*** – sizes along ***Y'*** axis.

6 Algorithm of Scanning Rays

6.1 Main Idea

Brief consideration of ASR is required because of the following reasons:

1. ASR may be called classical approach.
2. Methodical aspect for comparison with the algorithm CONTOUR.
3. The algorithm CONTOUR is based on two procedures from ASR for collecting preliminary information.

Figure 2 presents schematically the describing rectangle ***REG***, scanning rays parallel to ***X'*** axis, 3D-object and the section is marked. ASR allows logically finding the points of intersection of scanning rays with the object boundaries and, as the result, all the segments, which form the inside of the section (Fig. 2).

6.2 Localization Procedures

In the process of implementing ASR and CONTOUR algorithm the two procedures have basic importance. The author of the paper applied the procedures developed by the mathematicians from RFNC-VNIIF and presented in paper [1].
Procedure 1. Localization of a point from R^3 relative to the system of ***G*** objects.
Procedure 2. Determination of the nearest distance from a point from R^3 at the given direction till the boundary of G^* object, from ***G*** system.

The detailed description of the given procedures is beyond this paper. The implementation of the procedures is based on the set-theoretic approach considering the information on nesting objects in ***G*** system and the feature of a fixed sign.

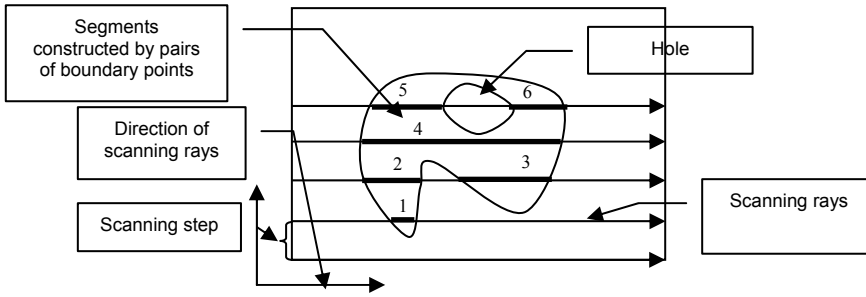


Fig. 2. Scheme of presenting of section inner segments

6.3 Localization of ASR Section Boundaries

We introduce the 1D list of inner segments (LIS) for coordinate wise successive writing of boundary points of G^* object. The dimension of LIS is not known beforehand. Scanning of the area of ***VIEW*** output is arranged as is presented in Figure 2. For each ray, with using **procedures 1** and **2,** all the boundary points of G^* object are localized successively and are written successively into LIS.

6.4 Visualization

ASR work results in the successive, arranged of the generation scheme for LIS. Figure 2 presents schematically the result of scanning. The inner segments are marked with thickness. These segments are numbered according to the number of their localization. **Example:** Illustrations at Figures 3a, 3b, 4a, 4b, 8a, 8b, 8c are obtained by the section by $Y = -1$ plane of the same G^* object. G^* object represents a muff with various curvature radii (Fig. 3a). Figure 3b presents 3D image of G^* object. G^* object is specified by the twelve surfaces:

$1.\ F_1: 5x + z - 15 = 0;$ $2.\ F_2: x^2 + z^2 - 9 = 0;$
$3.\ F_3: -5x + z - 15 = 0;$ $4.\ F_4: x^2 + z^2 - 10z + 21 = 0;$
$5.\ F_5: z = 0;$ $6.\ F_6: z = -5;$
$7.\ F_7: 5x + z - 10 = 0;$ $8.\ F_8: x^2 + z^2 - 4 = 0;$
$9.\ F_9: -5x + z - 10 = 0;$ $10.\ F_{10}: x^2 + z^2 - 10z + 24 = 0;$
$11.\ F_{11}: y = 0;$ $12.\ F_{12}: y = -5.$

The set-theoretical description of G^* object: $G^* = ((-2 \cup -4 \cup (+5 \cap -6 \cap -3 \cap -1)) \cap -11 \cap +12) \cap ((-8 \cup -10 \cup (+5 \cap -6 \cap -9 \cap -7)) \cap -11 \cap +12).$

The signs ± in the formula imply, which R^3 subspace takes part in the formula, since all the surfaces are of fixed sign [6].

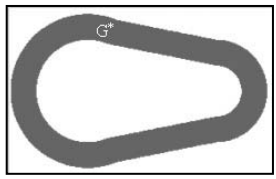


Fig. 3a. Section of 3D muff with ***Y = -1*** plane.

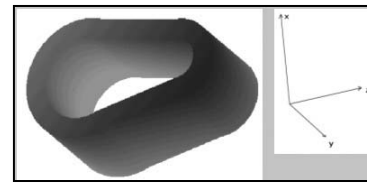


Fig. 3b. 3D presentation of **G*** object.

To visualize the section boundaries of G^* object it is enough to draw all the limit points of segments from LIS (Fig. 4a). To visualize the inside of section of G^* object it is enough to draw successively all the LIS segments (Fig. 4b).

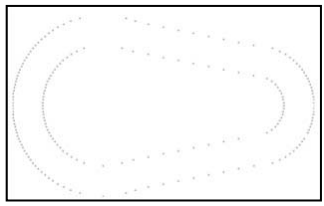


Fig. 4a. Visualization of section boundary.

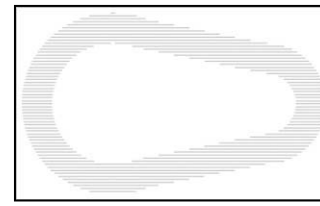


Fig. 4b. Visualization of inside of section at nondense scanning.

6.5 Imperfections of ASR at Visualizing Sections of 3D Objects

ASR imperfections and comments:

1. The section contour is not described with polygons, therefore drawing of the contour is performed by points (Fig. 4a), and filling of the inside is performed with the help of drawing of inner segments (Fig. 4b).
2. The method of generation of successive line of the section contour is possible in the form of oriented polygons after or while ASR working. However, at complex geometry (objects with holes, thin objects, non-convex objects) this method becomes practically difficult to apply.
3. To obtain good dense image a great number of scanning rays is required. The example of dense scanning is presented in Figure 3a.
4. For qualitative visualization of magnified part of an image it is required to perform scanning anew, since the image becomes striped.
5. When visualizing the contour boundary by points at the screen or in paper the image quality is bad - the staircase effect is obvious, the different visible thickness of lines. At this approach it is impossible to apply standard mechanisms for qualitative drawing, for example, vector graphics.
6. The filling of flat figures with segments provides very bad quality at 3D-visualization.

7 Algorithm of Localization and Description of Section Contour by Segment Successively. Method of Isolines Constructing by Special Scalar Field. (CONTOUR Algorithm)

7.1 Preliminary Constructions

Main idea: To construct in the area of ***VIEW*** output at rectangular 2D grid the special discrete scalar field with the help of ASR, then with the help of any correct algorithm of isolines generation to restore the successive contour of section of a 3D-object in the form of polygons.

In rectangular area of ***VIEW*** output we determine the 2D regular additional grid (AG) with the dimension ***(M_Grid x N_Grid)*** (Fig.5). The coordinates of AG nodes into BCS: ***X[M_Grid, N_Grid], Y[M_Grid, N_Grid]], Z[M_Grid, N_Grid]***.

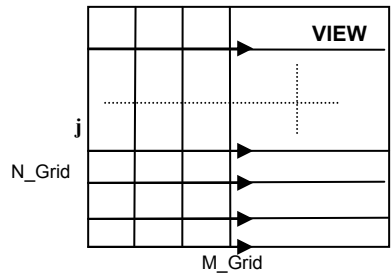


Fig. 5. Method of AG construction.

To describe CONTOUR algorithm the AG is considered, at which guiding lines are horizontal and vertical. This assumption may be generalized: guiding grids cannot be obligatory parallel to each other, that is the grid may be formed not obligatory rectangular, but the guiding grids should not intersect in the area of object section. We introduce the special empirical value ***DAT***, by which we will construct the scalar field. We introduce the matrix ***DAT[M_Grid, N_Grid]*** to write the value ***DAT***, which would be formed in nodes of 2-D grid. **The idea**: for each AG node one can indicate the belonging of the node to the inside of section of G^* object:
if ***DAT[i, j] >= 0***, then the node ***(i, j)*** $\in$ to the section ***G****,
if ***DAT[i, j] < 0***, then the node ***(i, j)*** $\notin$ to the section ***G*.***

7.2 Generation of Scalar Field at Additional Grid by Horizontal Scanning

Filling of ***DAT*** we perform with the help of ASR by a horizontal. We arrange the cycle by horizontal rays: ***j*** varies from ***0*** to ***N_Grid*** (Fig.5). Illustration of the algorithm of generation and AG fragment can be found in Figure 6.
Step 1. The current ray ***j***. We analyze the point of the ray start: ***B=(X[0, j], Y[0, j])***. We apply **procedure 1**. If the point ***B*** $\in$ ***G*,*** then ***DAT[0, j] = 1***, otherwise ***B*** $\notin$ ***G**** and ***DAT[0, j] = -1***. Let ***B*** $\notin$ ***G**** and ***DAT[0, j] = -1***.

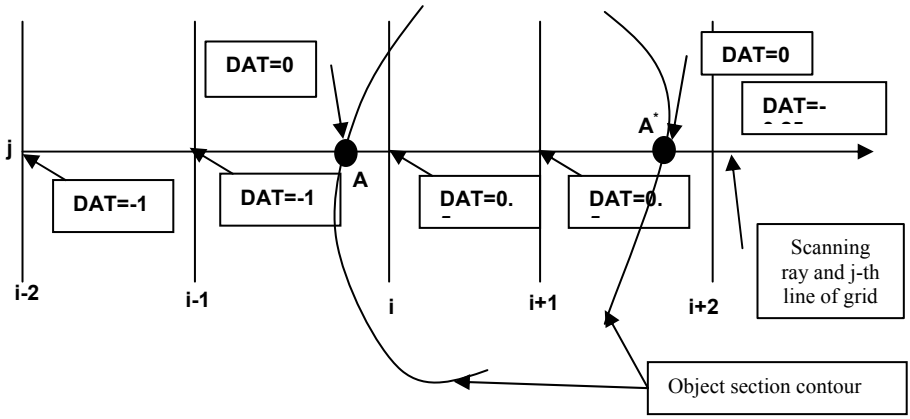


Fig. 6. Scheme of algorithm of generation of the value ***DAT*** at AG.

Step 2. We put out a ray from ***B*** point in horizontal direction. We apply **procedure 2** with horizontal directional vector. If there are no points of intersection with the boundary of ***G**** object, then ***DAT[i, j] = DAT[0, j]*** for ***1 ≤i ≤M_Grid***.
We pass to the next ray ***j+1*****,** and to **Step 1**. If there is a point of intersection with the boundary of ***G**** object: ***A(Xa, Ya)*****.**
Step 3. Let the point ***A*** lies between the nodes ***(i-1, j)*** и ***(i, j)***.
Basic moment: It is necessary that at linear interpolation of the value ***DAT*** into the point ***A*** from the nodes ***(i-1, j)*** and ***(i, j)*** the value of ***DAT*** in the point ***A*** was equal to ***0***, that is ***DAT(A) = 0***.
Hence, knowing the value of ***DAT*** in the node ***(i-1, j)*** and the condition that in the point ***A*** the value of ***DAT*** should be equal to ***0***, it is necessary to extrapolate the value of ***DAT*** into the node ***(i, j)***. Along the horizontal direction the coordinate ***X*** varies. We compile the proportion: ***(Xa-X[i-1, j])/(X[i, j]–Xa) = DAT[i-1, j] / DAT[i, j];***
We calculate the value ***DAT*** in the node ***(i, j)***: ***DAT[i, j] = DAT[i-1, j] * ((X[i, j] – Xa) / (Xa - X[i-1, j])).*** Figure 7 presents the curve for the value ***DAT*** by the grid guiding line and shows the linear extrapolation.
Step 4. We replace the current point: ***B = A***. Then from the point ***B*** we put out the ray further, that is we apply the **procedure 2**. If there are no points of intersection with the boundary of ***G**** object, then ***DAT[k, j] = DAT[i, j]*** for ***i+1 ≤k ≤M_Grid***.
We pass to the following ray ***j+1*****,** and to **Step 1**. If one can find the point of intersection with the boundary of the object ***G*: A*(Xa, Ya)*****.** We assume that the point ***A**** lies between the nodes ***(k-1, j) u (k, j)***, then ***DAT[n, j] = DAT[i, j]*** for ***i+1 ≤n ≤k-1***. We pass to **Step 4**.
Algorithm is complete. Generation of the empirical value ***DAT*** at AG is complete.

7.3 Generation of Section Contour

In the result of generation of the value ***DAT*** we obtain the discrete 2D scalar field.
Note 2: For the given distribution of the value ***DAT*** any correct algorithm of isoline construction at 2D grid by the value is applied: ***DAT = 0***.

The result of the operation of the algorithm of isoline construction is successive description of a corresponding line with points. The author applies the modified algorithm of isoline tracking from [7]. Figure 8a presents the results of operation of CONTOUR algorithm with horizontal generation of the value ***DAT*** .

Conclusions:

1. Smoothness of line of the section contour along the whole contour does not comply with the required quality.
2. To the left and to the right the lines smoothness is adequate.
3. It is necessary additionally for generation of the empirical value ***DAT*** to apply scanning by vertical guiding lines of the grid.

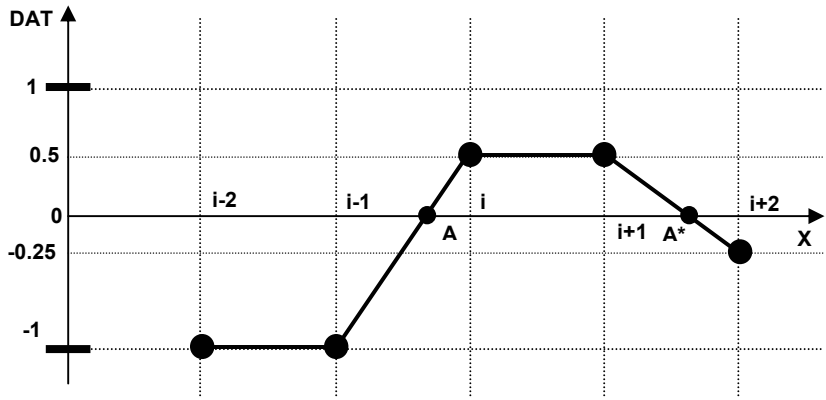


Fig. 7. Curve of the value DAT by scanning ray and intersection of ray with ***G**** object in the points ***A*** and ***A****.

7.4 Generation of Scalar Field at Additional Grid with Vertical Scanning

We introduce the empirical value ***DAT1***, which is similar to the value ***DAT*** by the idea, but is formed with the help of vertical scanning. Generation of the value ***DAT1*** at AG is similar, accurate within change of indices of the grid nodes, generation of the value ***DAT*** in Section 7.2. By that reason it is not necessary to repeat the stepwise description of generation algorithm for the value ***DAT1***. Let the array ***DAT1[M_Grid, N_Grid]*** is formed. Then we repeat the work of Section 7.3 for isoline generation for the value ***DAT1 = 0***. Figure 8b presents the results of operation of CONTOUR algorithm with vertical generation of the value ***DAT1***.

Conclusion:

1. Smoothness of the line of the section contour along the whole contour does not comply with the required quality.
2. Upward and downward the smoothness of the line if adequate.
3. When constructing the isoline it is necessary to consider the both accumulated empirical values.

7.5 Integral Generation of Object Section Contour

It is proposed in the process of isoline generation to consider the value of one empirical value or another. We consider in brief this solution in the process of implementation of one algorithm of isoline generation. The construction of an isoline is performed successively from one grid cell to a neighboring cell. The intersections of cells edges with the isoline and the direction of the isoline out from the cell is analyzed. The algorithm is added with verification of the two conditions:

1. If an isoline intersects the horizontal edge of the cell, then the coordinates of its point of intersection with the edge are found by linear interpolation with using the value ***DAT*** in the nodes of the corresponding edge.
2. If an isoline intersects the vertical edge of a cell, then the coordinates of its point of intersection with the edge are found by linear interpolation with using the value ***DAT1*** in the nodes of the corresponding edge.

Figure 8c presents the result of operation of integral algorithm CONTOUR with vertical and horizontal generation of the values ***DAT*** and ***DAT1*** correspondingly.
Conclusion: The solution is found and the algorithm of solution of the localization problem and successive description of the boundary section of 3D-object is constructed, the result of operation of CONTOUR algorithm is adequate. CONTOUR algorithm is a second-order algorithm.

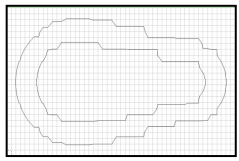


Fig. 8a

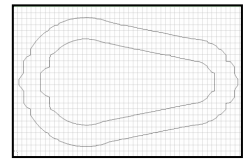


Fig. 8b

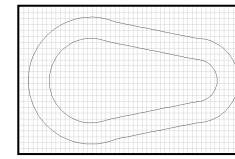


Fig. 8c

Fig. 8a. Example of presentation of section contour with horizontal section of the value ***DAT***.
Fig. 8b. Example of presentation of section contour with vertical generation of the value ***DAT1***.
Fig. 8c. Example of presentation of section contour with horizontal and vertical generation of the values ***DAT*** and ***DAT1***.

Figure 9 presents the example of localization and visualization of flat section of a more complex 3D object.

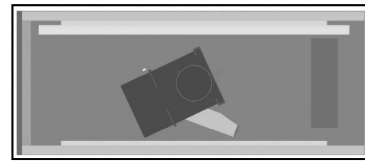


Fig. 9. Visualization of section with the plane ***Y=0*** of objects system

8 Resume and Conclusions

Result: On the basis of generation of special 2D additional grid and application of the algorithm of isoline construction the algorithm CONTOUR is developed and implemented for generation of successive contour of flat section of 3D object.
Implementation is simplified and quality of many visualization operations is improved, for example:

1. Filling of the section inside is performed in the form of filling of a polygon, in contrast to drawing by segments in ASR, that is especially important in 3D-visualization.
2. Magnification of an image becomes more simpler and faster operation.
3. When rotating an object in the section plane the drawing of boundaries and filling of the inside is performed correctly.
4. The screen cutting-off is performed simpler and faster.
5. It is possible to qualitatively and correctly draw the lines at vector printing devices.

The idea of CONTOUR algorithm may be transferred to a 3D case for 3D-visualization of a surface of a 3D-object. This idea has been already implemented in the system "**VIZI_PRIZMA"**. In the conclusion one should note that the described algorithms are implemented on the system of scientific visualization "**VIZI_PRIZMA"** and are applied for presenting and check of geometry in MCT.

References

1. A.V. Adeev, I.V. Adeeva, N.A. Pavlova. Set-Theoretical Approach for 3D Geometry in MC Software Package at ELBRUS-2//VII All-Union Conference "Monte Carlo Technique in Calculation Mathematics and Mathematical Physics". Theses. – Novosibirsk: AUC SB AS USSR, 1991.
2. Ogibin V. N., Kholushkin V. S., Nesterenko L. V. Problem-Oriented Interface for Solution for Equations with Partial Derivatives//Computational Technologies. – Novosibirsk: IVT RAS, 1992.
3. A.V. Adeev, I.V. Addeva, N.A. Pavlova On Geometrical Provision of Calculations by Monte Carlo Technique in RFNC-VNIITF. Preprint №160. Published in RFNC-VNIITF. Snezhinsk 1999.
4. Arnautova M. A., Kandiev Ya. Z., Lukhminsky B. E., Malyshkin G.N. Monte Carlo Simulation in Nuclear Geophysics. Intercomparison of the PRIZMA Monte Carlo Program and Benchmark Experiments // Nucl. Geophys. – 1993. – Vol. 7, N3. – P. 407-418.
5. Kandiev Ya. Z., Malyshkin G. N. Modeling by Value Implemented in Prizma Code / V Joint Russian-American Computational Mathematics Conference. Sandia Report. SAN98-1591.-1998.-P. 149-158.
6. Kuratovski K., Mostovski A. Set Theory. – M:.Mir, 1970.
7. Bayakovsky Yu. M., Galaktionov V. A., Mikhailova T. N. Grafor. Graphical enhancement of Fortran. – M.:Nauka, 1985.

A Non-uniform Binary Space Partition Algorithm for 2D Implicit Curves

Francisco Morgado and Abel Gomes

Networks and Multimedia Group – Institute for Telecommunications (IT)
Department of Informatics, University of Beira Interior, 6200-001 Covilhã, Portugal
fmorgado@di.estv.ipv.pt, agomes@di.ubi.pt

Abstract. Current graphical systems include primitives to draw straight-line segments, circles, Bézier curves and surfaces, NURBS (Non-Uniform Rational B-Splines), but many of them fail displaying curve singularities (self-intersections) correctly. This paper introduces a fast and robust non-uniform binary space partition (BSP) algorithm for implicit curves possibly with self-intersections and other differentiable singularities. These singularities are computed without using traditional differential techniques.

1 Introduction

An implicit curve $\mathcal{C}$ is a level set (or zero set) of some analytic function f from $\mathbb{R}^n$ to $\mathbb{R}$, say $\mathcal{C} = \{\mathbf{x} \in \Omega \subseteq \mathbb{R}^n : f(\mathbf{x}) = 0\}$. In this paper, we consider the problem of representing 2D implicit curves defined by analytic functions. In other words, we aim to compute a polygonal approximation for a curve $\mathcal{C} = \{(x, y) \in \Omega \subseteq \mathbb{R}^2 : f(x, y) = 0\}$ defined implicitly by some analytic function $f : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}$. Note that the corresponding algorithm applies not only to algebraic curves as in [4], but also to more general analytic curves (e.g. curves defined by transcendental functions). Basically, there are three categories of algorithms to represent implicit curves on a display screen, namely:

- *Representation conversion.* These algorithms convert an implicit curve into a parametric curve in order to easily display it on a screen [2], [5], [11]. However, rarely there is a global parameterization for an implicit curve, unless it is regular. In fact, a local parameterization always exist in a neighborhood of a regular point of an implicit curve, i.e. a point $\mathbf{p} = (u, v)$ such that $f(\mathbf{p}) = 0$ and $\nabla f \neq 0$. In this way, we can render a regular implicit curve through rendering algorithms for parametric curves.
- *Curve tracking.* The idea behind this class of algorithms is to follow the curve point after point in a way similar to parametric curves [7], [12]. This approach has its roots in the Bresenham's algorithm for rendering circles, which is basically a continuation method in image space. A simple continuation method consists of integrating the Hamiltonian vector field $(-\partial f/\partial y, \partial f/\partial x)$, combining a simple numerical integration method with a Newton corrector [1]. These methods are attractive because they concentrate effort where it is

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 418–427, 2003.
© Springer-Verlag Berlin Heidelberg 2003

needed, and may adapt the computed approximation to the local geometry of the curve. Unfortunately, they need a starting point on each component of the curve.
- *Space subdivision.* It consists of dividing the ambient space into subspaces, discarding those not intersecting the curve. The subdivision is done recursively and terminates when the resulting approximation to the curve by a set of small subspaces (e.g. small rectangles) [5], [6], [15] is good enough. Robust algorithms can be implemented by using algebraic techniques and interval arithmetic [14], algebraic or rational techniques [9], [8], and floating-point arithmetic [13]. However, the space subdivision techniques were not attractive for many researchers because of their low speed-up for most applications [14]. But, recently [10] presented a technique using interval arithmetic that allows speeding up the curve rendering.

This paper deals with a new space subdivision algorithm for planar implicit curves. Unlike other space subdivision algorithms, it is shape-adaptive in the sense that there are more space subdivisions where the shape of the curve changes more significantly. As a consequence, the space subdivision is not uniform. The paper is organized as follows. Section 2 describes the non-uniform binary space partition, including the BSP data structure. Section 3 describes the continuation of points embedded in the BSP data structure that enables rendering the curve. Section 4 shows some relevant experimental results. At last, some conclusions are drawn in Section 5.

2 Non-uniform Binary Space Partition

2.1 BSP Data Structure

Fig. 1 illustrates the non-uniform BSP (Binary Space Partition) technique for a planar implicit curve $\mathcal{C}$. It recursively splits up the subspace $\Omega \in \mathbb{R}^2$ where the curve lies in into two spaces Ω_{left}, Ω_{right} by some straight-line l intersecting the curve, called the BSP line. If a subspace contains a curve segment, it is partitioned again, unless the distance between the endpoints of the curve segment is less than or equal to a small tolerance ϵ, or the curvature along the curve segment does not change too much. Subspaces without any curve segment are discarded, and those where the curvature of the curve changes are further split up. This way, our technique adapts to the shape of the curve.

In terms of C++ code, the corresponding BSP data structure is as follows:

```
class BSP {
    List *fr;
    List *lip;
    BSPline *l;
    BSP *left, *right;
    BSP *next;
}
```

The variable `fr` denotes the frontier $\mathrm{Fr}(\Omega)$ of a convex subspace Ω. It consists of a list of straight-line segments bounding Ω; `lip` is a list of points resulting from the intersection $\mathrm{Fr}(\Omega) \cap \mathcal{C}$; `l` is the line that splits up Ω into two subsidiary subspaces Ω_{left} and Ω_{right}; `left` denotes Ω_{left}, while `right` denotes Ω_{right}; `next` is used for rendering $\mathcal{C}$ and represents the next left leaf of the BSP tree as explained further ahead.

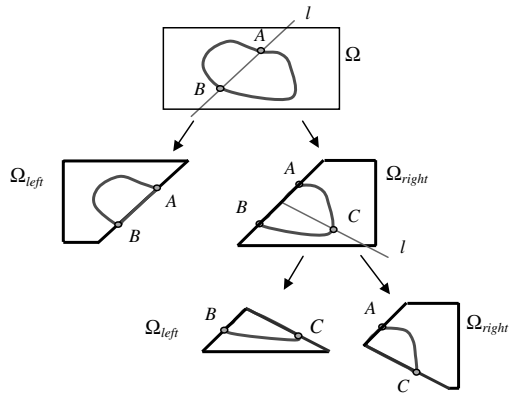


Fig. 1. The BSP tree.

2.2 BSP Line

Determining a BSP line involves the following computations on the frontier of the subspace to be split up:

- Determine the intersection points $\mathrm{Fr}(\Omega) \cap \mathcal{C}$;
- Choose the most distant points P, Q amongst those in $\mathrm{Fr}(\Omega) \cap \mathcal{C}$;
- Determine the mediatrix l of the segment $\overline{PQ}$.

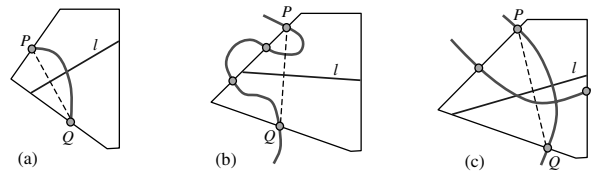


Fig. 2. BSP lines for three subspaces.

The BSP line is precisely the mediatrix of $\overline{PQ}$ (Fig. 2). The idea behind the choice of the most distant points P, Q is to partition the subspace equally as much as possible. The result is a more balanced BSP tree data structure. This eventually prevents memory stack overflow.

The corresponding algorithm to compute the BSP line is then as follows:

Algorithm 1 (BSPLINE)
INPUT:

(a) Ω: a subspace of $\mathbb{R}^2$

OUTPUT:

(a) l: a BSP line

Begin

1. **if** (Ω is the initial subspace)
   - $l \leftarrow$ an arbitrary line through the center of Ω intersecting $\mathcal{C}$
2. **else**
   (a) $lip \leftarrow \mathrm{Fr}(\Omega) \cap \mathcal{C}$
   (b) **if** ($!lip$)
   - $l \leftarrow$ an arbitrary line through the center of Ω
   (c) **else**
   - determine the most distant points $P, Q \in lip$
   - $l \leftarrow$ mediatrix of $\overline{PQ}$
   - **if** $(\mathrm{d}(P,Q) < \epsilon)$ **return** NULL
   - **if** $(\mathrm{d}(P,Q) < \tau)$ and $(\#(\mathrm{Fr}(\Omega) \cap \mathcal{C}) == 2)$
     - $R \leftarrow l \cap \mathcal{C}$
     - **if** $(\angle(\overline{RP}, \overline{RQ}) \approx 180°)$ **return** NULL
3. **return** l

End

The BSP technique does not depend on the existence of curve singularities such as cusps and self-intersections. Besides, if a subspace contains a self-intersection, its recursive partition tends to converge to such a singularity. Therefore, it would be interesting to exploit the non-uniform BSP technique to determine self-touching points and self-intersections somehow. For that, we would start by identifying subspace leaves of the BSP tree with at least three curve points in its frontier, i.e. $\#(Fr(\Omega) \cap \mathcal{C}) \geq 3$. However, for intermediate nodes, this is not a valid condition for the existence of, for example, a self-intersection as illustrated in Fig. 2(b). Singularities can be only determined in subspace leaves. In fact, subspace leaves are small enough that if the condition $\#(Fr(\Omega) \cap \mathcal{C}) \geq 3$ is true, we have a singularity in it surely. Note that subspace leaves are small because they satisfy the condition $\mathrm{d}(P,Q) < \epsilon$, i.e. the distance between the most distant points $P, Q \in \mathrm{Fr}(\Omega) \cap \mathcal{C}$ is less than ϵ. Thus, a partition stopping criterion is that the maximum distance between any two curve endpoints within a subspace is less than a small ϵ, as shown in BSPLINE algorithm. Under these

conditions, we assume that a self-touching point, and also a self-intersection point, can be approximated to the midpoint of $\overline{PQ}$.

A second partition stopping criterion has to do with the control of the curvature. This is done by the conditions $\mathrm{d}(P,Q) < \tau$ and $\#(\mathrm{Fr}(\Omega) \cap \mathcal{C}) == 2$ in BSPLINE algorithm, where $\tau = 3\epsilon$. In this case, if the angle $\angle(\overline{RP}, \overline{RQ}) \approx 180°$, the subspace partition stops, i.e. no BSP line is returned. (The point R is the intersection point between the mediatrix l and the curve $\mathcal{C}$.) Otherwise, the BSP line previously determined is returned and the subspace partition proceeds.

2.3 Position of a Curve Point in Relation to the BSP Line

Before splitting up a subspace Ω into two subspaces Ω_{left}, Ω_{right} by a straight line l, we have to classify the curve points intersecting the frontier of Ω as belonging to either Ω_{left} or Ω_{right}.

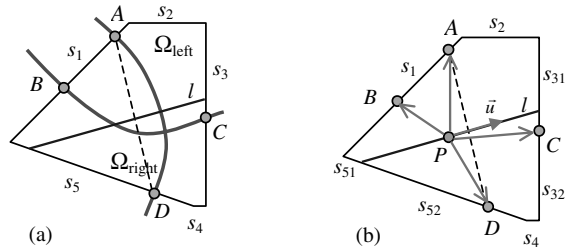


Fig. 3. Positioning curve points in relation to the BSP line.

Let P be an intersection point of the curve with the BSP line l (Fig. 3). The classification algorithm is as follows:

- Determine the unit vector $\overrightarrow{u}$ at P with the direction of l;
- Determine the unit vector $\overrightarrow{w}$ orthogonal to Ω;
- Determine the unit vector $\overrightarrow{PX}$ for any curve point X intersecting the frontier of Ω;
- If the mixed product $(\overrightarrow{U} \times \overrightarrow{V}).\overrightarrow{w} > 0$, with $\overrightarrow{U} = \overrightarrow{u} \times \overrightarrow{PX}$ and $\overrightarrow{V} = \overrightarrow{u} \times \overrightarrow{PY}$, the points X, Y belong to the same subspace, either Ω_{left} or Ω_{right};
- If the mixed product $(\overrightarrow{U} \times \overrightarrow{V}).\overrightarrow{w} < 0$, with $\overrightarrow{U} = \overrightarrow{u} \times \overrightarrow{PX}$ and $\overrightarrow{V} = \overrightarrow{u} \times \overrightarrow{PY}$, the points X, Y belong to distinct subspaces.

In Fig. 3, the points A, B belong to Ω_{left}, while C, D belong to Ω_{right}. If the mixed product $(\overrightarrow{U} \times \overrightarrow{V}).\overrightarrow{w} = 0$, the point X belongs to both Ω_{right} and Ω_{left}. This happens when X (e.g. P in Fig. 3)(b) is an intersection point between the curve and the BSP line.

2.4 Creation of the Subsidiary Subspaces Ωright and Ωleft

After classifying the points $\mathrm{Fr}(\Omega) \cap \mathcal{C}$ as belonging to either Ω_{right} or Ω_{left}, we have to form these subsidiary subspaces. This is equivalent to form their frontiers. This is illustrated in Fig. 3, where $\mathrm{Fr}(\Omega) = \{s_1, s_2, s_3, s_4, s_5\}$ is the frontier of a convex subspace Ω in $\mathbb{R}^2$ and l a straight-line intersecting $\mathrm{Fr}(\Omega)$ at exactly two points. The first intersection point subdivides s_3 into two smaller segments, s_{31} and s_{32}, while the second subdivides s_5 into s_{51} and s_{52}. These two intersection points originate the segment $s_l \subset l$ that splits up Ω into $\Omega_{right} = \{s_l, s_{32}, s_4, s_{52}\}$ and $\Omega_{left} = \{s_l, s_{51}, s_1, s_2, s_{31}\}$.

Of course the creation of both subspaces requires the classification of its bounding segments in relation to the splitting line l. The segment classification is based on the point classification described in the previous subsection. The mixed product $(\overrightarrow{U} \times \overrightarrow{V}).\overrightarrow{w}$ is greater or equal to zero for any two segment endpoints bounding the same subspace, and less than zero for endpoints belonging to distinct subspaces.

The corresponding algorithm to compute the subsidiary subspaces is:

Algorithm 2 (SUBSPACES)
INPUT:

(a) l: the partition line of Ω
(b) Ω: a subspace of $\mathbb{R}^2$

OUTPUT:

(a) Ω_{left}: $\Omega_{left} \subset \Omega$
(b) Ω_{right}: $\Omega_{right} \subset \Omega$

Begin

1. $\Omega_{left} \leftarrow \varnothing$
2. $\Omega_{right} \leftarrow \varnothing$
3. Determine two intersection points $\mathrm{Fr}(\Omega) \cap l$
4. Classify points $\mathrm{Fr}(\Omega) \cap \mathcal{C}$ and $\mathrm{Fr}(\Omega) \cap l$
5. Update frontier points Ω_{left}
6. Update frontier points Ω_{right}

End

2.5 Intersection between a Curve and a Straight-Line

The secant method is a root-finding algorithm which assumes that a function to be approximately linear in the region of interest [3]. Otherwise, there is no guarantee that it converges. Basically, it starts from two distinct estimates A and B for the root of $f(x) = 0$. An iterative process involving linear interpolation

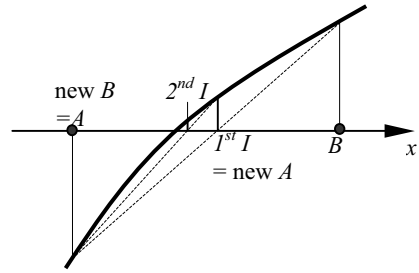


Fig. 4. Illustration of the secant approximation method.

updates A and B, but only the most recent estimate, say A, is retained. The interpolation is given by the formula

$$I = B - f(B).\frac{(B-A)}{f(B)-f(A)} \tag{1}$$

being I is the next point that approximates the intersection point.

The secant approximation method is part of the algorithm that computes the intersection point between the curve $\mathcal{C}$ and a subspace mediatrix l. Initially, the mediatrix is subdivided into several smaller segments. Then, the secant method is applied to each small segment that satisfies the condition $f(A).f(B) < 0$. This condition guarantees that there is an intersection point $\mathcal{C} \cap l$ between A and B. The division of the mediatrix into smaller segments is needed to guarantee that the secant method converges for each segment satisfying the above condition, and also because it there may be two or more intersection points.

Let $MAXLENGTH$ be the length of the admissible longest mediatrix, i.e. the length of the diagonal of the initial space Ω, and $MAX = 10$ be the maximum number of segments in such a longest mediatrix. The subdivision of a subspace mediatrix is adaptive in the sense that the number of its subsidiary segments depends on its $LENGTH$. If $LENGTH \leq (MAXLENGTH/MAX)$, then one applies the secant method to the mediatrix itself; otherwise, one subdivides the mediatrix into a number of segments given by

$$NSEG = (MAX\,.\,LENGTH)/MAXLENGTH \tag{2}$$

applying then the secant method to each segment.

3 Continuation and Rendering of the Curve

To render an implicit curve, we have some mechanism to traverse the BSP tree in order to sequence the computed points of $\mathcal{C}$. Looking at Fig. 1, we see that we have only to sequence the left leaves of the BSP tree. These left leaves are collected into a list to speed up the rendering process. This has the advantage

that there is no need to visit all tree nodes. Visiting only the left leaves represents a gain of 50% in rendering $\mathcal{C}$ in relation to a full traversal of the BSP tree. In fact, if the BSP tree goes down to the level N, its overall number of nodes is $2^{N+1} - 1$, but only 2^N leaves are visited.

The overall rendering algorithm for implicit curves is as follows:

Algorithm 3 (CURVE)
INPUT:

(a) $\mathcal{C}$: the curve

Begin

1. $\Omega \leftarrow$ a rectangular subspace of $\mathbb{R}^2$;
2. BSP(Ω,7);
3. Render $\mathcal{C}$;

End

The sub-algorithm BSP (step 2) is the main part of the non-uniform binary space partition algorithm. It can be described as follows:

Algorithm 4 (BSP)
INPUT:

(a) Ω: a subspace of $\mathbb{R}^2$
(b) d: degree of BSP tree or recursion level

Begin

1. **if** (d == 0) **return** NULL;
2. $l \leftarrow$ BSPLINE(Ω)
3. **if**(l)
   - SUBSPACES($l, \Omega, \Omega_{left}, \Omega_{right}$);
   - BSP($\Omega_{left}, d-1$);
   - BSP($\Omega_{right}, d-1$);

End

Note that there is a third partition stopping criterion given by the recursion level or BSP tree degree, which is assumed to be equal to 7.

4 Experimental Results

Although our algorithm uses a space subdivision technique, it proved to be fast. Unexpectedly, we noted that the algorithm performs faster where the shape of the curve changes more significantly. This happens because the number of subspaces increases where the shape of the curve is non-symmetric or irregular. The curves pictured in Fig. 5 were drawn with a precision of $\epsilon = 10^{-3}$ and maximum

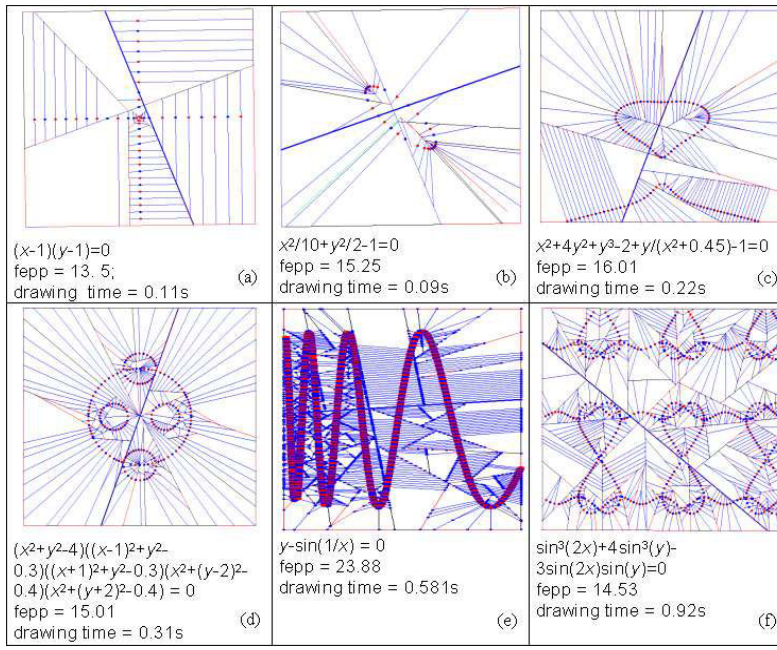


Fig. 5. Experimental results of some implicit curves.

recursion level of the BSP tree equal to 7. The term "fepp" denotes the number of function evaluations per computed point. This is so because every solution point is determined by an iterative approximation method (i.e. the secant method). The runtime performance tests were performed on a PC equipped with 500MHz Intel Pentium, 128MB RAM, and running Windows NT.

Commercial software packages such as the Maple and Mathematica do not incorporate accurate algorithms to draw implicit curves. In general, they are able to draw implicit curves somehow, but they fail at self-intersection points. For example, they cannot draw correctly the curve $(x-1)(y-1) = 0$ in Fig. 5(a) about the point $P = (1, 1)$. In fact, P is a self-intersection point of this curve that is the union of two straight lines, $x - 1 = 0$ and $y - 1 = 0$.

In general, space subdivision algorithms draw implicit curves correctly (see, for example, [10]). But, unlike the algorithm proposed in this paper, they are not able to distinguish a singularity from a regular point. However, similar to other space subdivision algorithms, our algorithm cannot identify isolated points.

5 Conclusions

A general implicit curve algorithm has been proposed. It is based on a non-uniform space partition technique. A curve needs not be closed or connected,

though single-point components are not detectable anyway. Besides, a curve may possess singularities and ripples.

Remarkably, this algorithm detects and accurately draws self-intersection and self-touching points without using any differential calculus tools. The computation of points is done through an iterative approximation technique, i.e. the secant method.

The algorithm proved to be fast enough to be included into current graphical systems. But, its speed depends on the efficiency of the algorithm that computes the intersection between a straight-line segment and the curve.

References

1. Allgower, E. and Georg, K. Numerical Continuation Methods: An Introduction. Springer-Verlag, (1990)
2. Allgower, E., Gnutzmann, S.: Simplicial Pivoting for Mesh Generation of Implicitly Defined Surfaces. Computer Aided Geometric Design, Vol. 8, (1991) 305–325
3. Akai, Terrence J.: Applied Numerical Methods for Engineers. John Wiley Λ Sons Inc (1994)
4. Blinn, J.: A Generalization of Algebraic Surface drawing, ACM Transactions on Graphics. Vol. 1, No. 3, (1982) 235–256
5. Bloomenthal, J.: Poligonisation of implicit surfaces. Computer Aided Geometric Design, Vol. 5, (1988) 341–355
6. Bloomenthal, J.: An Implicit Surface polygonizer. Graphics Gems, IV, (1994)
7. Chandler, R.: A tracking algorithm for implicitly defined curves. IEEE Computer Graphics Λ Applications, Vol. 8, No. 2, (1988) 83–89
8. Keyser, J., Culver, T., Manocha, D., Krishnan, S.: MAPC a library for efficient and exact mnipulation of algebric points and curves. Proceedings of the 15th ACM Symposium on Computational Geometry, (1999) 360–369
9. Krishnan, S., Manocha, D.: Numeric symbolic algorithms for evaluating one-dimensional algebric sets. Proceedings of ACM Symposium and Algebraic Computation, (1995) 59–67
10. Lopes, H., Oliveira, J.B., Figueiredo, L. H.: Robust Adaptive Polygonal Approximation of Implicit Curves. Proceedings of SibGrapi (2001)
11. Lorensen, W., Cline, W.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, Vol. 21, No. 4, (1987) 163–169
12. Moller, T., Yagel, R.: Efficient Rasterization of Implicit Functions. (1995). (http://citeseer.nj.nec.com/357413.html)
13. Shewchuk, J.: Adaptative precision floating-point arithmetic and fast robust geometric predicates. Discrete Computational Geometry, Vol. 18, No. 3, (1997) 305–363
14. Snyder, J.: Interval arithmetic for computer graphics. Proceedings of ACM Siggraph, (1992) 121–130
15. Triquet, F., at al.: Fast Polygonization of Implicit Surfaces. WSCG' 2001, Vol 2. (2001) 283-290

Web Visualization of Function-Defined Shapes

Alexei Sourin[1], Lai Feng Min[2], Konstantin Levinski[1]

[1]Nanyang Technological University, Singapore
{assourin, P44-0521757}@ntu.edu.sg
http://www.ntu.edu.sg/home/assourin
[2]Creative Technology Ltd., Singapore
fmlai@ctl.creative.com

Abstract. This paper describes how function-based shape modeling can be expanded to web visualization, as well as how web-based visualization can be greatly improved by using the function-based shape modeling technique. We have proposed a function-defined VRML shape node, which allows the content creators to describe any complex models with relatively small functions compared to the large-size polygonal mesh models. These function-defined shapes can be used together with the common VRML shapes. The design, the implementation details, and the application examples of the proposed node are discussed.

1 Function-Based Shape Modeling

Dramatic advances in computer technology have driven up the methods and tools of interactive 3D shape modeling where the unique geometric reasoning of the human brain couples with the rapid computing and rendering abilities of the computer. Some geometric models, which existed rather theoretically just a few years ago, now revolutionize the ways of shape representation and rendering. Thus, there can be noticed a trend toward function-based shape modeling.

The idea of function-based approach to shape modeling is that complex geometric shapes can be produced from a "small formula" rather than thousands of polygons. Usually, parametric or implicit functions and their modifications are used to define the shapes. For rendering such function-defined shapes, either ray tracing or polygonization followed by fast polygon rendering is used. Alternatively, the function-defined shapes can be voxelized and rendered as a set of points. It must be mentioned, though, that function-defined shapes may sometimes suffer a serious problem, which is large time needed for evaluation of the defining functions. Nevertheless, many works have been done in the direction of accelerating the function evaluation when performing rendering.

In the next sections, we overview the existing web visualization projects, justify the usefulness of the function-based modeling in web visualization, and propose our own solution to function-based web visualization. This work is a continuation of the project briefly described in [1].

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 428-437, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Web-Based Visualization

A variety of web-based visualization systems have been developed. Progressive reconstruction and isosurface transmission suggested by Engel et al. [2][3] aimed to reduce the amount of data to be reconstructed and transmitted during visualization. Seidel et al. [4] have presented a framework to acquire high quality 3D models of real world objects including both geometry and appearance information presented by textures, bump maps or bi-directional reflectance distribution function used to describe the way a surface reflects light. Fogel et al. [5] have proposed a web architecture for progressive delivery of 3D content, which is based on a progressive compression representation integrated into X3D framework. Taubin et al. [6] have proposed another popular approach to delivering a 3D content over the Internet within a reasonable time, besides introducing a new adaptive refinement scheme [7] for storing and transmitting the triangular meshes in progressive and highly compressed form. Pajarola and Rossignac [8] have proposed a compressed progressive meshes approach, which used a new technique to refine the topology of the mesh in batches.

Among these services, the polygon mesh representation is commonly used for representing geometric shapes. Thus, the most popular 3D web content format Virtual Reality Modeling Language (VRML) [9] uses this representation for complex objects. To improve the performance of VRML and to enrich its content, various VRML extensions have been implemented. For example, Alexa et al. [10] have suggested the Morph node that helps to interpolate between several shapes. Another example is customized GeoVRML [11] node, which is defined to provide a suite of solutions for representing and visualizing geographic data using a standard VRML97 browser. Wyvil and Guy [12] have introduced a new VRML extension for skeletal implicit surfaces with a limited set of operations like blending, warping and Boolean operations. Besides that, Pittet et al. [13] have proposed a new real-time Isosurfacing node, which is based on the so-called Marching Cubes algorithm [14] and allows real time rendering of an isosurface from 3D source data. Grahn et al. [15] have introduced trimmed NURBS in VRML to allow visualization of complex CAD models in VRML. Ginis and Nadeau [16] have presented new VRML extensions for scientific visualizations.

3 FShape Node for VRML

In this project, a generic function-based FShape node for VRML is proposed as a plug-in to VRML browsers. The idea of introducing such a node is to replace large polygonal representations of complex shapes with small formulae, which will be transmitted through the Internet much faster and therefore will improve the efficiency of web visualization. Besides that, these function-based shape models have an unlimited level of detail.

The *FShape* node is designed in such a way that it offers a generic infrastructure to support various function-defined models to be plugged-in during runtime. This can be achieved if the respective parsing and polygonization modules are implemented according to the defined plug-in interface. With this node, the content publishers can

present any proprietary function-defined model to the viewers since they can develop the respective parsing and polygonizing modules.

In general, a VRML node can be implemented either via plug-in customization or through VRML EAI interface [17]. For plug-in customization, the overall performance of the extended node will be much better than the other one, since the module is implemented with a low level programming language. The performance improvement becomes obvious when it involves complex geometric models, which may require long computations and algorithm evaluations. Although the plug-in customization may closely tie the node extension with the supported VRML browser plug-ins, the overall performance may need to be considered first if the node extension involves heavy computation processes. Another possible approach to implementation of the extension is via a VRML Script node and its EAI interface. The implementation using JavaScript or Java may allow extension across different browsers, however this approach may have a serious performance setback. The polygonization implemented using JavaScript language will only be interpreted when it is about to be executed. Therefore, the function models represented using a script language will always need more time to get executed as compared with the precompiled modules.

As overall performance is very important in the context of web visualization, the proposed *FShape* node is VRML browser plug-in dependent, in which the VRML extension is implemented via customizing a VRML browser plug-in. The proposed function-based geometric node could be extended via different browser plug-ins, e.g. Blaxxun Contact 3D [18], OpenVRML [19], FreeWRL [20], and other customizable browsers. In order to prove the concept proposed in this paper, we have extended Blaxxun Contact 3D to support *FShape* node. With the extension, viewers are able to visualize function-based objects in VRML paradigm. The extension of other VRML browsers can be done in a similar way, and we are working on it. The proposed *FShape* node definition is listed in Figure 1.

The visualization pipeline in *FShape* node is categorized into three main modules: ***Seeker***, ***Parser*** and ***Polygonizer*** components.

The ***Seeker*** component is used to obtain any type of function definition, which is either located inline with the VRML source or is remotely located at the content server. The inline source will always have higher precedence than the external source. The Seeker component will be triggered first whenever the scene graph traversal engine encounters the *FShape* node. As data seeking is independent from any function-based shape modeling approach, the Seeker component is embedded as part of the core module.

In order to allow for future extension of the ***Parser*** component in runtime, the component is dynamically loaded into the *FShape* core module. The component is dependent on the data used to describe the model. For every supported function based model, the respective Parser module is to be developed separately with a set of standard function calls. The Parser component is to be created as a dynamic linked library and stored into the relative *Parser* directory of the core module

Like the Parser component, the ***Polygonizer*** component is dependent on the chosen function language/model used to describe the shape. For every supported function-based model, the respective Polygonizer component is to be developed and a set of standard function calls must be supported. The Polygonizer component is to be cre-

ated as a dynamic linked library and stored into the relative *Polygonizer* directory of the core module.

```
FShape {  exposedField SFString sourceString     ""
          # contains the definition of a function-defined shape.
          exposedField SFString sourceURL        ""
          # alternatively, defines the path to the model source file.
          exposedField SFString objectName       "my_model"
          # specifies the model name for a function-defined shape.
          field SFString sourceType              ""
          # identifies the type of the model referred in the scene.
          field SFBool  ccw                      TRUE
          # specifies the order of points in a face created by Polygonizer
          field SFBool  convex                   TRUE
          # specifies if the faces created by Polygonizer are convex or not.
          field SFFloat creaseAngle              0
          # specifies an angle threshold.
          field SFBool  normalPerVertex          TRUE
          # specifies whether the normal of each vertex is computed.
          field SFBool  solid                    TRUE
          # determines whether to draw both sides of a face or just the front side.
          field SFBool  reduce                   FALSE
          # defines whether to produce the least number of polygons.
          field SFBool  searchVertex             FALSE
          # turns on recursive searching for vertex positions.
          field SFFloat searchPer                0.1
          # specifies the search accuracy in percentage of cell lengths.
          field MFFloat modelPar                 0.0
          # specifies model-specific parameters if any.
          field MFInt32 gridSize                 30
          # specifies the rendering resolution used in polygonization.
          field MFFloat boundingBoxMax           10
          # specifies the bounding box for the shape.      }
```

Fig. 1. FShape node definition with default values.

4 Support for Implicit Functions and the F-Rep Model

In order to support any proprietary function-defined models in *FShape* node, one has to develop respective Parser and Polygonizer components based on the predefined plug-in interfaces. Developing a Parser depends on the model and/or language used, while developing a Polygonizer depends only on the model. To illustrate the proposed design and to prove the correctness of our assumptions, we shall describe in this section just two of the developed plug-ins. These are the *FShape* node plug-ins for shapes defined with implicit functions and the function representation model named *F-Rep* [21, 22], as well as the F-Rep based geometric high-level language *HyperFun* [23].

The F-Rep, being an extension of implicit functions, assumes that geometric shapes are defined with an inequality $f(x,y,z) \geq 0$, where the real function f is positive for the points inside the shape, equal to zero on its border and negative outside it. The F-Rep functions can be defined either analytically, or procedurally with a function evaluation algorithm. Different operations can then be applied to the F-Rep geometric models. In fact, any operation that can be defined with a function $f(x,y,z) \geq 0$, which provides at least C^0 continuity, may be used for making a shape. All the functions are applied as function superpositions thus allowing one to create sophisticated hierarchically organized shapes, which are in turn function-defined shapes. F-Rep allows one to use together practically any geometric models since this representation is very generic and highly extensible.

When using F-Rep models developed elsewhere and stored in a proprietary data format, the respective Parser module is a rather simple procedure. It just reads the function model from the data file and makes it available for the Polygonizer. For testing purposes, we used our own F-Rep models developed in the project Interactive Function-based Shape Modeling [24], [25] and [26].

To facilitate the inclusion of any proprietary function-defined model, we developed a generic Polygonizer, which works efficiently for a large range of implicitly defined and F-Rep defined shapes. The algorithm, described in details in [25] and [27], is an extension and improvement of the so-called *continuation algorithm*, which calculates polygons only on the surface of the shape. Since its description is beyond the scope of this paper, we refer the reader to the printed papers for the details.

The *FShape* node will dynamically load the corresponding Parser and Polygonizer components based on the string value stored in *sourceType* field. The plug-in polygonizer supports several level of detail (LOD) depending on the distance between the observer and the shape. The LOD switching is to be done outside the *FShape* node acting like a group node. The example of a VRML source code that links to this proprietary function model is shown in Figure 2. In this code, the model is encoded to ASCII using base64 format and used inline. The examples of using such function-defined shapes in VRML scenes are presented in Figure 3.

```
#VRML V2.0 utf8
EXTERNPROTO FShape [...]
geometry DEF vase-FACES FShape {
sourceString "AQAAAAEAAABFAAAABwAAADDFFGGHJJHJKL..."
sourceType "tetrapoly" gridSize   24 ccw FALSE  }
```

Fig. 2. VRML source code for the proprietary F-Rep shape model used inline.

Since the Polygonizer was developed for a large class of models, in order to use any other implicitly or F-Rep defined models, only the respective Parser procedure is to be written. It just should be able to evaluate the shape defining function $f(x,y,z) \geq 0$ for any given coordinates. The examples of such an implementation can be found in [28].

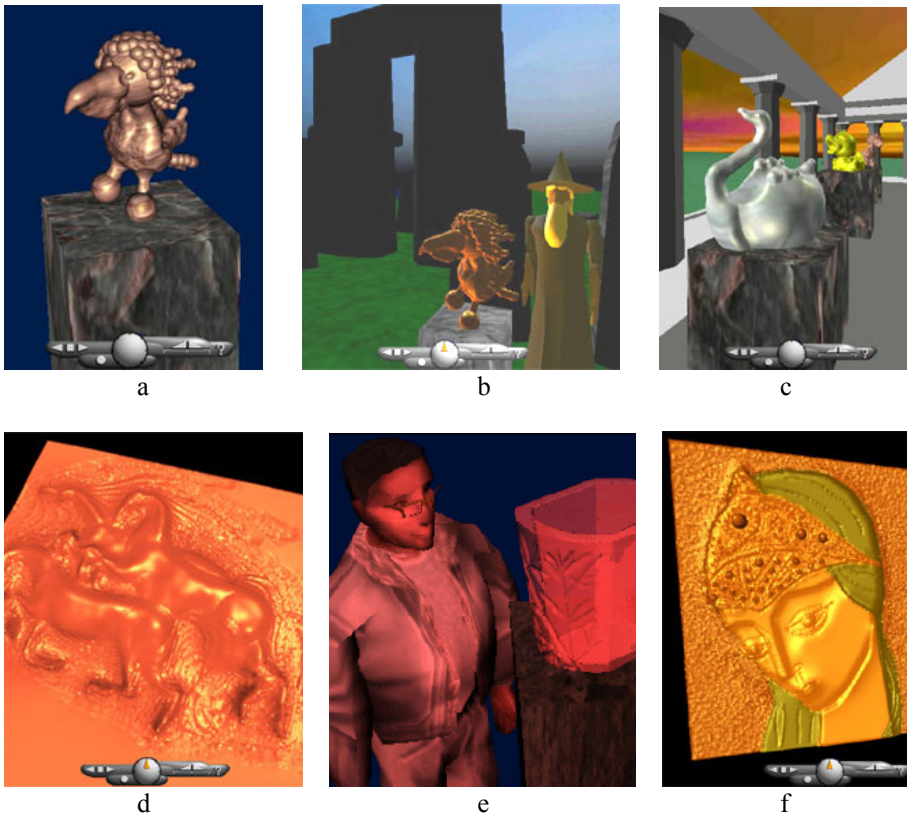


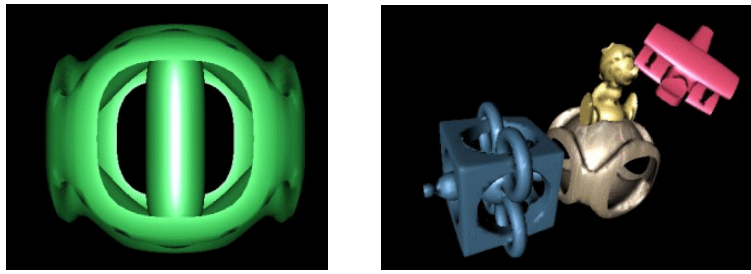
Fig 3. a-c: Examples of VRML scenes where F-Rep models are used together with VRML shapes. **d-e:** FShape node is used for visualizing very complex function-defined shapes, which polygonal equivalents consist of millions of triangles and hardly can be visualized through the Internet using conventional VRML IndexedFaceSet node.

Besides developing a generic plug-in for F-Rep defined shapes, where the model is stored in a proprietary data format and can be modified only with the respective software, we have also developed a plug-in for *HyperFun* [23]--a high-level open source modeling language designed to describe F-Rep models. We developed a plug-in supporting HyperFun to illustrate how a high-level modeling language can be incorporated into the framework of the Fshape node. In case of using HyperFun, the F-Rep model description can be included into VRML code and edited along with other VRML lines if needed.

In case of HyperFun, the integration with *FShape* node has been achieved easily since the respective Parser and Polygonizer modules are based on the original HyperFun codes kindly provided by the HyperFun developers. Only modifications to fit those modules with basic interfaces defined by *FShape* node have been required. A sample VRML and *HyperFun* definitions are given in Figure 4 for the resulting VRML shape shown in Figure 5a. In Figure 5b, several HyperFun objects are used in a VRML scene.

```
#VRML V2.0 utf8
EXTERNPROTO FShape [...]
geometry DEF model-FACES FShape {
sourceString
my_model(x[3], a[1]){
array center[3], p[3], vertex[3]; center = [0, 0, 0];
p[1] = x[1]/10; p[2] = x[2]/10; p[3] = x[3]/10;
dX = p[1]^2; dY = p[2]^2; dZ = p[3]^2;
cylz =hfCylinderZ(p,center,0.548);
cylx = hfCylinderX(p,center,0.316); cyl = cylz | cylx;
spcyl = fSphere(p,center,0.894)|cyl;
hole = hfCylinderY(p,center,0.316);
inside = spcyl \ hole; cube = 0.9-dX*dX-dY*dY-dZ*dZ;
blend = hfBlendInt(cube,inside,0.8,0.2,0.3);
my_model = blend;}
sourceURL   " "
sourceType  "hyperfun"  gridSize    [50,50,50]   }
```

Fig. 4. Using a HyperFun code for specifying a function-based model description in VRML.



a b

Fig. 5. Examples of models defined in VRML with HyperFun.

More examples as well as all the developed VRML FShape plug-ins can be freely downloaded from our project site at [28]. An alternative approach to using HyperFun in VRML based on employing a Script node and the EAI interface, as it is described in Section 3, is reported in [29].

5 Performance Analysis

Under different configurations shown in Table 1, the loading times for multiple complex objects such as the ones presented in Figure 3c are shown in Table 2. The loading time for the shapes represented using *FShape* nodes is shorter than the loading time for the shapes converted to *IndexedFaceSet* nodes. When the scene complexity increases, the FShape nodes can be exploited to greatly reduce the loading time of the scene. When the network is highly congested and the model description includes complex mathematical functions, the total time used to download the model description and to evaluate the complex mathematical functions can be quite significant. This may explain why the performance improvement in Configuration C and D is rela-

tively lesser, as compared with other configurations. The Polygonizer uses about 65 seconds on average to complete polygonization, which results in 55900 triangles. This is much better than what the conventional IndexedFaceSet node can offer, i.e. about 600 seconds on average.

Table 1. Configurations tested

Configuration	**A**	**B**	**C**	**D**
Speed (MHz)	Athlon 650	P III 733	P II 300	PII 350
Connection	56 KB/s	100 MB/s	100 MB/s	56 KB/s

Table 2. Loading times for multiple F-Rep objects from Fig. 3c.

For Multiple Objects	Representation		Rate
	IndexedFaceSet	*FShape*	
File Size (Bytes)	7938059	113686	
Loading Time for A	743 sec	54 sec	13.76
Loading Time for B	477 sec	32 sec	14.91
Loading Time for C	520 sec	76 sec	6.84
Loading Time for D	695 sec	101 sec	6.88

For the HyperFun plug-in, we used shapes from Fig. 5b and we set the *gridSize* of the *FShape* node to the same LOD as the one of the respective IndexedFaceSet. The average loading times are shown in Table 3.

Table 3. Loading times for multiple HyperFun objects from Fig. 5b.

For Multiple Objects	Representation		Rate
	IndexedFaceSet	*FShape*	
File Size (Bytes)	3056690	3052	
Loading Time for A	272 sec	18 sec	15.11
Loading Time for B	163 sec	14 sec	11.64
Loading Time for C	178 sec	16 sec	11.13
Loading Time for D	286 sec	22 sec	13.00

The loading time for HyperFun models in FShape nodes is shorter than the loading time for the respective IndexedFaceSet. The loading time includes the downloading time and the rendering time. The time difference between these two representations for all the configurations becomes greater as compared to the time difference between them for loading a single object. This means that the efficiency of using *FShape* node, as compared to using *IndexedFaceSet* node, becomes more obvious for more complex objects. Objects represented using *FShape* nodes take longer time to load for slower machine. This may be due to the complex computation routines, which require faster CPU to process the *HyperFun* scripts and execute the polygonization.

6 Further Extension

Other plug-ins for different function-based models, as well as plug-ins for other VRML browsers, are being developed now. Thus, we have recently developed a plug-in for the Visualization Toolkit (VTK) [30] and for a large class of parametric functions. The examples of using this plug-in for importing 3D Studio models into VRML scenes are given in Figure 6.



Fig. 6. Using one of the VTK classes for importing 3D Studio models into VRML scenes.

7 Conclusion and Future Work

In this paper, a generic function-based VRML geometric shape node has been described. The integration between function-defined models and VRML is proposed to implement through a VRML browser plug-in, where the customized node can be referred to as like a normal VRML node together with other conventional VRML nodes. Currently, the Blaxxun's Contact3D VRML browser has been extended to support this integration. The extension of other VRML browsers is on the way.

The use of the function-based geometric representations in VRML has open new prospects for VRML modeling and indeed improved the overall performance under the bottleneck of the Internet bandwidth. Complex geometric models with an unlimited level of detail can be easily represented with a smaller file size, as compared to the conventional polygonal based representation.

References

1 Lai Feng Min, Alexei Sourin. Function-defined shape node for VRML, Eurographics 2002, Short Presentations, ISSN 1017-4565, 2002, pp.207-215.
2 K.Engel, R.Grosso, and T.Ertl. Progressive Isosurfaces on the Web, Proc. Visualisation 98, 1998, pp. 37-40.
3 K.Engel, R.Westermann and T.Ertl. Isosurface Extraction Techniques for Web-based Volume Visualisation, Proc. IEEE Visualisation '99, 1999, pp. 139–146.
4 H-P.Seidel, H.P.A.Lensch, M.Goesele and J.Kautz. A Framework for the Acquisition, Processing, Transmission,and Interactive Display of High Quality 3D Models on the Web, Tech Report MPI-I-2001-4-002, Max-Planck-Institut fur Informatik, Germany.

5 E.Fogel, D.Cohen-Or, R.Ironi and T.Zvi. A Web Architecture for Progressive Delivery of 3D Content, Proc. On 3D technologies for the World Wide Web, Proc Virtual Reality Modeling Language Symposium, 2001, pp. 15–22.
6 G.Taubin, W.Horn, F.Lazarus, and J.Rossignac. Geometry Coding and VRML, Proc. of the IEEE, 96:6, 1998, pp. 1228-1243.
7 G.Taubin, A.Gueziec, W.Horn and F.Lazarus. Progressive Forest Split Compression, SIGGRAPH'98, 1998, pp. 123-132.
8 R.Pajarola and J.Rossignac. Compressed Progressive Meshes, IEEE Trans Vis Comput Graph, 6:1, 2000, pp. 79–93.
9 VRML, http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm.
10 M.Alexa, J.Behr and W.Müller. The Morph Node, Proc. of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language, 2000, pp. 29–34.
11 GeoVRML 1.1 Specification, http://www.geovrml.org.
12 B.Wyvill and A.Guy. The Blob Tree, Implicit Modeling and VRML, Proc. Int Conf From the Desktop to the Webtop: Virtual Environments on the Internet, WWW and Networks, NMPFT, Bradford, 1997, pp. 193–206.
13 J.J.Pittet, A.Engel and B.Heymann. Visualizing 3D Data Obtained from Microscopy on the Internet, JSB 125, 1999, pp. 123–132.
14 W.E.Lorensen and H.E.Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm, Computer Graphics, 21:4, 1987, pp. 163-169.
15 H.Grahn, T.Volk and H.J.Wolters. NURBS in VRML, Proc. of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language, 2000, pp. 35–43.
16 R.Ginis and D.Nadeau. Creating VRML Extensions to Support Scientific Visualisation, Proc. 1995 Symposium on Virtual Reality Modeling Language, 1995, pp. 13–20.
17 C.Marrin. Proposal for a VRML 2.0 Informative Annex: External Authoring Interface Reference, http://www.web3d.org/WorkingGroups/vrml-eai/history/eai_draft.html.
18 Blaxxun Contact. http://www.blaxxun.com.
19 OpenVRML. http://www.openvrml.org.
20 FreeWRL. http://freewrl.sourceforge.net.
21 F-Rep site. http://www.ntu.edu.sg/home/assourin/Frep.htm
22 A.A.Pasko, V.D.Adzhiev, A.I.Sourin, V.V.Savchenko. Function Representation in Geometric Modeling: Concepts, Implementations and Applications, Visual Comput, 11:8, 1995, pp. 429-446.
23 HyperFun site. http://www.hyperfun.org
24 A.Sourin. Functionally based virtual embossing. Visual Comput, 17, 2001, 4, pp.258-271.
25 K.Levinski and A.Sourin. Interactive polygonisation for function-based shape modelling, Eurographics 2002, Short Presentations, ISSN 1017-4565, pp.71-79, 2002.
26 Interactive Function-based Shape Modelling. http://www.ntu.edu.sg/home/assourin/Intshape.html
27 K.Levinski and A.Sourin, "Interactive function-based artistic shape modeling", 2002 International Symposium Cyber Worlds: Theory and Practice 2002, Tokyo, Japan 6-8 November, 2002 pp.521-528.
28 Function-based web visualization. http://www.ntu.edu.sg/home/assourin/FVRML.htm
29 Web-page by Jiri Zara. http://www.cgg.cvut.cz/~zara/HyperFun/
30 Visualization Toolkit. http://www.vtk.org

Planar Subdivisions by Radical Axes Applied to Structural Morphology

R. Togores, C. Otero

EGI-CAD Research Group.
Department of Geographical Engineering and Graphic Expression Techniques.
University of Cantabria. Spain.

Abstract. In previous articles the relation between Lattice and Plate structural systems to Delaunay and Voronoi planar diagrams has been demonstrated. The present contribution shows how Geotangent Mesh designs can also be formulated as a bi-dimensional problem stated as the Planar Subdivision of Radical Axes arising from a packing of circles. This way the origin of all of the Spatial Mesh Structural Typologies can be formulated by means of the basic elements of Computational Geometry.

1. Introduction

1.1. Structural Morphology

Structural morphology (see references [17] and [12]) deals with the study of the relation between geometric form and structural behavior. In a strict sense, morphology refers to the study of form, that is, the type, geometric appearance and characteristics of the surface. The concept of structure is much more ambiguous. Any kind of organized physical matter has structure and thus, *"structural morphology"* shall be associated as a characteristic to everything that possesses order, although it is also a term associated to a set of elements intended to support loads.

Factors conditioning design are many and varied and, in many occasions, are based in arrangements already proposed either in nature or in cultural tradition i.e. figures 1 and 2 compare the structural configurations from natural organisms with a well known man-made structure.

We can see how the animal's rigid shell adopts a spherical form approximated by a polyhedron with triangular or polygonal faces (figure 1). In general, these arrangements attaining a minimal amount of material or weight for each structure. Such structures do not have a dominant bi-dimensional stress-resisting element; stresses are transmitted along the whole structure in such a way that it is impossible to study the isolated parts. Their unitary character prevails during the calculus and design process. They are Spatial Structures [15]; the one of the Lattice Shell Mesh type, the other one of Plate Shell Mesh type.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 438-447, 2003.
© Springer-Verlag Berlin Heidelberg 2003

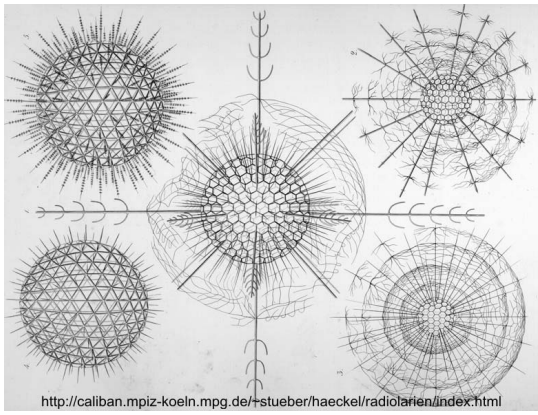


Fig. 1. The skeleton of radiolaria can be made by triangles or other type of polygons

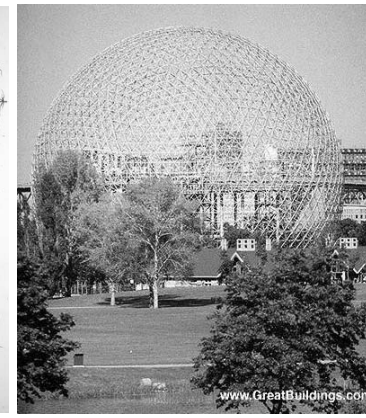


Fig. 2. Geodesic dome.

1.2. Types of Spatial Meshes

Among the possibilities for generating by means of nodes, bars and panels a spherical form, what we refer to as the Polyhedrical System [10] has received a vast amount of contributions. Essentially it deals with generating a triangular mesh over the face of a regular or semi-regular polyhedron or over the portion of the sphere [1] that is circumscribed around this polyhedron. All of the symmetry planes in a regular polyhedron produce maximum circles over its circumscribed sphere; the strong influence that these maximum circles have on the design of the triangular mesh account for the name it receives, *Geodesic Dome.*

In all of them, the procedure for generating the triangular mesh can be sketched according to these rules (more detailed in [6], [7] and [8]): (1) Selection of the initial polyhedron (usually a regular icosahedron). (2) Determination of *any triangular network* on the polyhedron's face. This network can be formulated in many ways but Coxeter's notation [2] is valid for all of them: any procedure can be formulated by means of the $\{3, 5^{+}\}_{b,c}$ notation (if we start from a regular icosahedron). (3) Projection of this network on the sphere circumscribing the polyhedron.

Spatial meshes like the one shown in figure 2, are named *Lattice structures* [8] (the basic structural element is a bar). Wester [16] has proposed an interesting alternative, suggesting that the *Plate Structure* -where plates (planar elements which are able to resist forces in their own plane only) are stabilized by shear forces- constitutes a new type and concept of structure with an applicability that is, perhaps, deeper than the former.

1.3. Spatial Meshes Made by Computational Geometry Procedures. C-TANGENT MESHES

The topological similarity between the *Plate Structures* and the *Voronoi Diagram* and between *Lattice Meshes* (geodesic domes) and a *Delaunay Triangulation* gave rise to

a whole procedure for making up a Spatial Mesh starting just from a simple set of 2D points [9]. Indeed, given the set S = { P_1, P_2, . . . , P_n} of points in the plane z=1, it results that the inversive image of the Voronoi Diagram [5], [13] of S leads to a polyhedron that approximates the sphere E $[X^2+Y^2+(Z-1/2)^2=1/4]$ in such a way that each one of its faces is tangential to the sphere. There is a symmetric correspondence between each Voronoi polygon and each face of the polyhedron; that is, the polyhedron is a Plate Structure. Similarly, the inversive image of the Delaunay Triangulation of S approximates the same sphere E as a Lattice Structure.

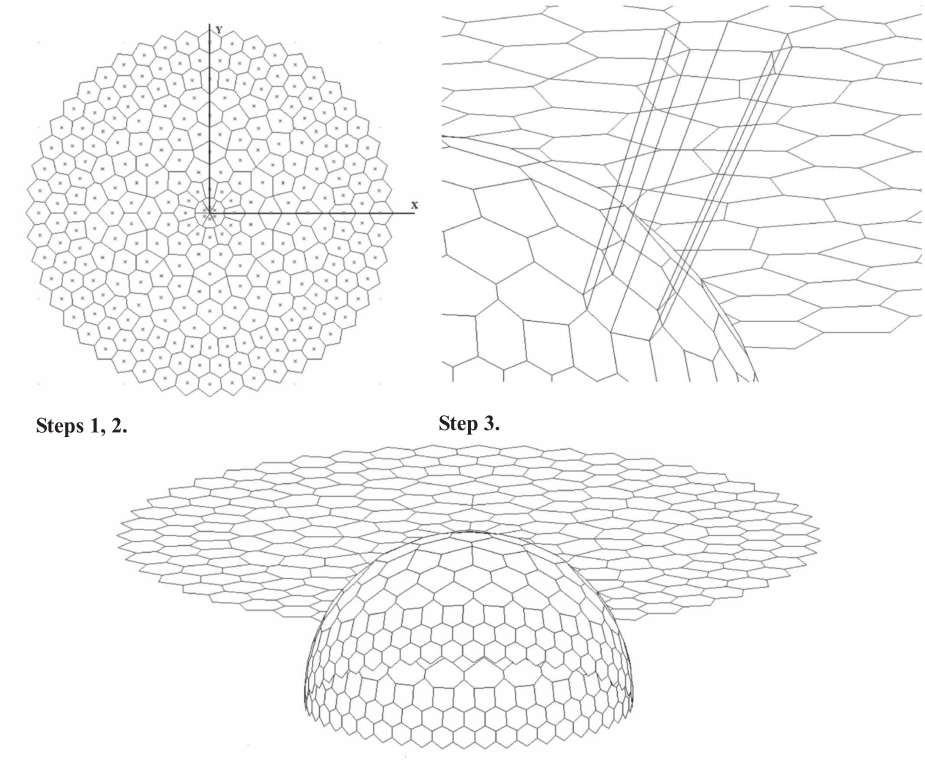


Fig. 3. A Spatial Mesh derived from a Voronoi Diagram of a set of points on z=1 is a C-Tangent Mesh.

The following procedure summarizes the way in which the property explained in the previous paragraph permits the creation of spatial meshes known as C-Tangent Meshes [11] (previously this procedure, either numeric or graphical, was always highly sophisticated):

Step 1: A set of points belonging to the plane z=1. *Step 2*: The Voronoi Diagram associated to this set is constructed. *Step 3*: The polyhedron derived from this diagram is obtained. Figure 3 describes the relation between the Voronoi edge and the polyhedron edge: the south pole of the sphere is the center of the projective transformation.

2. Geotangent Domes

A really interesting alternative procedure for the design of Spatial Meshes appeared under the name of *"Geotangent Domes"* or *"Geotangent Meshes"*, claimed by Davis and Yacoe [4], [18], which patented a polyhedrical body that approximates the sphere according to these rules: (1) It has polygonal faces, the edges of which are tangent to circles; these circles are the intersection between the plane holding the polygon and the sphere. Each vertex joins 3 or 4 polygons. (2) The first step considers a ring of hexagons (six or more) surrounding the equator circle. The next steps fill successive horizontal rings towards the sphere's north pole; each one holds new polygons (non regular) of 5, 6, 7 or 8 edges. (3) The inscribed circle within each polygon is tangent to the inscribed circles of every one of their neighbors. (4) The size of any polygon is chosen in such a way that it is the most similar possible to its neighbors in the ring immediately below. (5) In a ring, the number of polygons can be the same as the ring immediately below, half of the ring immediately below, or only one polygon (on the North Pole only). (6) When a ring reduces the number of polygons to the half of its previous ring, it is possible that a vertex joins four edges. As this is not desirable, some filler polygons can be introduced at these vertices. (7) A polyhedron made up from these propositions is presented in figure 4, left. Dimensioning such a body involves the solution of a non-linear system of equations arising from the analytical expression of the seven indicated rules. In general, we cannot say that the solution of this system is always easy, neither geometrically intuitive.

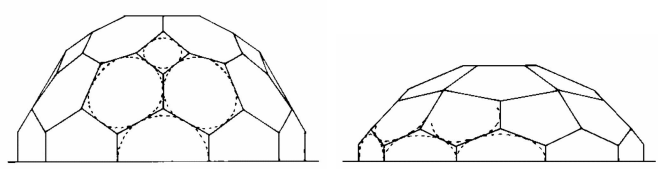


Fig. 4. Geotangent Models; left: approximating a sphere; right: approximating an ellipsoid

Two years later, Yacoe proposed, following a similar procedure, the approximation of an ellipsoid [19]. In Figure 4, right, a sketch of this solution is shown.

3. Circumsphere, Intersphere, and Insphere Associated to a Polyhedron

As it is well known, the classical study of polyhedra resorts to the association of this type of bodies with a sphere that can be inscribed, circumscribed or tangent to the edges. The polyhedron's symmetry conditions and its fundamental dimensions are closely related to each one of these spheres. Not always do these three spheres exist for a given polyhedron, but if it is a regular polyhedron, it possesses all of them. More information on this fact can be found in classical treaties on polyhedral bodies [3], [12].

Lattice and Plate Spatial Meshes correspond precisely with an inverse formulation of the relation between a sphere and a polyhedron. In fact, both cases deal with the problem of finding a polyhedron that approximates a sphere by means of planar faces. Lattice solutions generate a polyhedron with triangular faces inscribed in the sphere; Plate solutions generate a polyhedron with non-triangular faces which is circumscribed to the sphere. Does a similar formulation exist for solutions of the Geotangent type? It is evident that there is one, because as it is shown in the previous section, the edges of the approximating polyhedron are tangent to the sphere. In fact, the Geotangent model is built by means of a polyhedron with non-triangular faces whose edges are tangent to the sphere.

The generation of C-Tangent meshes [11] allows to obtain solutions of Plate type or Lattice type from a simple set of points placed in the plane z=1. If this set of points is associated by means of a Voronoi Diagram, a solution of Plate structure type is originated; if it is associated by means of a Delaunay Triangulation, a Lattice type solution is originated. Does some formulation exist in the field of Computational Geometry that will originate a Geotangent solution? Should one exist, the three spheres classically associated to a polyhedron could be formulated in a single context; on the other hand, the three spheres could be treated at a discrete level by means of sets of points in a plane.

4. Planar Subdivision by Radical Axes

Given two circumferences, their radical axis is defined [14] as the locus of the points that have the same power with respect to both. Given three circles lying in the same plane, there exists a point in this plane which has the same power with respect to the three; this point is known as the radical center of the three circles. The radical center is obtained by intersection of the radical axes of two of the three possible combinations of circles.

Let us consider three points, C_a, C_b, C_c and let us try to find three circumferences with their respective centers on those points and that are tangent to each other. As it can be seen in figure 5, left, such circumferences exist and the value of their radii is:

$$R_a = \square (d_{ab} + d_{ac} - d_{bc}) \quad R_b = \square (d_{ba} + d_{bc} - d_{ac}) \quad R_c = \square (d_{ca} + d_{cb} - d_{ab}) \qquad (1)$$

being d_{ij} the distance between the centers C_i and C_j.

The circumferences with centers in C_a, C_b, C_c and with radii R_a, R_b and R_c respectively arrange their three tangent radical axes two to two; however, as it can be seen in figure 5, right, the incorporation of a fourth point, subject to the restriction of being the center of a tangent circumference to other two can no longer be entirely random; in fact, a fourth point C_d will have to be located at distances $R_a + R_d$ and $R_b + R_d$ of centers C_a and C_b if maintaining the tangency condition is desired. Consequently, the process of positioning each new point has only one degree of freedom, not two.

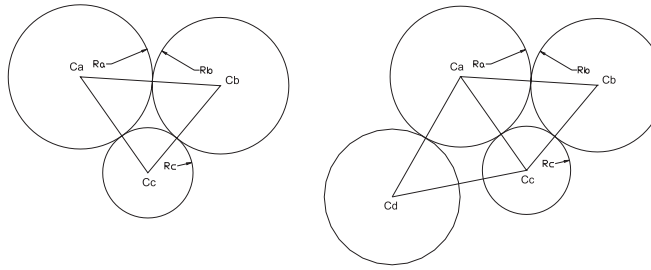


Fig. 5. A fourth point, subject to the restriction of being the center of a tangent circumference to other two can no longer be entirely random.

Nevertheless, this condition can be satisfied in sets of points that maintain a symmetry of rotation and a radial distribution. Figure 6, left, displays a distribution of this type; the points are located on concentric rings, so that: (1) two points C_a and C_b are located in the same ring of radius R_1 with an angular dephasing of value α; (2) two points C_c and C_d located in a ring of radius R_2 immediately outside implies a dephasing of angular value $\alpha/2$ relative to the previous C_a and C_b and with an angular dephasing of α between them. In such conditions it results: (1) that on points C_a and C_b two circumferences tangent to each other of radii

$$R_a = R_1 \cdot tg\ \alpha/2 \qquad (2)$$

can be located; (2) that on points C_c and C_d two circumferences tangent to each other of radii

$$R_c = R_2 \cdot tg\ \alpha/2 \qquad (3)$$

can be located; (3) that for a circumference C_c belonging to the ring of radius R_2 to be tangent to other two C_a and C_b belonging to the ring of radius R_2, the condition that distance $R_a + R_c$ = distance (C_a C_b) must be satisfied. If we name d = distance (C_a , C_b) and O the center of the circumferences of radii R1 and R_2 (O is the rotation symmetry center), it results that

$$d^2 = R_1^2 + R_2^2 - 2R_1R_2 \cos \alpha/2 \qquad (4)$$

In this description it is adequate to consider the angle α as a degree of freedom, so that the three equations shown allow us to relate the three unknown values R_a, R_c and R_2 with R_1. Manipulating these expressions the usual way we arrive to the conclusion that:

$$R_2 = R_1(\lambda + \sqrt{\lambda^2 + 1}), \text{ being } \lambda = \cos\alpha/2 \Big/ (1 - tg^2\alpha/2)$$

This way, accepting that the first ring is placed in a discretionary manner, the arrangement of tangent circumferences is univocally determined from the angle α. In figure 6, right, a configuration of tangent circumferences thus obtained is shown.

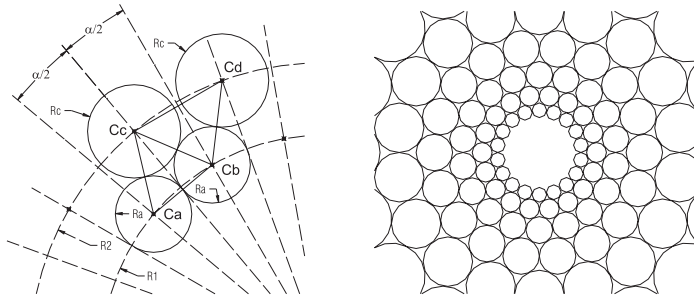


Fig. 6. The arrangement of tangent circumferences is univocally determined from angle α.

5. Geotangent Meshes Generated from a Planar Subdivision of Radical Axes

The radical axis of two tangent circumferences is their common tangent. From this simple property, a planar subdivision SD is obtained directly from any packing of tangent circumferences as the one shown, for example, in figure 6, right. The outcome is shown in figure 7.

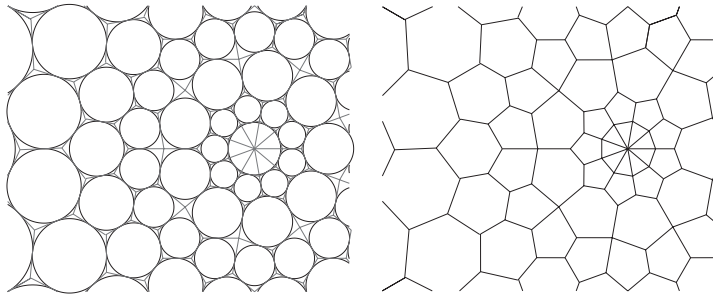


Fig. 7. A planar subdivision is obtained directly from any packing of tangent circumferences.

Let us relocate that planar subdivision SD to the z = 1 plane and let us apply an inversive transformation I with center in the origin of coordinates (0, 0, 0) and power 1, so that the points of tangency between circumferences (or what is the same, between the radical axes and the circumferences) are transformed into points of the sphere E of center in (0, 0, □) and radius □. This inversion transforms each circumference of the packing into a lesser circle of the sphere E. In addition, since the tangency is maintained in the transformation, the lesser circles stay tangent to each other. See figure 8.

Let V (see figure 8, right) represent the radical center of C_1, C_2 and C_3, point where the three edges a_1, a_2 and a_3 of SD concur. These edges are therefore, tangent to those three circumferences in points T_1, T_2 and T_3. A projective transformation TP exists that relates a_1 , a_2 and a_3 with three lines a'_1, a'_2 and a'_3 tangent to the sphere E at points T'_1, T'_2 and T'_3 , inverse of T_1, T_2 and T_3 according to the inversion I. In the

same way a_1, a_2 and a_3 concur in a point V, it results that a'_1, a'_2 and a'_3 must necessarily concur in a point V'; it is so because T'_1, T'_2 and T'_3 determine a plane α on the sphere that originates a lesser circle C_v that intersects the lines a'_1, a'_2 and a'_3. In fact a'_1, a'_2 and a'_3 are generators of the cone circumscribed to the sphere along C_v and the vertex of that cone is the pole of plane α with respect to sphere E. It is evident in the projective transformation TP that the fact that V and V' must be aligned with the origin O (0,0,0) is maintained.

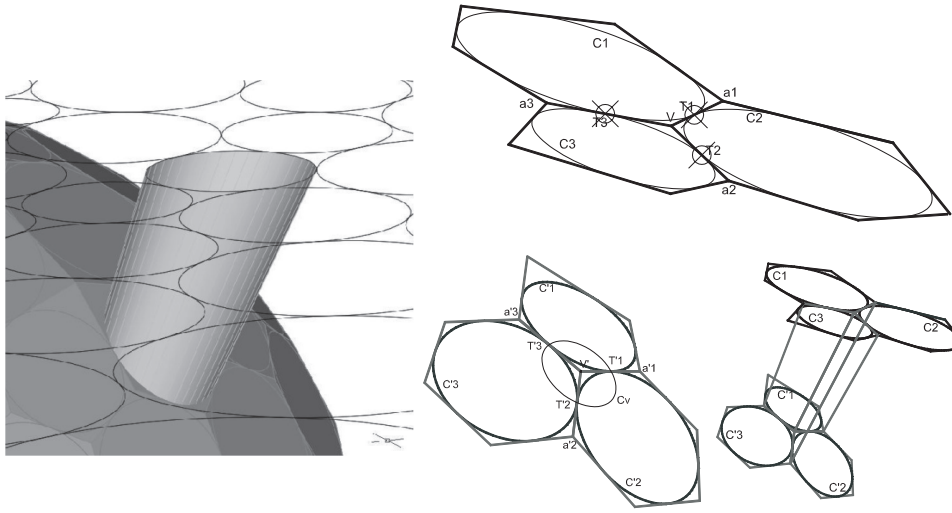


Fig. 8. Left: Circles on plane z = 1 are transformed into lesser circles of the sphere remaining tangent between themselves. Right: Relations between tangent circumferences, radical axes, points of tangency, and radical center in the z = 1 plane and on the sphere.

Transformation TP is the key to convert the planar subdivision SD into the sphere's approximating polyhedron whose faces have all of their edges tangent to the sphere. It only entails the application of the following algorithm:

PROCEDURE in order to create a polyhedron geotangent to the sphere of center in (0, 0, ½) and radius ½.
DATA: a packing EC of tangent circumferences in the z=1 plane.

1. Obtain SD: subdivision of the z=1 plane in convex polygons. The sides of these polygons are the radical axes (common tangents) to the circumferences of packing EC.

2. For each vertex V of SD; verify that V has not been processed previously; in the other case, advance to the following one; obtain the edges $a_{i\ (i=\ 1,\ 2,\ 3,\ \ldots\ )}$ concurrent in V; for each one of these edges a_i obtain T_i point of tangency with two circumferences of EC; obtain T'_i , inverse points of the T_i in inversion I.

2.1. Obtain the expression for the plane α that contains the points T'_i . Even if i is higher than 3, these T'_i points are coplanar because they come from a set of points T_i that are cocircular in the plane z=1.

2.2. Obtain the V' pole of the plane α with respect to sphere E.

3. End of 2. The proper union of the V' vertices, preserving the connectivity of the SD subdivision lines generates the Approximating Polyhedron, that turns out to be Geotangent to the sphere E. See figure 8.

The expression of inversion I specified in point 2.1 in homogenous coordinates is as follows:

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ T' \end{pmatrix} = \begin{pmatrix} 1/D^2 & 0 & 0 & 0 \\ 0 & 1/D^2 & 0 & 0 \\ 0 & 0 & 1/D^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}, \text{ where } D^2 = X^2 + Y^2 + Z^2,$$

while, in relation to point 2.2, if the plane α has the equation AX + BY + CZ+ 1 = 0, then V' has the following as its expression in homogeneous coordinates:

$$X_{v'} = -A / 2(1+C) \qquad Y_{v'} = -B/2(1+C) \qquad Z_{v'} = -1 / 1+C \qquad \textbf{(5)}$$

Conclusion

The construction of Space Meshes in the form of domes has generally pursued to achieve a polyhedron approximating a surface of double curvature. Other than exceptional cases, this surface has been the sphere.
On the one hand, the outcome of these approaches has been the subject of patents on which legally protected constructive procedures are based. Among them, the most outstanding are the Geodesic Dome [6] and the Geotangent Dome [18] and [19]. Lately, the Geotangent concept has been broadened under the idea of Panel Domes [16].

Furthermore, the three classic ways for relating a polyhedron to a sphere consist in finding a sphere that is inscribed in the polyhedron (insphere), one that is circumscribed to it (circumsphere) or one that is tangent to its edges (intersphere).

Along several publications [9], [11], we have demonstrated that these polyhedron-sphere relations are implicit in classic bidimensional configurations of Computational Geometry. In fact, if we had already demonstrated that: (1) Given a cloud of whichever points on a plane, its DIAGRAM OF VORONOI directly originates a PANEL Type Space Mesh solution. The approximate sphere is INSCRIBED in the resulting polyhedron. (2) Given a cloud of whichever points on a plane, its TRIANGULATION OF DELAUNAY directly originates a GEODESIC Type Space

Mesh solution. The approximate sphere is CIRCUMSCRIBED to the resulting polyhedron.

In this article we close this set of relations demonstrating that a PLANAR SUBDIVISION OF RADICAL AXES directly originates a GEOTANGENT Type Space Mesh solution. The approximate sphere is tangent to the edges of the resulting polyhedron.

By means of Methods belonging to Computational Geometry, a Tri-dimensional and geometrically sophisticated constructive technique is reduced to the systematic use of a profusely studied bidimensional problem. Its impact in time saving and new possibilities of design increase, as we will try to continue showing in future contributions.

References

1. Alvaro J. I., Otero C. Designing optimal spatial meshes: cutting by parallel trihedra procedure. IASS Journal (International Association for Shell and Spatial Structures) VOL. 41 (2000) N. 133.
2. Coxeter H. S. M. Regular Complex Polytopes. Ed. Cambridge University Press. 1974. 9-11.
3. Critchlow, K. Order in Space. Ed. Thames and Hudson. 1969. 75-77
4. Davis, W.E., Yacoe J., "A New Polyhedral Approximation to an Ellipsoid of Revolution" International Journal of Space Structures, Vol. 5, nr. 3 & 4. 1990.
5. Edelsbrunner H., Sheidel R. "Voronoi Diagrams and Arrangements" Discrete Computational Geometry 1, 25-44 (1986)
6. Fuller B., U.S. Patent 2,682,235 6/1954
7. Makowski, Z.S. Analysis, Design and Construction of Braced Domes. Cambridge University Press. Great Britain, 1984.
8. Margarit J., Buxad "C. Las Mallas Espaciales en Arquitectura. Ed. Gustavo Gili.1972.
9. Otero C., Gil V., Alvaro J. I. CR-Tangent Meshes. IASS Journal VOL. 41 (2000) n. 132. 41-48.
10. Otero C. Dise"o Geom"trico de C"pulas no esf"ricas aproximadas por mallas triangulares con un n"mero m"nimo de longitudes de barra. Ph D. University of Cantabria. 1990. Not available, contact the author.
11. Otero C. , Togores R. "Computational Geometry and Spatial Meshes" Lecture Notes On Computer Science 2002 . Vol. 2. Springer. 2002.
12. Pearce P. Structure in Nature is a Strategy for Design. Ed. Cambridge University Press. 1974
13. Preparata F, Shamos I. "Computational Geometry: An Introduction" Springer. 1985. 244-247.
14. Pedoe, D. "Geometry: a comprehensive course" Cambridge Univ. Press. 1970. 71-121.
15. Tsuboi Y. Analysis, design and realization of space frames. (Working Group of Spatial Steel Structures). IASS Bulletin, N"84. April 1984 and N"96, April 1988. 11-30.
16. Wester, T. "A Geodesic Dome-Type Based on Pure Plate Action" International Journal of Space Structures, Vol. 5, nr. 3 & 4. 1990. 155-167
17. Wester, T. "The Structural Morphology of Basic Polyhedra", Chapter 11 in "Beyond The Cube", pp. 301-342. John Wiley & Sons. 1997. 301-342
18. Yacoe J. U.S. Patent 4,679,361 7/1987
19. Yacoe J. U.S. Patent 4,825602 5/1989

Distribution of Vertex Indices in Edgebreaker

Youngsong Cho[1], Deok-Soo Kim[1], Hyun Kim[2], Hyun Chan Lee[3], and Joon Young Park[4]

[1] Department of Industrial Engineering, Hanyang University,
17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791, South Korea
ycho1971@ihanyang.ac.kr, dskim@hanyang.ac.kr
[2] Concurrent Engineering Research Team,
Electronics and Telecommunications Research Institute,
161 Gajeon-dong, Yuseong-gu, Daejeon, 305-350, South Korea
hyunkim@etri.re.kr
[3] Department of Industrial Engineering, Hongik University,
72-1, Sangsu-dong, Mapo-gu, Seoul, 121-791, Korea
hclee@wow.hongik.ac.kr
[4] Department of Industrial Engineering, Dongguk University,
26, 3-ga, Pil-dong, Chung-gu, Seoul, 100-715, South Korea
jypark@dgu.edu

Abstract. As the use of Internet has become an ordinary activity, every kind of file is transmitted through network and shape models are not an exception at all. Hence, the compression of topology, geometry, etc. has been studied since the initial work by Deering. Since the compression of geometry and other attributes of mesh model turns out to use the information produced by topology compression, the topology compressor such as Edgebreaker has been extensively analyzed. Presented in this paper is another important characteristic of Edgebreaker: the distribution of vertex indices. The distribution is very important to achieve higher compression ratio of shape models with attributes defined at vertices, especially when there are multiple attributes at each vertex.

1 Introduction

As the use of Internet has become an ordinary activity, almost every kind of file is transmitted through networks. For example, downloading text files, sound files, movies, etc. are common practices in an everyday life. The transmission of shape models is not an exception at all and there have been several researches for the various aspects of shape models for rapid transmission [1], [2], [3], [4], [5], [7], [8], [9], [10], [11], [12].

Among several important parts of shape model, topology compression has been started earlier. While researches on topology has been focused in the lossless compression, the achievements in the compression of other parts of shape model, such as vertices or normal vectors, are generally lossy compression to achieve higher compression ratio. It turns out that the compression of vertices and normal vectors take advantage of the vertex sequence information produced

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 448–457, 2003.
© Springer-Verlag Berlin Heidelberg 2003

by topology compression. It is important to know the performance characteristics of topology compression algorithms. Among several algorithms, Edgebreaker by Rossignac has been known to be efficient for both compression and decompression. Besides, the performance characteristics of Edgebreaker have been well analyzed rigorously [9].

In this paper, we present another important characteristic of Edgebreaker: the distribution of vertex indices. The analysis of this distribution turns out to be very important to achieve higher compression ratio of shape models with attributes, such as normal vectors and colors, defined at vertices. It is especially important when there are multiple attributes at each vertex and each face sharing the vertex uses different attribute instance at the vertex.

Fig. 1 shows the use of multiple normal vectors at some vertices of mesh models. Shown in Fig. 1(a) and (c) are a bolt and a fan disk models, respectively, where the vertices at sharp corners have multiple normal vectors so that the angles at these corners remain in their visual appearances. On the other hand, Fig. 1(b) and (d) are same models with only one normal vector defined at each vertex. Note that the sharp corners are not visible at all. Fan disk model even misleads by the blending-like appearance.

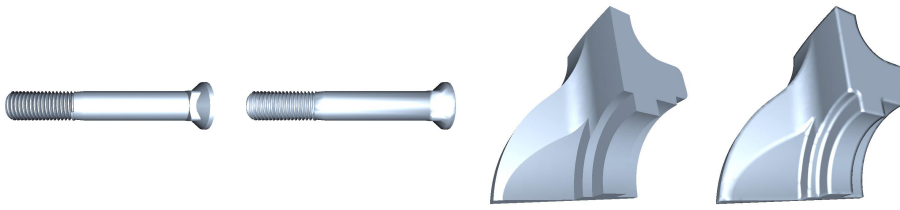


Fig. 1. Mesh models. (a) a bolt model with multiple normals per vertex, (b) a bolt model with a unique normal per vertex, (c) a fan disk model with multiple normals per vertex, and (d) a fan disk model with a unique normal per vertex

In this paper, we are presenting the distribution of vertex indices simply because the word *vertex* can be replaced by the word *normal vector* or *color* depending on the applications. Note that Edgebreaker itself does not require any information on this distribution. The details of the compression issue of normal vectors for mesh models are discussed in [6]. Shape models discussed in this paper are assumed to be simple meshes.

2 Brief of Edgebreaker

Including VRML, there are various representations of mesh models and the minimum information needed in the representation is the coordinates of vertices and the topology among them. Depending on the requirements in applications,

other information such as normal vectors, colors, etc., is also stored in the model. The relationship between vertices and topology can be either explicit or implicit. In VRML, for example, the topology is defined as consecutive three integer indices of a particular triangular face and an index points to an appropriate vertex that is predefined as consecutive three float numbers.

We review Edgebreaker briefly here since we are interested in its performance characteristic: the distribution of vertex indices. Edgebreaker compresses the topology of a mesh model by classifying a triangle into one of five op-codes, C, R, E, S, and L, depending on the states of two triangles. Hence, Edgebreaker transforms a mesh model into a string consisting of five alphabets [7]. Through experiments on several large models of up to 200,000 triangles, Rossignac reported the relative frequencies, or probabilities, of these five alphabets as P(C) = 0.5, P(R) = 0.363, P(E) = 0.056, P(S) = 0.056, and P(L) = 0.025 [7].

For the convenience of presentation in this article, the following notions are introduced via Fig. 2. When a triangle X is entered from another triangle through an edge E_G, the edge is called a *gate* as named by Edgebreaker. When the vertices of X are referenced, the order of the vertices is in CCW direction as the following: the left vertex of gate v_1, the right vertex of gate v_2, and the other vertex v_3. The edge connecting v_1 and v_3 is called a left edge E_L of X, and the edge connecting v_2 and v_3 is called a right edge E_R of X. The triangle over E_L is called a left face F_L, and the one over E_R is called a right face F_R.

Suppose that the right face F_R is being parsed to be compressed after X was compressed. Then, we call F_R a current face *CF* and X a previous face *PF*, respectively. When a face is of type S, in Edgebreaker notion, we call the face *SF* for short.

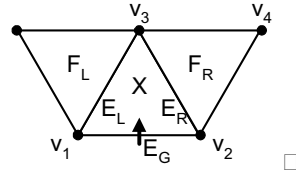


Fig. 2. Notions for topology in Edgebreaker

3 Representation of Face-Vertex Relations

Consider a small mesh model, shown in Fig. 3(a), that has three faces f_1, f_2, and f_3, and five vertices v_1 through v_5. Assuming that vertices are defined in a counterclockwise orientation around a face, let f_1 be defined in the order of v_1, v_3, and v_4. Similarly, f_2 is defined by v_1, v_4, and v_2, and f_3 by v_2, v_4, and v_5. If the face sequence produced by Edgebreaker is f_1, f_2, and f_3, then the whole face-vertex relation can be represented by an array as shown in Fig. 3(b). Note that the subscripts are denoted by the integers in the array.

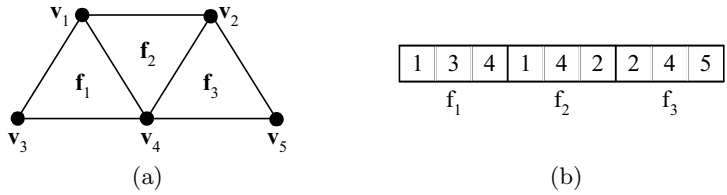


Fig. 3. Face-vertex relation. (a) a mesh model, (b) a vertex index array

When it is necessary to store this array for some reason, the array takes 3F indices, where F is the number of faces. If indices are in integer form, the array takes 12F bytes. If indices are in a more compact format, an index may be shortened as few as $\lceil \log_2 V \rceil$, where V is the number of vertices. However, the discussions in the next section can help to make the representation of this array further compact.

4 Analysis of Index Array

It turns out that a particular index value in the index array tends to appear quite shortly after the index value was used. Fig. 4(a), for example, is the array given in Fig. 3(b). with recurring indices marked with arrows. In this example, it is shown that the currently recurring index is not very far from the latest occurrence of the same index value. Let an *index distance D* be the relative distance between the currently recurring index and the latest occurrence of the same index.

If the recurring indices are replaced with D, it may take fewer bits than using the absolute values of the indices as shown in Fig. 4(b). Note that the bit saving by employing the mixed use of absolute and relative values for indices, *mixed indexing scheme*, depends on the distribution of index distances, and therefore, the investigation on the distribution of D is necessary. In this paper, we are particularly interested in analyzing the distribution of D for values of 1, 2, 3, 4, and higher than 4.

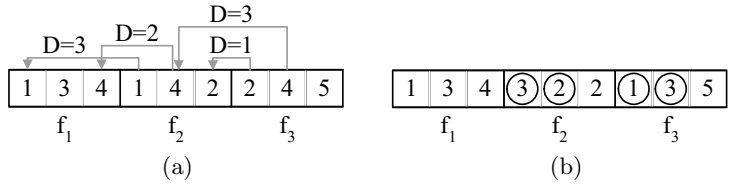


Fig. 4. Absolute and relative indices. (a) absolute indices, (b) mixed indices

Shown in Fig. 5 is the frequency distribution of new and recurring indices for several mesh models. The models tested for this and the following experiments

are downloaded from other websites and shown in Fig. 6. As shown in the figure, approximately 80 % of the indices are recurring and only 20 % of the indices are new.

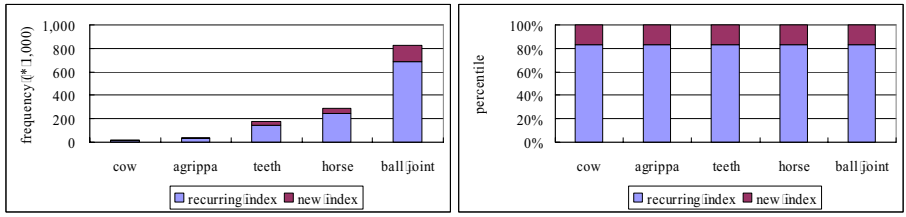


Fig. 5. Frequency distribution of new and recurring indices: frequency(left), percentile(right)

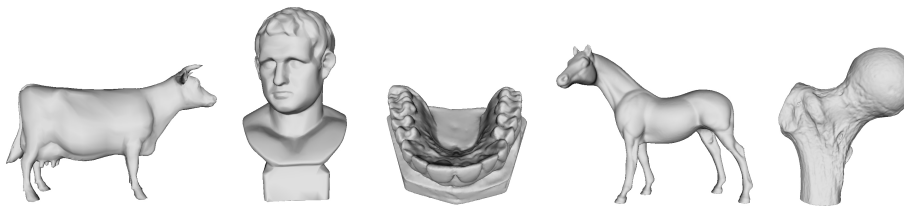


Fig. 6. Mesh models tested. (a) cow (5,804 faces), (b) agrippa (12,660 faces), (c) teeth (58,300 faces), (d) horse (96,966 faces), (e) ball joint (274,120 faces)

To analyze the distribution of indices in the array, we rearranged the mesh configurations of Edgebreaker. CF and PF may be connected through a vertex or an edge. In some cases, however, CF and PF may not be directly connected. When CF and PF are connected through an edge, they are called *edge-connected*. Similarly, it is called *vertex-connected* if there is a vertex shared by both CF and PF. When CF and PF are separate, it is called *disconnected*. Fig. 7 illustrates examples for these cases: Fig. 7(a) are edge-connected cases, and Fig. 7(b) is a vertex-connected case. In the example of Fig. 7(c), there are two intermediate faces separating CF from PF.

4.1 Edge-Connected Cases

Shown in Fig. 8 are edge-connected cases denoted as *Econ*. Example in Fig. 8(a) is a case when PF is of type C, S, or L. In this case, CF is always F_R of PF by sharing an edge by Edgebreaker rule and the corresponding index array is always

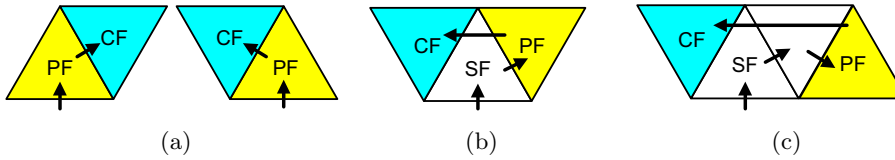


Fig. 7. Mesh configurations. (a) edge-connected case, Econ, (b) vertex-connected case, Vcon, (c) disconnected case, Dcon

given as shown in the right of mesh configuration. Note that, in the index array, i th element is always the index to the first vertex of CF and this index should always point to the last vertex of PF which is i-1 th element in the array. Hence, in these cases, i th element is always identical to i-1 th element and therefore the index distance of the first vertex of CF is 1. Similarly, i+1 th element is always identical to i-2 th element and the index distance of the second index of CF is always 3. On the other hand, i+2 th element for CF does not have any index with identical value in PF. Note that index distances of 2 or 4 cannot occur at all.

When PF is of type R as shown in Fig. 8(b), the index array becomes as shown in its right. In this case, the index distance values of 2 and 3 are only possible by the similar argument as the above. These two cases are the only possible configurations that CF and PF share an edge.

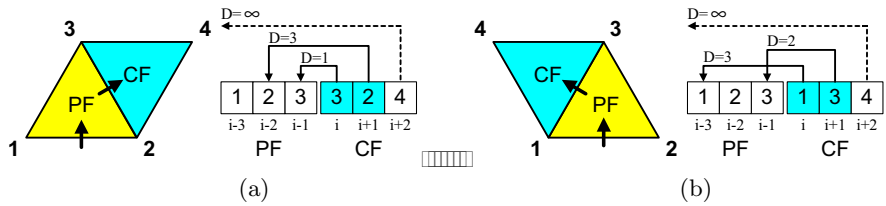


Fig. 8. Mesh configurations and index arrays of edge-connected cases. (a) Op-codes of C, L, and S, (b) Op-code of R

4.2 Disconnected Cases

Fig. 9 shows a disconnected case denoted as *Dcon*. Note that there should be at least two faces in-between CF and PF and these two faces are edge-connected. Similarly to vertex sharing cases that will be elaborated shortly, this case occurs when PF was arrived after an SF, a face of type S. PF should be always of type E and CF is a face popped from a stack. In this case, the minimum index distance is always greater than 4.

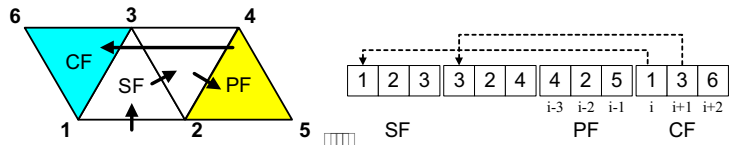


Fig. 9. Disconnected case: mesh configuration and index array

4.3 Vertex-Connected Cases

On the other hand, a general case that CF shares a vertex with PF in common so that they form a vertex-connected case denoted as *Vcon*. If this case occurs as illustrated in Fig. 10, there should be a SF similarly to a disconnected case and SF shares its left edge with CF. In addition, SF, PF and CF shares a vertex in common. Since PF should be of type E in this case, Edgebreaker pops a face from a stack and the process jumps to CF.

Let α, β, and γ be three edges of PF as shown in the figure. Since a gate to PF can be either one of the three edges, there can be different path sets leading to α, β, and γ gates of PF and accordingly there are three different representation sets of index array depending on the paths to the gates of PF.

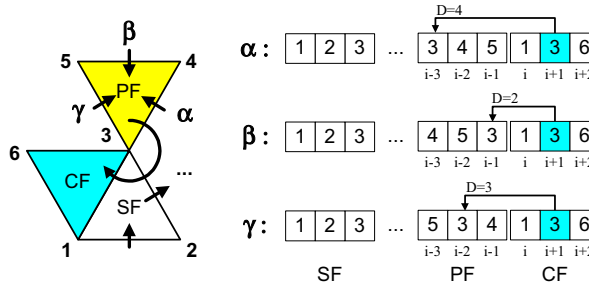


Fig. 10. Vertex-connected cases: mesh configurations and index arrays

Case α is a case that the edge α is used as the gate of PF. Similarly, Case β and Case γ can be defined. Even though there can be several paths from SF to PF in Case α, the resulting index array between CF and PF remains similar as shown in the figure. In this case, the second vertex of CF has always index distance of 4 while the other two indices have distances higher than 4. Note that the number of intermediate faces between SF and PF does not affect the relationship between CF and PF in index array. Similarly, in Case β, the second index of CF has an index distance 2 and the other vertices have distances higher than 4. In Case γ, the middle index of CF has always index distance of 3 and the other two indices have distances higher than 4.

5 Probability Distributions

5.1 Probabilities of Face Configurations

Assuming that the probability distribution of op-codes C, R, E, S, and L is known a priori and are independent,

$$P(Econ) = P(C) + P(S) + P(L) + P(R) \ . \tag{1}$$

since the edge-connected cases can happen when PF is of type either C, S, L, or R, where P(X) denotes the probability of X. On the other hand, P(Vcon), which cannot be known, has to be estimated in a reasonable fashion. Since PF has three possible gates, α, β, and γ, all of the paths from SF to PF can be grouped into three groups passing through each gate. If P(α), P(β), and P(γ) can be estimated, it holds that

$$P(Vcon) = P(\alpha) + P(\beta) + P(\gamma) \ . \tag{2}$$

The exact probability of disconnected case cannot be known immediately as well. However, this case occurs only when there is a SF and PF is of type E. Since vertex-connected cases can also happen in this mesh configuration,

$$P(S) = P(E) = P(Vcon) + P(Dcon) \ . \tag{3}$$

Let a path π be a sequence of faces between two particular faces including both faces. Then, $P_\alpha(\pi)$ denotes the probability of path π which enters through the gate α. If a path π_1 is a path with maximum probability among all possible paths leading to the gate α, we call $P_\alpha(\pi_1)$ the *majorizing* probability of the gate α. In other words, the example given in Fig. 7(b) gives $P_\alpha(\pi_1)$ which majorizes the probabilities of all cases to arrive gate α, and therefore $P_\alpha(\pi_1)$ can be safely interpreted as a lower-bound of $P(\alpha)$. $P_\beta(\pi_1)$, $P_\gamma(\pi_1)$ and $P_{Dcon}(\pi_1)$ are defined similarly. Therefore,

$$\begin{aligned} P(E) &= P(S) \\ P(E) &= P(Vcon) + P(Dcon) \\ P(Vcon) &= P(\alpha) + P(\beta) + P(\gamma) \\ P(C) + P(R) + P(E) + P(S) + P(L) &= 1 \ . \end{aligned} \tag{4}$$

Hence, we have to estimate the probabilities of $P(\alpha)$, $P(\beta)$, $P(\gamma)$ and $P(Dcon)$ using known probability distributions. Among several ways, we have chosen to allocate P(E) to $P(\alpha)$, $P(\beta)$, $P(\gamma)$ and $P(Dcon)$ proportionally to the ratios of the majorizing probabilities $P_\alpha(\pi_1)$, $P_\beta(\pi_1)$, $P_\gamma(\pi_1)$, and $P_{Dcon}(\pi_1)$.

5.2 Probabilities of Index Distances

When PF is of type C, S, or L, the index array is as shown in Fig. 8(a) and $D_i = 1$, $D_{i+1} = 3$, and $D_{i+2} = \infty$. Let P(CSL) = P(C) + P(S) + P(L). Then, P(D=1

| PF=CSL) = $\frac{1}{3}$, P(D=3 | PF=CSL) = $\frac{1}{3}$, P(D=∞ | PF=CSL) = $\frac{1}{3}$, and P(D=2 | PF=CSL) = P(D=4 | PF=CSL) = 0. Therefore, it can be shown that

$$P(D = 1 \cap PF = CSL) = P(D = 1 \mid PF = CSL)P(CSL) = \tfrac{1}{3}P(CSL)\ . \quad (5)$$

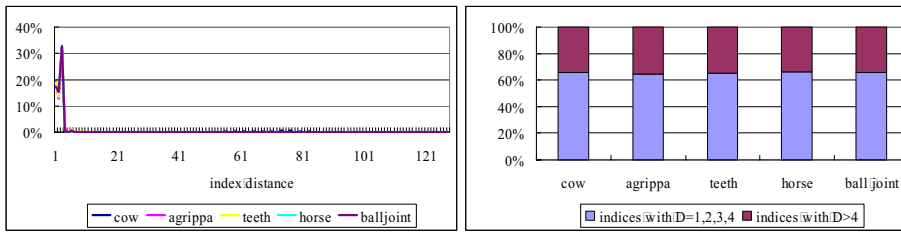
Similarly, P(D=3 $\cap$ PF=CSL) = P(D=∞ $\cap$ PF=CSL) = $\frac{1}{3}$P(CSL), and P(D=2 $\cap$ PF=CSL) = P(D=4 $\cap$ PF=CSL) = 0. When the op-code for PF is R, similar observation can be made and P(D=2 $\cap$ PF=R) = P(D=3 $\cap$ PF=R) = P(D=∞ $\cap$ PF=R) = $\frac{1}{3}$P(R), and P(D=1 $\cap$ PF=R) = P(D=4 $\cap$ PF=R) = 0.

In the vertex-connected case, op-code of PF should be always E. In this case, the second vertex of CF only has a relative index value smaller than five. In the case of gate α, it turns out that there can be only two values of index distances: D=4 or D=∞, since it is always guaranteed that D_i=∞, D_{i+1}=4, and $D_{i+2} = \infty$. If gate β, similar observation yields D=2 or D=∞ since $D_i = \infty$, D_{i+1}=2, and $D_{i+2} = \infty$. If gate γ, $D_i = \infty$, D_{i+1}=3, and $D_{i+2} = \infty$ and therefore it can be deduced that D=3 or ∞. Hence, the index distance which is not infinity always only depends on the gate to PF, and can have value of either 2, 3, 4 or ∞.

Let's now compute the probability of D=2. Since D=2 can only occur when the gate β is used to enter to PF of type E, it is necessary to know $P(\beta)$. Therefore, P(D=2 $\cap$ Connectivity=Vcon) = P(D=2 $\cap$ Gate=β) = P(D=2 | Gate=β)P(β) = $\frac{1}{3}$P(β). Similarly, P(D=3 $\cap$ Connectivity=Vcon) = $\frac{1}{3}$P(γ), P(D=4 $\cap$ Connectivity=Vcon) = $\frac{1}{3}$P(α) and P(D=1 $\cap$ Connectivity=Vcon) = 0. In addition, it can be shown that P(D=∞ $\cap$ Connectivity=Vcon) = $\frac{2}{3}$P(Vcon).

In the disconnected case, on the other hand, there is no relative index with distance less than 5. All three indices will have distances of ∞.

Shown in Fig. 11 is distribution of index distances. Fig. 11(a) shows the frequency distribution of recurring indices of the mesh models with respect to index distances. Note that the distribution shows a wide spectrum, highly skewed to the left with the others having only very few data items. Fig. 11(b) shows the distributions for D is 1 through 4 and higher than 4. As shown in the figure, approximately 60 % through 65 % of the whole indices has values less than 4 as their distances. Note that initially occurring indices are considered as indices with index distances higher than 4.



(a) (b)

Fig. 11. Distribution of index distances (percentile). (a) frequency distribution, (b) distribution for D with 1 through 4 and higher than 4

6 Conclusions

Since compression is one of the core technologies for a seamless transmission through Internet, the compression of topology, geometry, etc. for 3D shape models have been extensively studied. In this paper, we have presented an analysis of index distance distributions of Edgebreaker which can be very useful for the compression of attributes defined at vertices. Especially when there are multiple attribute values at each vertex, it can be used for higher compression ratio.

Acknowledgments

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center(CPRC) at Hanyang University.

References

1. Bajaj, C.L., Pascucci, V., Zhuang, G.: Single resolution compression of arbitrary triangular meshes with properties. Computational Geometry: Theory and Application, Vol. 14. (1999) 167–186
2. Chow, M.: Optimized Geometry Compression for Real-Time Rendering. Proceedings of IEEE Visualization '97. (1997) 347–354
3. Deering, M.: Geometry Compression. Proceedings of ACM SIGGRAPH' 95. (1995) 13–20
4. Gumhold, S., Strasser, W.: Real-Time Compression of Triangle Mesh Connectivity. Proceedings of ACM SIGGRAPH '98. (1998) 133–140
5. Isenburg, M., Snoeyink, J.: Spirale Reversi: Reverse decoding of Edgebreaker encoding. Computational Geometry: Theory and Application, Vol. 20. (2001) 39–52
6. Kim, D.-S., Cho, Y., Kim, D.: The Compression of the Normal Vectors of 3D Mesh Models Using Clustring. Lecture Notes in Computer Science, Vol. 2330. Springer-Verlag, Berlin Heidelberg New York (2002) 275–284
7. Rossignac, J.: Edgebreaker: Connectivity Compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics, Vol. 5. (1999) 47–61
8. Rossignac, J., Szymczak, A.: Wrap & Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker. Computational Geometry: Theory and Application, Vol. 14. (1999) 119–135
9. Szymczak, A., King, D., Rossignac, J.: An Edgebreaker-based efficient compression scheme for regular Meshes. Computational Geometry: Theory and Application, Vol. 20. (2001) 53–68
10. Taubin, G., Horn, W.P., Lazarus, F., Rossignac, J.: Geometric Coding and VRML. Proceedings of the IEEE, Vol. 86. (1998) 1228 –1243
11. Taubin, G., Rossignac, J.: Geometric Compression Through Topological Surgery. ACM Transactions on Graphics, Vol. 17 . (1998) 84–115
12. Touma, C., Gotsman, C.: Triangle Mesh Compression. Proceedings of Graphics Interface '98. (1998) 26–34

A Comparison Study of Modern Heuristics for Solving the Partitioning Problem in Distributed Virtual Environment Systems

P. Morillo, M. Fernández, and J.M. Orduña

[1] Instituto de Robótica. Universidad de Valencia. SPAIN.
Pedro.Morillo@uv.es
[2] Departamento de Informática. Universidad de Valencia. SPAIN
Juan.Orduna@uv.es

Abstract. Fast Internet connections and the widespread use of high performance graphic cards are making Distributed Virtual Environment (DVE) systems very common nowadays. However, there are several key issues in these systems that should still be improved in order to design a scalable and cost-effective system. One of these key issues is the partitioning problem. This problem consists of efficiently assigning clients (3-D avatars) to the servers in the system.
In this paper, we present a comparison study of different modern heuristics for solving the partitioning problem in DVE systems, as an alternative to the ad-hoc heuristic proposed in the literature. Performance evaluation results show that some of the heuristic methods can greatly improve the performance of the partitioning method, particularly for large DVE systems. In this way, efficiency and scalability of DVE systems can be significantly improved.

1 Introduction

The widespread use of both fast Internet connections and also high performance graphic cards have made possible the current growth of Distributed Virtual Environment (DVE) systems. These systems allow multiple users, working on different computers that are interconnected through different networks (and even through Internet) to interact in a shared virtual world. This is achieved by rendering images of the environment as if they were perceived by the user. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user input. Since DVE systems support visual interactions between multiple avatars, every change in each avatar must be propagated to the rest of the avatars in the shared virtual environment.

DVE systems are currently used in different applications [17], such as collaborative design [16], civil and military distributed training [14], e-learning [15] or multi-player games [11]. Due to their different requirements, communication rate of avatars may significantly differ among these different applications. One of the key issues in the design of a scalable DVE system is the *partitioning problem* [12]. It consists of efficiently assigning the workload (avatars) among different servers in the system. The partitioning problem may seriously affect the overall performance of the DVE system, since it determines not only the workload each server is assigned to, but also the inter-server communication requirements (and therefore the network traffic). Some methods for solving

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 458–467, 2003.
© Springer-Verlag Berlin Heidelberg 2003

the partitioning problems have been already proposed [12,13,17]. These methods provide efficient solutions even for large scale DVE systems. However, there are still some features in the proposed methods that can be improved. For example, different heuristic search methods can be used for finding the best assignment of clients to servers, instead of using ad-hoc heuristics.

In this paper, we present a comparison study of several heuristics for solving the partitioning problem in DVE systems. We have implemented five different heuristics, ranging over most of the current taxonomy of heuristics: Genetic Algorithms (GA) [8], two different implementations of Simulated Annealing [10], Ant Colony Systems (ACS) [4], and Greedy Randomized Adaptive Search (GRASP) [3]. Performance evaluation results show that the execution cost of the partitioning algorithm (in terms of execution times) can be dramatically reduced, while providing similar or even better solutions than the current best solutions, provided by the ad-hoc heuristic proposed in [13]. We plan to use these results to improve the efficiency and scalability of our DVE system, a collaborative driving simulator [7].

The rest of the paper is organized as follows: Section 2 describes the partitioning problem and the existing proposals for solving it. Section 3 describes the proposed implementations of the heuristics considered for this study. Next, Section 4 presents the performance evaluation of the proposed heuristics. Finally, Section 5 presents some concluding remarks.

2 The Partitioning Problem in DVE Systems

Architectures based on networked servers are becoming a de-facto standard for DVE systems [17,12]. In these architectures, the control of the simulation relies on several interconnected servers. Multi-platform client computers must be attached to one of these servers. When a client modifies an avatar, it also sends an updating message to its server, that in turn must propagate this message to other servers and clients. Servers must render different 3D models, perform positional updates of avatars and transfer control information among different clients. Thus, each new avatar represents an increasing in both the computational requirements of the application and also in the amount of network traffic. When the number of connected clients increases, the number of updating messages must be limited in order to avoid avoid a message outburst. In this sense, concepts like areas of influence (AOI) [17], locales [1] or auras [9] have been proposed for limiting the number of neighboring avatars that a given avatar must communicate with.

Depending on their origin and destination avatars, messages in a DVE system can be intra-server or inter-server messages. In order to design a scalable DVE systems, the number of intra-server messages must be maximized. Effectively, when clients send intra-server messages they only concern a single server. Therefore, they are minimizing the computing, storage and communication requirements for maintaining a consistent state of the avatars in a DVE system. Lui and Chan have shown the key role of finding a good assignment of clients to servers in order to ensure both a good frame rate and a minimum network traffic in DVE systems [12,13]. They propose a quality function, denoted as C_p, for evaluating each assignment of clients to servers. This quality function takes into account two parameters. One of them consists of the computing workload

generated by clients in the DVE system, denoted as C_p^W. In order to minimize this parameter, the computing workload should be proportionally shared among all the servers in the DVE system, according to the computing resources of each server. The other parameter of the quality function consists of the overall inter-server communication cost, denoted as C_p^L. In order to minimize this parameter, avatars sharing the same AOI should be assigned to the same server. Quality function C_p is defined as

$$C_p = W_1 \, C_p^W + W_2 \, C_p^L \tag{1}$$

where $W_1 + W_2 = 1$. W_1 and W_2 are two coefficients that weight the relative importance of the computational and communication workload, respectively. These coefficients should be tuned according to the specific features of each DVE system. Using this quality function (and assuming $W_1 = W_2 = 0.5$) Lui and Chan propose a partitioning algorithm that re-assigns clients to servers [13]. The partitioning algorithm should be periodically executed for adapting the partition to the current state of the DVE system as it evolves (avatars can join or leave the DVE system at any time, and they can also move everywhere within the simulated virtual world). Lui and Chan also have proposed a testing platform for the performance evaluation of DVE systems, as well as a parallelization of the partitioning algorithm [13].

Some other approaches for solving the partitioning problem have been also proposed. One of them groups avatars following regular distributions [2]. In order to ensure good performance, this algorithm generate a number of regular distributions equal to the number of servers in the DVE system. However, this proposal does not obtain good performance when avatars are located following a non-uniform distribution. Another different approach rejects dynamic concepts associated to avatars like AOI, aura or locale [18]. Although this approach provides a fast way of solving the partitioning problem, the performance of the static partitioning is quite low when avatars show a clustered distribution. In this case, the servers controlling the areas of the clusters are overloaded, increasing the overall cost of the quality function.

The partitioning method proposed by Lui and Chan currently provides the best results for DVE systems. However, it uses an ad-hoc heuristic. We propose a comparative study of several heuristics, ranging over most of the current taxonomy of heuristics, in order to determine which one provides the best performance when applied to the partitioning problem. In this study we will follow the same approach of Lui-Chan: using the same quality function, we will obtain an initial partition (assignment) of avatars to servers, and then we will test the implementation of each heuristic to provide a near optimal assignment.

3 Heuristic Descriptions

In this section, we present five implementations of different heuristics for solving the partitioning problem in DVE systems. Following the approach presented by Lui and Chan [13] (and using the same quality function C_p), the idea is dynamically applying a heuristic search method that provides a good assignment of clients to servers as the state of the DVE system changes. In this section, we describe the implementation of each

heuristic search method and the tuning of its parameters for solving the partitioning problem.

All of the implemented heuristics start from an initial partition (assignment) of avatars. We tested several clustering algorithms for obtaining this initial partition. Although they are not shown here due to space limitations, we obtained the best results for a *density-based algorithm* (DBA) [5]. This algorithm divides the virtual 3D scene in square sections. Each section is labeled with the number of avatars that it contains (na), and all the sections are sorted (using Quick-sort algorithm) by their na value. The first S sections in the sorted list are then selected and assigned to a given server, where S is the number of servers in the DVE system. That is, all the avatars in a selected region are assigned to a single server. The next step consists of computing the mass-center (mc) of the avatars assigned to each server. Using a round-robin scheme, the algorithm then chooses the closest free avatar to the mc of each server, assigning that avatar to that server, until all avatars are assigned. Since the assignment of avatars follows a round-robin scheme, this algorithm provides a good balancing of the computing workload (the number of avatars assigned to each server does not differ in more than one). On other hand, avatars that are grouped in a small region and close to the mass-center of a server will be assigned to that server by the density-based algorithm. However, since these avatars are located so closely, they will probably will share the same AOI. Therefore, the density-based algorithm also provides an initial partition with low inter-server communication requirements for those avatars.

However, the assignment of avatars equidistant or located far away from the mass-centers is critical for obtaining a partition with minimum inter-server communication requirements (minimum values of the quality function C_p), particularly for large virtual worlds with only a few servers. Density-based algorithm inherently provides good assignments for clustered avatars, but it does not properly focus on the assignment of these critical avatars. Each of the following heuristic methods should be used at this point to search a near optimal assignment that properly re-assigns these avatars.

Simulated Annealing (SA) This heuristic search method is based on a thermodynamic theory establishing that the minimum energy state in a system can be found if the temperature decreasing is performed slowly enough. Simulated Annealing (SA) is a heuristic search method that always accepts better solutions than the current solutions, and also accepts worse solutions according to a probability system based on the system temperature.

SA starts with a high system temperature (a high probability of accepting a worsening movement), and in each iteration system temperature is decreased. Thus, SA can leave local minima by accepting worsening movements at intermediate stages. The search method ends when system temperature is so low that worsening movements are practically impossible. Since the method cannot leave local minima, it can not find better solutions, neither (the algorithm ends when certain amount of iterations N are performed without finding better solutions).

Each iteration consists of randomly select two different critical avatars assigned to different servers. Then, the servers that two critical avatars are assigned to are exchanged. If the resulting value of the quality function C_p is higher than the previous

one plus a threshold T, that change is rejected. Otherwise, it is accepted (the search method must decrease the value of the quality function C_p associated with each assignment). The threshold T used in each iteration i of the search depends on the rate of temperature decreasing R, and it is defined as

$$T = R - \left(\frac{R \times i}{N} \right) \quad (2)$$

where N determines the finishing condition of the search. When N iterations are performed without decreasing the quality function C_p, then the search finishes.

As literature shows [10], the two key issues for properly tuning this heuristic search method are the number of iterations N and the temperature decreasing rate R. Although they are not shown here due to space limitations, we obtained the best results for SA method with N=3000 iterations. Performing more iterations increased the required execution times and it did not provide better values of C_p. Regarding to the rate of temperature decreasing, it did not have an effect on the required execution time of the algorithm. However, we obtained the best C_p values with R=1.25.

Random Search (RS) We have also implemented a simpler heuristic based on SA. We have denoted it as random search (RS), and it consists of the SA method when eliminating the system temperature. That is, in each iteration the threshold T is not considered, and if the resulting C_p is higher than the current minimum value, then that change is simply rejected. In this case, the only parameter to be tuned is the number of iterations N that determines the finishing condition. As in the SA case, the best results were obtained for N = 3000 iterations.

Ant Colony System (ACS) This heuristic search method is derived from the behavior of ant colonies when searching for food [4]. Each ant adds an hormone called *pheromone* to the path she follows, and the ants behind her will select their path depending on the pheromone each path contains (positive feedback). On other hand, pheromone evaporates at a given rate. Therefore, the associated pheromone for every path decreases if that path is not used during certain period of time (negative feedback). Evaporation rate determine the ability of the system for escaping from local minima.

The first step in the ACS method is to select the subset of border avatars from the set of all the avatars in the system. A given avatar is selected as a border avatar if it is assigned to a certain server S in the initial partition and any of the avatars in its AOI is assigned to a server different from S. For each of the border avatars, a list of candidate servers is constructed, and a certain level of pheromone is associated to all the elements in the list. This list contains all of the different servers that the avatars in the same AOI are assigned to (including the server that the avatar is currently assigned).

ACS method consists of a population of *ants*. Each ant consists of performing a search through the solutions space, providing a given assignment of the B border avatars to servers. The number of ants N is a key parameter of the ACS method that should be tuned for a good performance of the algorithm. Each iteration of the ACS method consists of computing N different ants (assignments of the B border avatars). When each ant is completed, if the resulting assignment of the B border avatars produces a

lower value of the quality function C_p, then this assignment is considered as a partial solution, and a certain amount of pheromone is added to the servers that the border avatars are assigned to in this assignment (each ant adds pheromone to the search path she follows). Otherwise, the ant (assignment) is discarded. When each iteration finishes (the N ants have been computed), the pheromone level is then equally decreased in all the candidate servers of all of the border avatars, according to the evaporation rate (the pheromone evaporates at a given rate). ACS method finishes when all the iterations have been performed.

In the process described above, each ant must assign each border avatar to one of the candidate servers for that avatar. Thus, a *selection value* is computed for each of the candidate servers. The selection value S_v is defined as

$$S_v = \alpha \times pheromone \ + \ \beta \times C_p \tag{3}$$

where $pheromone$ is the current pheromone level associated to that server, C_p is the resulting value of the quality function when the border avatar is assigned to that server instead of the current server, and α and β are weighting coefficients that must be also tuned. The server with the highest selection value will be chosen by that ant for that border avatar.

On other hand, when a partial solution is found then the pheromone level must be increased in those servers where the border avatars are assigned to in that solution. The pheromone level is increased using the following formula:

$$pheromone \ = \ pheromone \ + \ Q \times \frac{1}{C_p} \tag{4}$$

We have performed empirical studies in order to obtain the best values for α, β and Q coefficients. Although the results are not shown here due to space limitations, we have obtained the best behavior of the ACS method for $\alpha = 1.0$, $\beta = 7.0$ and $Q = 1000$. Additionally, we have tuned the values for the number of ants N, the pheromone evaporation rate and the number of iterations that ACS method must perform to obtain a near optimal partition. We obtained the best results for N=25 iterations, an evaporation rate of 1% and a population of 100 ants.

Genetic Algorithms (GA) This heuristic consists of a search method based on the concept of evolution by natural selection [8]. GA starts of an initial population (the initial partition) and then it evolves a certain number of *generations* (iterations), providing an *evolved population* (final solution).

The proposed implementation for solving the partitioning problem starts with a population composed of a set of elements called *genomes* or *chromosomes*. The number of chromosomes is the number of partial solutions that each iteration must provide. Each chromosome is defined by a descriptor vector containing a given assignment of avatars to servers.

Starting from the initial population, each generation (iteration) is found by exchanging some elements of the population, in such a way that in each of the N chromosomes two border avatars assigned to different servers are randomly chosen and exchanged. Thus, an iteration performed on a population of N chromosomes will produce a new

population of $2N$. From these $2N$ chromosomes, the N elements with the lower value of C_p will be chosen.

GA is also capable of escaping from local minima due to the *mutation* process. In each iteration, a mutation consists of randomly changing the server assigned to one of the elements (chromosomes) of the population.

The main parameters to be tuned in GA search method are the population size P, the number of iterations N and the mutation rate M. Although they are not shown here due to space limitations, we obtained the best results for P=15 individuals, N= 300 iterations and M= 1%.

Greedy Randomized Adaptive Search (GRASP) This search method is a constructive technique designed as a multi-start heuristic for combinatorial problems [6]. In this case, the initial partition does not provide any assignment for the border avatars, and the GRASP method is used to make these assignments.

Each iteration consists of two steps: construction and local search. The construction phase builds a feasible solution choosing one border avatar by iteration, and the local search also provides a server allocation for the AOI of that border avatar in the same iteration, following the next procedure: First, the resulting cost C_p of adding each non-assigned border avatar to the current (initial) partition is computed. Since each border avatar can be assigned to different servers, the cost for assigning each border avatar to each server is computed, forming the *list of candidates (LC)* (each element in this list has the form (*non-assigned border avatar, server, resulting cost*). This list is sorted (using Quick-sort algorithm) by the resulting cost C_p in descendent order, and then is reduced to its top quartile. One element of this reduced list of candidates (RLC) is then randomly chosen (construction phase). Next, an extensive search is performed in the AOI of that selected avatar. That is, all the possible assignments of the avatars in the AOI of the selected avatars are computed, and the assignment with the lowest C_p is kept.

The quality of the solution provided by GRASP search method depends on the quality of the elements in the RLC, and the range of solutions depends on the length of the RLC. Thus, the main parameter to be tuned in this case is the number of *non-assigned avatars* N that the initial partition must leave. Although they are not shown here due to space limitations, we obtained the best performance for N=5 in the case of small virtual worlds and N=20 in the case of large virtual worlds (a detailed description of both a small and also a large virtual world is shown in the next section).

4 Performance Evaluation

In this section, we present the performance evaluation of the heuristics described in the previous section when they are used for solving the partitioning problem in DVE systems. Following the evaluation methodology shown in [13], we have empirically tested these heuristics in two examples of a DVE system: a small world, composed by 13 avatars and 3 servers, and a large world, composed by 2500 avatars and 8 servers. We have considered two parameters: the value of the quality function C_p for the partition provided by the search method and also the computational cost, in terms of execution

time, required by the search method in order to provide that partition. For comparison purposes, we have also implemented the *linear optimization technique* (LOT) [13]. This method currently provides the best results for the partitioning problem in DVE systems. In the case of small worlds we have also performed an exhaustive search through the solution space, obtaining the best partition as possible. The hardware platform used for the evaluation has been a 1.7 GHz Pentium IV with 256 Mbytes of RAM.

Since the performance of the heuristic search methods may heavily depend on the location of the avatars, we have considered three different distributions of avatars: uniform, skewed, and clustered distribution. Figure 1 shows the 2D spatial location of avatars in each of these distributions.

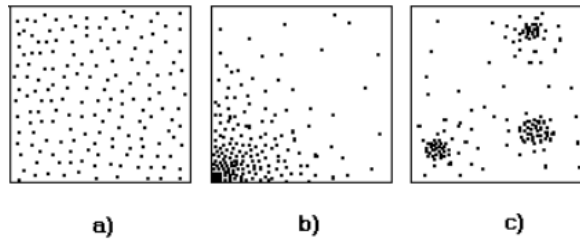


Fig. 1. Different avatar distributions in a DVE: (a) uniform, (b) skewed, and c) clustered

Table 1 shows the C_p values corresponding to the final partitions provided by each heuristic search method for a small virtual world, as well as the execution times required for each heuristic search method to obtain that final partition. It can be seen that all of the heuristics provide better (lower) C_p values than the LOT search method for a uniform distribution of avatars. For the skewed and clustered distributions, most of the heuristics also provides better C_p values than the LOT search method, and some of them (GA and SA methods) even provide the minimum value. However, the execution times required by most of the heuristics are longer than the ones required by the LOT method. Only GRASP method provides worse C_p values than the LOT method, but it requires much shorter execution times. Although these results does not clearly show which heuristic provides the best performance, they validate any of the proposed heuristics as an alternative to the LOT search method.

However, in order to design a scalable DVE system the partitioning method must provide good performance when the number of avatars in the system increases. That is, it must provide a good performance specially for large virtual worlds. Table 2 shows the required execution times and the C_p values obtained by each heuristic search for a large virtual world. In this case, all of the heuristics provides similar values of C_p than the LOT heuristic for the uniform distribution, while requiring much shorter execution times. When non-uniform distributions of avatars are considered, then all of the heuristics provide much better C_p values than the LOT method and they also require much shorter execution times than the LOT method. In particular, ACS method provides the best C_p values for the non-uniform distributions, requiring also the shortest execution time in the case of a clustered distribution.

	Uniform distrib.		Skewed distrib.		Clustered distrib.	
	Time (s.)	C_p	*Time (s.)*	C_p	*Time (s.)*	C_p
Exhaustive search	3.411	6.54	3.843	7.04	4.783	7.91
Linear opt. method	0.0009	6.56	0.001	8.41	0.0011	8.89
Simul. Annealing	0.004	6.82	0.005	7.46	0.005	7.91
Random Search	0.002	7.37	0.005	8.06	0.006	8.35
Ant Colony System	0.0007	6.59	0.003	7.61	0.0024	8.76
Genetic Algorithms	0.002	6.54	0.003	7.04	0.005	7.91
GRASP	0.0002	7.42	0.0002	8.63	0.0003	11.88

Table 1. Results for a small DVE system

	Uniform distrib.		Skewed distrib.		Clustered distrib.	
	Time (s.)	C_p	*Time (s.)*	C_p	*Time (s.)*	C_p
Linear opt. method	30.939	1637.04	32.176	3460.52	43.314	5903.80
Simul. Annealing	6.35	1707.62	13.789	2628.46	29.62	4697.61
Random Search	8.90	1687.55	13.826	2685.62	28.792	4676.22
Ant Colony System	5.484	1674.08	14.05	2286.16	23.213	3736.69
Genetic Algorithms	6.598	1832.21	14.593	2825.64	29.198	4905.93
GRASP	6.622	1879.76	13.535	2883.84	26.704	5306.24

Table 2. Results for a large DVE system

These results show that the performance of the partitioning algorithm can be significantly improved by simply using any of the proposed heuristics instead of the LOT method, thus increasing the scalability of DVE systems. In particular, ACS method provides the best performance as a partitioning algorithm.

5 Conclusions

In this paper, we have proposed a comparison study of modern heuristics for solving the partitioning problem in DVE systems. This problem is the key issue that allows to design scalable and efficient DVE systems. We have evaluated the implementation of different heuristics, ranging over most of the current taxonomy of modern heuristics. We have tested the proposed heuristics when applied for both small and large DVE systems, with different distributions of the existing avatars in the system. We have compared these results with the ones provided by the Linear Optimization Technique (LOT), the partitioning method that currently provides the best solutions for DVE systems.

For small virtual worlds, we can conclude that in general terms any of the implemented heuristics provides similar values of the quality function C_p, but the execution times required by the implemented heuristics are longer than the time required by the LOT search method. Although SA and GA methods provide the minimum value of the quality function, only GRASP method provides execution times shorter than the ones required by the LOT method for all the avatar distributions. These results validates any of the proposed heuristics as an alternative to the LOT search method when considering

small DVE systems. However, for large virtual worlds any of the proposed heuristics provides better C_p values and requires shorter execution times than the LOT method for non-uniform distributions of avatars. In particular, ACS method provides the best results. Since a scalable DVE system must be able to manage large amounts of avatars, we can conclude that these results validates ACS search method as the best heuristic method for solving the partitioning problem in DVE systems.

References

1. D.B.Anderson, J.W.Barrus, J.H.Howard, "Building multi-user interactive multimedia environments at MERL", in *IEEE Multimedia*, 2(4), pp.77-82, Winter 1995.
2. P. Barham, T.Paul, "Exploiting Reality with Multicast Groups", in *IEEE Computer Graphics & Applications*, pp.38-45, September 1995.
3. H. Delmaire, J.A. Daz, E. Fernndez, and M. Ortega, "Comparing new heuristics for the pure integer capacitated plant location problem", Technical Report DR97/10, Department of Statistics and Operations Research, Universitat Politecnica de Catalunya, Spain, 1997.
4. M. Dorigo and L. Gambardella, "Ant colony system: A Cooperative Learning Approach to the Traveling Salesman Problem", in *IEEE Trans. Evolut. Comput.* ,1997.
5. R.Duda, P.Hart, D.Stork, "Pattern Classification", *Ed.Wiley Intescience, 2000*, pp. 567-580.
6. Thomas A. Feo, Mauricio G.C, "Greedy Randomized Adaptive Search Procedures", *Resende Journal of Global Optimization*, 1995.
7. M. Fernndez, I. Coma, G. Martn and S. Bayarri, "An Architecture for Optimal Management of the Traffic Simulation Complexity in a Driving Simulator", *Lecture Notes in Control and Information Sciences*, Springer-Verlag, Vol. 243, 1999. ISBN 1-85233-123-2.
8. Randy L. Haupt, Sue Ellen Haupt, "Practical Genetic Algorithms", *Ed. Willey*, 1997.
9. J.C.Hu, I.Pyarali, D.C.Schmidt, "Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-Speed Networks", *Proc. of the 2nd. IEEE Global Internet Conference*, November.1997.
10. P.V. Laarhoven and E. Aarts, "Simulated annealing: Theory and applications", *Reidel Pub.*, Dordrecht, Holland, 1987.
11. Michael Lewis and Jeffrey Jacboson, "Game Engines in Scientific Research", in *Communications of the ACM*, Vol 45. No.1, January 2002.
12. John C.S. Lui, M.F.Chan, Oldfield K.Y, "Dynamic Partitioning for a Distributed Virtual Environment", *Department of Computer Science*, Chinese University of Hong Kong, 1998.
13. Jonh C.S. Lui, M.F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems", *IEEE Trans. Parallel and Distributed Systems*, Vol. 13, March 2002
14. D.C.Miller, J.A. Thorpe, "SIMNET: The advent of simulator networking", in *Proceedings of the IEEE*, 83(8), pp. 1114-1123. August, 1995.
15. Tohei Nitta, Kazuhiro Fujita, Sachio Cono, "An Application Of Distributed Virtual Environment To Foreign Language", in *IEEE Education Society*, October 2000.
16. J.M.Salles Dias, Ricardo Galli, A. C. Almeida et al. "mWorld: A Multiuser 3D Virtual Environment", in *IEEE Computer Graphics*, Vol. 17, No. 2, March-April 1997.
17. S.Singhal, and M.Zyda, "Networked Virtual Environments", *ACM Press, New York, 1999.*
18. P.T.Tam, "Communication Cost Optimization and Analysis in Distributed Virtual Environment", *M. Phil second term paper, Technical report RM1026-TR98-0412*. Department of Computer Science & Engineering.The Chinese University of Hong Kong. 1998.

Optimal Exploitation of Client Texture Hardware Capabilities on a Client-Server Remote Visualization Framework

Imma Boada[1] and Isabel Navazo[2]

[1] Institut Informàtica i Aplicacions, Universitat de Girona, Spain
imma@ima.udg.es
[2] Dep. LSI , Universitat Politècnica de Catalunya, Spain
isabel@lsi.upc.es

Abstract. Given a client-server communication network, with workstations equipped with 3D texture hardware, we propose a technique that guarantees the optimal use of the client texture hardware. We consider the best representation of a data model that has to be rendered on the client side as the one that requires the minimal texture space while preserving image quality. Taking into account this consideration the basis of the proposed technique is the selection of the best multiresolution representation from a volume hierarchy, maintained by the server. The key points of our proposal are: (i) the hierarchical data structure used, either by the server and the client, to maintain the data; (ii) the data management process applied by the server to satisfy client requirements.(iii) the possibility of the client to predict part of one transmission by analyzing the previous one. Such a capability allows the client to perform some computations in advance and, therefore, reduces frame rates.

keywords: Remote Visualization, multiresolution, 3D Textures.

1 Introduction

Remote visualization is an increasingly important area of research in scientific computing. It is defined as the visualization that utilizes data and computing resources that are physically distributed.

The majority of recent remote visualization techniques are based on a client-server paradigm. These methods can be broadly classified in four groups characterized by the client and the server functions. In the first group the server renders images and streams them to the client [5,6]. In the second group the server performs some rendering calculations and the client finishes the rendering locally. In the third group the server performs only the large computations and leaves the client to handle all the rendering computations [7]. Finally in the last group the server only provides raw data to the client and the client performs the visualization and all the scientific computation [10,12]. To achieve an efficient remote visualization we require networks with high bandwidth and low latency, efficient inter stage communication mechanism on the network and

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 468–477, 2003.
© Springer-Verlag Berlin Heidelberg 2003

proper adaptation and partitioning of the visualization computation [3]. Thus determine from the above mentioned methodologies which is the best is completely dependent on the environment where it has to be developed and applied. It is not possible to consider a general remote visualization framework nor define the optimal strategy.

Taken into account all these considerations and due to the popularity of texture hardware capabilities of actual graphics cards we have centered our interest on a client-server 3D texture-based remote visualization. We propose a framework that guarantees an optimal use of client hardware resources. The proposed method proceeds as follows. The server maintains the data in a hierarchical data structure. During a connection session between the server and a client, the server receives the client's machine parameters, then based on these parameters the server sends a list of nodes selected from the hierarchical data structure. The selected nodes guarantee the optimal use of the client hardware capabilities. On receiving these nodes the client sends them to its graphics card to perform the rendering. Once the first visualization has been obtained the user located on the client side selects a region of interest (ROI). Then the server transmits data of the ROI in an incremental fashion. The transmission is done in such a way that the client can predict part of the information and perform some computations in advance to reduce frame rates. All received data is stored by the client in a hierarchical data structure that allows the client to visualize information from any point of view.

Since the 3D texture-based volume rendering is one of the key points of the proposed approach we summarize in next section the main features of this technique.

2 3D Texture-Based Volume Visualization

Hardware 3D texture-based rendering was first mentioned by Akeley [1] in 1993, and then used and enhanced in other works [9,4,15]. Fundamentally, this technique renders a volume dataset by a back-to-front composition of a set of planes that slices and samples the volumetric dataset which is loaded into the texture memory of the graphics subsystem. The resulting set of planes is drawn as a set of textured polygons that are blended together to obtain the final image. Efficiency is gained by performing an automatic volume data sampling on the cutting planes via the trilinear interpolation capability supported by the graphics subsystem. The rendering time is negligible if compared to other software-based approaches as texture mapping and compositing operations are performed by hardware very quickly.

In general, methods based on 3D textures consider that the entire volume data resides in texture memory. In this case the technique exploits all its capabilities. However, when the volume dataset does not fit the available texture memory, the common bricking process applied to render such datasets (partition of the datasets in bricks, each one loaded and processed independently) reduces frame rate and makes rendering not sufficiently interactive. The multiple texture

memory loads and swaps required to render the complete volume are the main reason for the overhead. Several strategies have been proposed to carry out such a bricking process in an efficient manner (straightforward data partition [8], no empty voxel representation [13][14], multiresolution representation [11][2] ...). The technique we propose to efficiently manage the texture memory of the client side is based on a multiresolution strategy that simplifies the representation of homogeneous and/or no important regions of the volume [2]. The advantages of this strategy are presented in next section.

3 Proposed Approach

Our main objective has been to develop a framework able to connect workstations equipped with 3D graphics hardware with the guarantee that it is exploited in an optimal way. To present our proposal we are going to consider the server and the client sides independently.

3.1 SERVER

The most important functions of the server are the maintenance of original volume data and the satisfaction of the client demands in an optimal way. The two main entities the server uses to reach both objectives are: the Volume Octree and the Volume Octree Manager.

The Volume Octree The Volume Octree (VO) is the data structure used by the server to maintain the data. The VO is an octree-based codification used to code homogeneous regions of 3D regular sampled data in a compressed way. Each VO node maintains information of the maximum error which is introduced if each sample data inside its associated octant is approximated by the eight values represented on the corners of the octant. This error, denoted the *nodal error*(ε_0) is related to the degree of homogeneity in the area covered by the node. This parameter is used to obtain error driven multiresolution representations of the volume that reduce the space required for the representation of no importance and/or homogeneous regions [2].

The Volume Octree Manager To render the volume on the client side the server transmits, in an incremental fashion, a multiresolution representation of the volume and the information required to determine how texture memory has to be efficiently managed. The VO Manager is the process that handles and selects the set of nodes that define the multiresolution representation. This process distinguishes between the first client request and the next requests.

The first client request. We consider that when exploring a large data set rarely all the volume is of maximal interest. Therefore the first client visualization aims to have a general view of the model, which will allow the client to select the ROI for the next rendering. Based on this assumption the two main

features of the first visualization are: (1) all the volume has to be visualized (with more or less precision) and (2) the time required for the rendering has to be minimal. Driven by these constrains, when the server receives the first request for rendering, the VO manager evaluates network and client machine parameters (bandwidth, texture memory, the client allowed error, ...) and determines the first set of nodes, $S_0 = \{n_0, n_1, \ldots, n_n\}$, to be sent. S_0 has to satisfy that for each possible root-leaf path one and only on of its nodes is contained in it. S_0 defines the block-based decomposition of the dataset domain that has to be used to render the volume on the client side.

Each S_0 node has associated a set of samples that will be used on the client side to define a texture (see next section for details). To determine the selection of these samples the next considerations are taken into account. The voxels of the original dataset covered by a node can be represented at a different accuracy modifying the number of samples used to represent it (and assigning on the client side texture bricks of different resolution) (see figure 1). Given ε_c the client-defined accuracy threshold we define the optimal representation of the node as the smallest representation of the node which satisfy that the nodal error is lower than ε_c. For performance reasons to select S_0 it has to be taken into account that the number of nodes that compose it cannot be too large, and also the spatial extent in voxel space cannot be too small (otherwise the number of textured polygons will increase excessively and aliasing could become massive) [2]. Taking into account all these considerations to select S_0 the VO is recursively traversed. Starting at the root the server tests if its optimal representation satisfies the client texture memory restrictions. If it satisfies them the node is transmitted and the traversal ends. Otherwise the server recursively evaluates the descendants of the current node, with the constraint that the sum of the texture representations chosen for all the descendants should satisfy texture restrictions. The texture restrictions are set by the server considering client texture space and volume resolution.

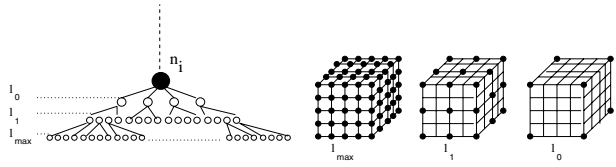


Fig. 1. The node n_i has three levels of descendants, therefore three different representations can be considered.

For each S_0 node the server transmits $\{pos_{VO}, l, l_{max}, < samples >\}$ where pos_{VO} identifies the position of the node in the VO. Note that pos_{VO} also identifies the origin, (x_0, y_0, z_0), and the end ,(x_e, y_e, z_e), of the volume area covered by the node. The parameter l is the VO level selected to represent the node and l_{max} is the level that gives the minimal error in the representation. The

last parameter $< samples >$ is the set of volume samples that will be used on the client side to define the texture. $< samples >$ is composed by the samples that define the lower resolution subvolume obtained when the original volume samples of the area covered by the node (i.e. from (x_0, y_0, z_0) to (x_e, y_e, z_e)) are taken every $2^{l_{max}-l} \times 2^{l_{max}-l} \times 2^{l_{max}-l}$ samples.

Non first request. Next transmissions correspond to refinements performed on S_0 once a ROI has been defined by the client. On receiving the ROI coordinates the server determines the node or nodes from the previous transmission that cover the ROI. For all the nodes contained in the ROI that are not represented with maximal precision (i.e. with $l \neq l_{max}$) the refinement process starts.

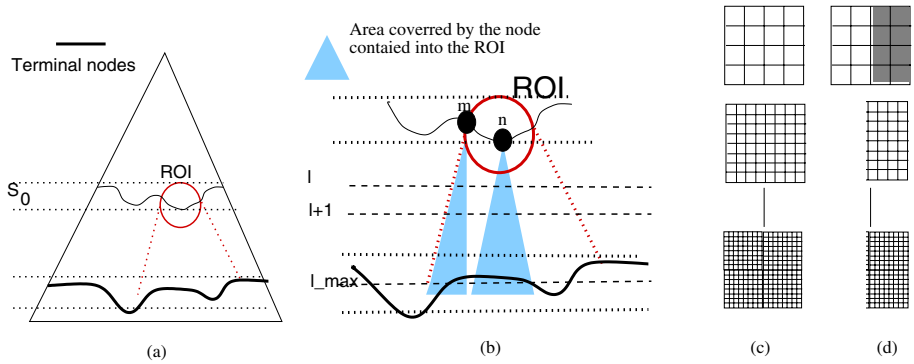


Fig. 2. (a) S_0, the distribution of terminal VO nodes and the ROI defined by the client. (b) Selection of a node n that have to be refined.(c)(d)(e) From less to more accurate subsampled representations of the volume covered by n.

Two possible situations are considered: (1) when the node is completely inside the ROI and (2) when the node intersects part of the ROI. The first situation corresponds to node n of Figure 2(b). In this case the texture has to be expanded until samples of l_{max} are represented. This expansion process requires $l_{max} - l$ transmissions (see Figure 2(c)). To avoid the duplication of information only the samples not sent in the previous transmission are considered. The second situation corresponds to node m of Figure 2(b). In this case only the part of the node contained into the ROI has to be refined. The descendants of the node are evaluated until they can be classified as completely (or almost completely) inside or outside the ROI. Nodes outside are represented with the resolution assigned to m in S_0 and the inside nodes are refined proceeding as in situation 1 (see Figure 2(d)).

The information that has to be transmitted in each case is represented in Table 1. When the node is in the ROI the transmitted parameters are pos_{OV} the position of the node, $flag_{ref}$ mark to indicate that an expansion process has to be applied and $< newsamples >$ the samples used for the expansion. When the node intersects the ROI the server sends: pos_{OV} the position of the node,

Node Completely inside ROI	Transmitted Information
Yes	$pos_{OV}, flag_{ref}, < newsamples >$
No	$pos_{OV}, l_{sub}, newpos_{OV}, flag_{noref} ...$ $newpos_{OV}, flag_{ref}, < newsamples >, ...$

Table 1. *Possible refinement situations and the information transmitted in each case.*

l_{sub} the level at which it has to be subdivided, $newpos_{OV}$ to identify the position of the node obtained in the subdivision, $flag_{noref}$ to indicate that no refinement has to be applied and $flag_{ref}$ to indicate that a refinement has to be applied. These two last parameters are associated to a node, and in the case of $flag_{ref}$ to a set of samples.

3.2 CLIENT

Taking advantage of its hardware capabilities the main functions of the client are the maintenance and the visualization of the data transmitted by the server.

Partial Volume Octree. The Partial Volume Octree (PVO) is an octree based codification that maintains the information transmitted by the server. Terminal PVO nodes are the nodes of S_0 (see Figure 3(a)). The PVO maintenance considers two situations. The first situation is related to PVO nodes classified as inside the ROI. The subsampled representations assigned to these nodes are expanded, thus new samples related to these nodes have to be properly stored. The second situation is related to nodes intersecting the ROI. These nodes have to be subdivided and their associated subsampled representations have to be updated (see Figure 3(b)). The information required to perform all these modifications is represented in Table 1.

Note that as the PVO maintains the original information the client is able to perform visualizations from any point of view just modifying the order of projection of the nodes.

Client Visualization. To visualize data the client applies the 3D texture-based volume visualization approach with the advantage that the main decisions related to the texture memory management have been taken by the server. The visualization process distinguish between a first and a refined visualization.

First Visualization. The server sends the parameters required for the octree initialization. Then S_0 is transmitted and stored in the PVO. For each received node the client defines one texture applying the transfer function to map volume values to $RGB\alpha$ values. The order of the transmitted nodes is back-to-front with respect the position determined by the client when starting the connection. Thus, as soon as a texture is constructed, it is loaded in texture space and mapped to a set of polygons (which is determined according the resolution of the node) Then it is projected on the screen. The position where it has to be projected is

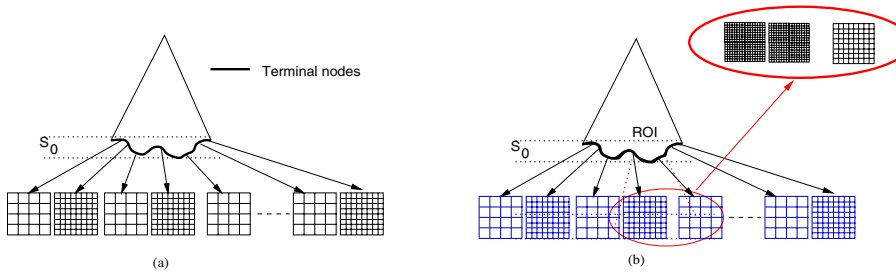


Fig. 3. (a) The Partial Volume Octree. Each terminal node maintains one pointer to one of the nodes of S_0. (b) When the refinement process starts the information related to nodes of the ROI is updated.

set according the resolution of the model and the resolution and origin of the node.

Refined Visualizations. The client updates the PVO and expands or generates the textures with the data stored in the terminal PVO nodes. These nodes are rendered in back to front order as soon as their corresponding textures are generated.

4 Client-Server Communication Messages

Before a connection session the server has performed an initialization process to generate the VO. Then it opens a public communication channel to listen for client connections. Once this preprocessing step has been carried out a connection session can be established. The main messages over a connection session are (see Figure 4): *Call for connection.* The connection is initiated by a client. The client sends its machine parameters: client name, memory and texture memory size, rendering capabilities and viewpoint position. *Approve connection.* The server evaluates client parameters and checks if it satisfies the minimum requirement to visualize the server's dataset. Then the server approves or refuses the connection. This message contains the information required to initialize internal lists and data structures. *Initialize rendering* The server selects and transmits the set of VO nodes that compose S_0. *ROI specification.* The client sends the coordinates of the ROI. *Refined rendering.* The server sends the information related to the refinement process. *Close Connection.* This message is send by the client to indicate the end of a connection session.

5 Evaluation

To evaluate the model we consider that the VO has been obtained in a preprocessing from a $2^X \times 2^Y \times 2^Z$ volume model. The depth of the VO is L, with $L = max\{X, Y, Z\}$. Note that L represents an upper bound of l_{max} value and is only reached in the worst situation when any kind of simplification can be

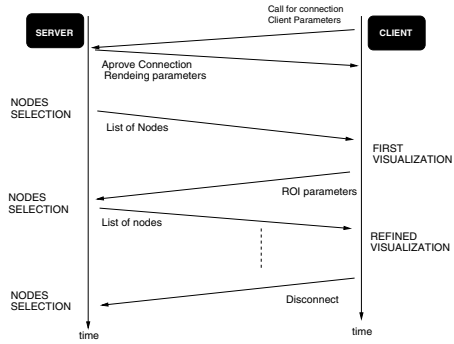


Fig. 4. Main messages over one connection session.

performed on the original model.The time to compute S_0 (t_{S_0}) includes the time to traverse the VO and detect the nodes that satisfies texture memory restrictions and error requirements. As the goal of the first visualization is to have a general view of the model, we have tested several models and it has been experimented that a model obtained from a $\frac{(2^X \times 2^Y \times 2^Z)}{8}$ subsampled representation gives enough information to the client (see first column of Figure 5). Thus the cost to obtain S_0 is the cost to represent $\frac{(2^X \times 2^Y \times 2^Z)}{8}$ samples (i.e. the nodes of level $L-1$). For performance reasons the number of S_0 nodes (i.e. the number of textures to be defined) cannot be too large, and also the spatial extent in voxel space cannot be too small. Considering 2^{3T} the maximal texture size that fits in the texture memory of the client, we have defined this value as the maximal texture size and 16^3 as the minimal [2]. Hence the number of nodes of S_0 (N_{S_0}) satisfies $8^{L-T-1} \leq N_{S_0} \leq 8^{L-5}$. Since the obtaining of S_0 requires traverse all the VO t_{S_0} is $\frac{8^{L-T}-1}{7} \leq t_{S_0} \leq \frac{8^{L-4}-1}{7}$.

The total cost of C_{S_0} transmission is

$$t_{S_0} + \frac{\sum_{i=0}^{N_{S_0}} (bpos_{OV} + bl + bl_{max} + resnode \times (b \times sample))}{10^6 B} + Lat$$

where the first term corresponds to the time required to obtain S_0, the second term corresponds to the transmission cost of S_0 nodes, being b the number of bits required for the representation of the associated value and B the network bandwidth (Mbs). The last term, Lat is the latency. To compute the total cost of the proposed model we have to take into account the refinement process cost C_{ref}. This cost depends on the number of nodes of S_0 contained in the ROI (N_{ROI}) and on the resolution assigned for the representation of these nodes on S_0. As S_0 maintains pointers to VO nodes the time of refinement of each N_{ROI} node is the time to traverse the octree from the level used to represent the node in question to its l_{max}. Taking into account all the terms the transmission cost, $C_{S_0} + C_{ref}$ is lower than the cost of transmitting all the model $2^X \times 2^Y \times 2^Z \times b$. Note that only the ROI has maximal accuracy thus we can ensure that S_0 samples + Samples from refinement are lower than $2^X \times 2^Y \times 2^Z$. Even in the

case that all the model is considered of maximal interest the proposed approach is of interest as the server determines the textures that have to be defined to exploit data homogeneity. A first version of the method has been implemented and tested. Two Pentium IV equipped with a NVidia Quadro2 graphics card have been connected via a 100MBit Ethernet network. The images of Figure 5 where obtained from a $512 \times 512 \times 81$ skull data model and from a $256 \times 256 \times 40$jaw data model. In the case of maintaining the viewpoint position almost interactive frame rates were achieved, when modifying the viewpoint position there was a delay of seconds. Texture memory space was artificially reduced to force texture bricking. New tests have to be done to analyze different compression techniques, the implementation has also to be speed optimized.

6 Conclusions and Future Work

We have presented a client-server communication framework designed to improve the use of client texture hardware capabilities. Driven by the client texture memory constrains, the server selects from a VO codification the multiresolution volume representation that better fits on the client side (giving a simplified representation to homogeneous and no importance regions). The most attractive feature of the method is that all texture memory management decisions are taken by the server, the client only has to generate and render the textures. Future work is addressed to the problem of combining surface and volume data on the transmissions.

Acknowledgements. This work has been partially supported by the project TIC2001-2226 of the Spanish Ministry of Science and Technology. The authors thank Dani Marin for his help on the software programming.

References

1. K. Akeley, Reality Engine Graphics. Computer Graphics (ACM Siggraph Proceedings), 27:109-116,1993.
2. I.Boada, I.Navazo and R.Scopigno., Multiresolution Volume Visualization with a Texture-based Octree. The Visual Computer, Springer International, 17 (3), pp. 185-197, 2001.
3. J. Bowie, H.W. Braun, R. Guerin, G.M. Perulkar and D. Stevenson, Remote Visualization: Challenges and Opportunities, Visualization 1991, Nielson and Rosemblum editors, IEEE Computer Society Press, pp.340-344.
4. B.Cabral, N.Cam and J.Foran, Accelerated Volume Rendering and Tomographic Reconstruction using Texture Mapping Hardware. In ACM Symposium on Volume Visualization,pp. 91- 98 Washington, D.C. October 1994.
5. K.Engel, O.Sommer, C.Ernst and T.Ertl, remote 3D visualization using image streaming techniques. In Advances in Intelligent Computing and multimedia Systems (ISIMADE'99) pp. 91-96, 1999.
6. K.Engel, O.Sommer and T.Ertl, Accelerated Remote 3D visualization Framework. In Proceedings of EG/IEEE TCGV Symposium on Visualization Vissym'00, May 2000.

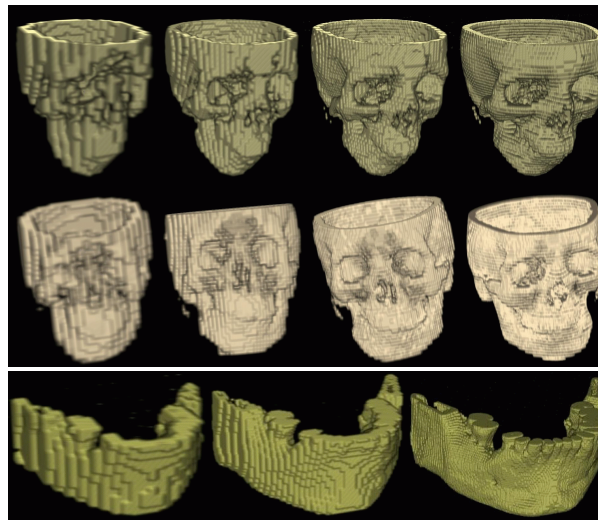


Fig. 5. Visualizations performed on the client side when all the volume has been considered of maximal interest. The first raw corresponds to a first transmission of the skull data set visualized from the same point of view. The second raw corresponds to some of the images obtained on a second transmission when the client modifies the viewpoint position. The last raw corresponds to the jaw data model.

7. K.Engel and T.Erlt, Texture-based Volume Visualization for Multiple Users on the Wordl Wide Web, In Gervautz and hilbebrand and Schmalstieg editors. Virtual Environments'99, pp. 115-124. Eurographics Springer 1999.
8. R.Grzeszczuk, C.Henn, and R.Yagel. Advanced Geometric Techniques for Ray Casting Volumes. Course Notes. SIGGRAPH '98. ACM July 1998.
9. S.Guan and R. Lipes. Innovative Volume rendering using 3D Texture Mapping. In Image Capture, Formatting and Display.SPIE 2164, 1994.
10. W. Hibbard, J.anderson and B.Paul A Java World Wide Web Implementation of VisAD In Interactive Information and Processing Systems for Meteorology, Oceonography and Hydrology, pp. 174-177, 1997
11. E.LaMar, B.Hamman and K.Joy, Multiresolution Techniques for interactive texture-based volume visualization. In IEEE Visualization 99. IEEE CS Press, October 1999.
12. C. Michaels and M. Bailey Viswiz: a Java Applet for Interactive 3D Scientifci Visualization on the Web. In Proceedings Visualization'97, pp. 261-267, IEEE Computer Society Press, 1997
13. R.Srinivasan, S.Fang, S.Huang, Volume Rendering by Template-Based Octree Projection. Workshop Eurographics 97. Visualization in Scientific Computing.
14. X.Tong, W.Wang, W.Tsang, Z.Tang. Efficiently Rendering Large Volume Data Using Texture mapping Hardware. Data Visualization '99.
15. O.Wilson, A.Van Gelder, J.Wilhems, Direct Volume rendering via 3D textures. Technical Report UCSC-CRL-94-19, University of California, Santa Cruz, June 1994.

AIF - A Data Structure for Polygonal Meshes

Frutuoso G.M. Silva and Abel J.P. Gomes

IT - Networks and Multimedia Group
Department of Informatics, UBI, Portugal
{fsilva,agomes}@di.ubi.pt

Abstract. Meshing is an important topic in geometric modelling and computer graphics. This paper introduces a concise and fast data structure, called AIF (Adjacency and Incidence Framework). Its conciseness results from the fact that it is an orientable, but not an oriented, data structure, i.e. an orientation can be topologically induced as necessary in many applications. It is an optimal C_4^9 data structure for polygonal meshes, manifold and non-manifold, which means that a minimal number of direct and indirect accesses are required to retrieve adjacency and incidence information from it. In fact, it operates close to real-time even for huge meshes, what becomes it appropriate for real-time applications (e.g. multiresolution meshing refinement and simplification operations).

1 Introduction

Polygonal meshes are used in geometric systems and applications such as, for example, multiresolution meshing, virtual reality, rendering of solids and surfaces, and so on. In fact, much research on polygonal meshes has been done in the last few years, particularly in multiresolution analysis [5],[6], mesh morphing [10], interactive mesh edition [20], and even in geometric compression and transmission over the web [16].

Different applications have specific needs. Some applications demand more storage space than speed, some demand fast retrieval of adjacency and incidence information. Unfortunately, these retrieval operations are not as fast and efficient as necessary for current applications and geometric data structures, in particular for meshes with a huge amount of cells (say, vertices, edges, and faces). This is due to the design of the data structures somehow. For example, finding a cell in the cell-tuple data structure [2] implies processing all cell-tuples, a time-consuming operation in particular for large meshes.

Some data structures represent a triangular mesh by a set of faces, each face composed by a tuple of vertices [8]. This is the case of VRML [4] (Virtual Reality Modelling Language) file-format. This makes it difficult to design and implement fast algorithms to retrieve adjacency and incidence data. For example, finding the set of faces incident at a given vertex implies traversing all the faces in the data structure. But, this search algorithm is time-consuming, in particular for large meshes.

A way to speed up search algorithms is to use oriented boundary representation data structures (Winged-Edge [1], Half-Edge [12] or Radial Edge [18]), but

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 478–487, 2003.
© Springer-Verlag Berlin Heidelberg 2003

they need additional storage space for oriented cells. In a way, these oriented cells are redundant. For example, in the Radial Edge data structure, each face has two associated loops, one for each face side; hence, three faces incident along an edge require six oriented edges.

Kallmann and Thalmann [9] proposed a new data structure, called Star-Vertex, to represent polygonal meshes. This data structure is based on the incidence information around a vertex. It is concise, but the retrieval of adjacency and incidence information is slow because it does not keep any information about edges and faces explicitly.

Other data structures were designed only for triangular meshes as, for example, Directed Edges [3], Tri-Edges [11], and, more generally, PSC (Progressive Simplicial Complexes) [15] data structures. The first two are oriented, while the PSC data structure is non-oriented. This means that the PSC data structure is stripped off any oriented simplexes or cells. Consequently, PSC can be used to represent both orientable and non-orientable objects. Thus, unlike oriented b-rep data structures, there is no simplex redundancy in PSC data structure. Besides, PSC are able to represent n-dimensional simplicial complexes. However, the lack of explicit oriented simplexes in PSC or a geometric orientation mechanism for simplexes poses some difficulties in rendering. In fact, unlike progressive meshes proposed by Hoppe [7], the PSC data structure avoids explicitly storing surface normals at vertices. Instead, it makes usage of smoothing group fields for different materials as used by Wavefront Technologies [17].

This paper introduces a new data structure, called AIF (Adjacency and Incidence Framework), for generic meshes. These meshes need not be manifold or triangular. Thus, it may accommodate simplicial complexes and, more generally, cell complexes with or without dangling cells. Topologically speaking, it is an orientable, but not oriented, data structure, i.e. it does not possess oriented cells. The result is a more concise and flexible data structure, without losing control on the local adjacency and incidence information.

This paper is organized as follows. Section 2 describes the AIF representation. Section 3 presents its companion operator, called mask operator, for fast retrieval of adjacency and incidence information. Section 4 presents the AIF data structure. A comparison of several data structures, including the AIF data structure, appear in Section 5. At last, Section 6 draws some conclusions and future work.

2 Adjacency and Incidence Framework

It is said that a cell x is adjacent to a cell y (symbolically, $x \prec y$) if x is contained in the frontier of y (or, equivalently, $fr(y) \cap x \neq \emptyset$) and the dimension of x is lesser than the dimension of y; equivalently, one says that y is incident on x (symbolically, $y \succ x$). For example, a vertex bounding an edge is said to be adjacent to it, but there may be many edges incident at the same vertex. The adjacency (incidence) relation is transitive, i.e. if $v \prec e$ and $e \prec f$ then $v \prec f$.

Besides, there are only two basic adjacency relations for 2-dimensional cell complexes, namely $V \prec E$ and $E \prec F$; $E \succ V$ and $F \succ E$ are their inverse relations or incidence relations. These four basic relations can be compounded to form nine adjacency relations introduced by Weiler [19]. For example, the Weiler $V \rightarrow F$ and $E \rightarrow E$ relations can be obtained as follows:

$$V \rightarrow F = (V \rightarrow E) \circ (E \rightarrow F) = (E \succ V) \circ (F \succ E)$$
$$E \rightarrow E = (E \rightarrow V) \circ (V \rightarrow E) = (V \prec E) \circ (E \succ V)$$

Thus, the four basic relations above constitute a representation in the class C_4^9 according to Ni and Bloor [13]. They form the best representation in the class C_4^9 in terms of information retrieval performance, i.e. it requires a minimal number of direct and indirect accesses to the data structure to retrieve those four explicitly-represented adjacency relations and the remaining five implicitly-represented adjacency relations, respectively. A direct access query involves a single call to the mask operator, while an indirect access requires two or more calls to the mask operator, i.e. a composed query. (The mask operator is described in Section 3.)

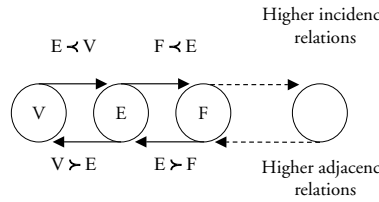


Fig. 1. AIF diagram.

The AIF (Adjacency and Incidence Framework) data structure accommodates the optimal C_4^9 representation for 2-dimensional meshes, but it can be easily generalized to $C_{2n}^{(n+1)^2}$ for n-dimensional objects (Fig. 1). It keeps the essential adjacency and incidence data that allows us to derive supplementary data to traverse any mesh.

A 2-dimensional mesh is defined by means of a triple $M = \{V, E, F\}$, where V is a finite set of vertices, E is a finite set of edges, and F is a finite set of simply connected faces.

A vertex $v \in V$ is defined by a non-ordered k-tuple of edges incident at v, i.e. $v = \{e_1, e_2, e_3, .., e_k\}$, where e_i $(i = 1..k)$ is an incident edge at v. This is the way as the $E \succ V$ relation is embedded in the AIF.

An edge $e \in E$ is a pair of tuples, being the first for vertices bounding it, and the second for faces incident on it, i.e. $e = \{\{v_1, v_2\}, \{f_1, f_2, .., f_k\}\}$, where v_1, v_2 are its bounding (or adjacent) vertices, and $f_1, f_2, .., f_k$ are its incident faces. So, we have $V \prec E$ and $F \succ E$ relations.

A face $f \in F$ is defined in terms of its k-tuple of bounding (adjacent) edges as follows $f = \{e_1, e_2, e_3, .., e_k\}$, where each e_i $(i = 1..k)$ is an edge bounding f. This definition includes the $E \prec F$ relation.

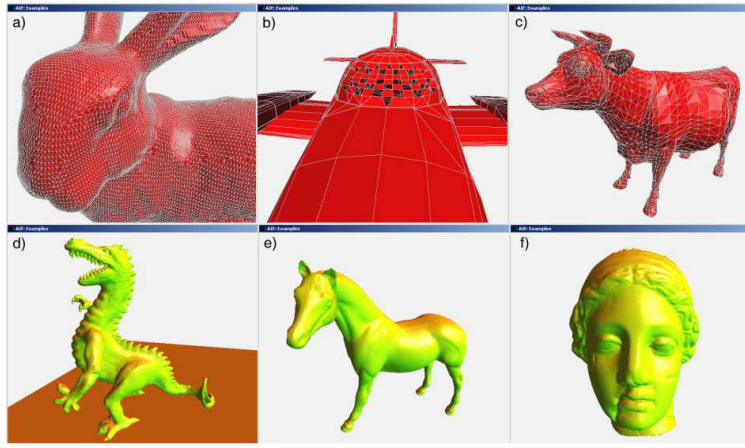


Fig. 2. AIF polygonal meshes.

With these four basic adjacency and incidence relations embedded in the AIF we are able to represent manifold (Fig. 2(a)(c)(d)(e)(f)) and non-manifold (Fig. 2(b)) polygonal meshes. In Fig. 2(b), the plane mesh is non-manifold because some cockpit faces (say, dark regions) are missing. Note that an edge may have any number of incident faces. An edge e without incident faces is represented by $e = \{\{v_1, v_2\}, \{\}\}$, i.e. a dangling edge. An edge with more than two incident faces (Fig. 1(b)) is represented through its bounding vertices and incident faces as follows $e = \{\{v_1, v_2\}, \{f_1, f_2, f_3\}\}$. Thus, the AIF data structure allows us to distinguishing between manifold and non-manifold objects.

Besides, the AIF may accommodate general polygonal meshes, regardless of whether they are triangular (Fig. 2(a)(c)(d)(e)(f)) or not (Fig. 2(b)).

3 Adjacency and Incidence Operator

Time is a critical factor in real-time meshing applications. We need fast query and retrieval algorithms to locally identify cells adjacent to or incident at/on other cells in order to quickly refine and simplify meshes.

Before proceeding, let us pay attention to the AIF incidence scheme. The incidence scheme of a mesh can be described in terms of a set of cell-tuples T=$\{(v_i, e_j, f_k)\}$, where f_k is a face incident on an edge e_j (symbolically, $f_k \succ e_j$) and e_j is incident at a vertex v_i ($e_j \succ v_i$); alternatively, we say that v_i is adjacent to e_j ($v_i \prec e_j$), and e_j is adjacent to f_k ($e_j \prec f_k$). For example, the incidence scheme for a tetrahedron (Fig. 3(a)) is as follows:

$$
\begin{array}{cccc}
(v_1, e_1, f_2) & (v_2, e_1, f_2) & (v_3, e_2, f_1) & (v_4, e_3, f_1) \\
(v_1, e_1, f_3) & (v_2, e_1, f_3) & (v_3, e_2, f_3) & (v_4, e_3, f_2) \\
(v_1, e_2, f_3) & (v_2, e_6, f_2) & (v_3, e_4, f_3) & (v_4, e_5, f_1) \\
(v_1, e_2, f_1) & (v_2, e_6, f_4) & (v_3, e_4, f_4) & (v_4, e_5, f_4)
\end{array}
$$

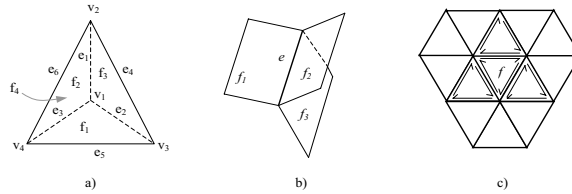


Fig. 3. (a) A manifold mesh; (b) a non-manifold object; (c) the orientation mechanism.

$$\begin{array}{cccc} (v_1, e_3, f_2) & (v_2, e_4, f_3) & (v_3, e_5, f_1) & (v_4, e_6, f_2) \\ (v_1, e_3, f_1) & (v_2, e_4, f_4) & (v_3, e_5, f_4) & (v_4, e_6, f_4) \end{array}$$

This incidence scheme can be considered as a data structure on its own. It is known as the cell-tuple data structure, and is due to Brisson [2]. The AIF data structure has the same adjacency and incidence descriptive power as the cell-tuple data structure, but it is more concise and less time-consuming. In fact, the AIF (Fig. 1) consists of a set of cells (not a set of cell-tuples), with each cell defined as follows: a vertex is defined in terms of its incident edges; an edge is defined in terms its bounding vertices and incident faces; a face is defined in terms of its frontier edges.

The AIF data structure uses a single adjacency and incidence operator, called mask operator. The mask operator is defined by $\bowtie\!\triangleleft_d : V \times E \times F \rightarrow C$, with $C = V \cup E \cup F$ being the union of the set V of vertices, the set E of edges, and the set F of faces, such that $\bowtie\!\triangleleft_d(v_i, e_j, f_k) = \{c_l^d\}$, i.e. a set of d-dimensional cells. The arguments of $\bowtie\!\triangleleft_d$ are cells in the set $V \times E \times F$. A NULL cell as argument of dimension $n = d$ means that all the n-cells satisfying the adjacency/incidence condition expressed by the cell arguments are to be returned; otherwise, if $n \neq d$ and the n-cell argument is also NULL, no n-cell imposes any adjacency/incidence restriction on the d-cells to be returned. In case $n = d$ and the n-cell argument is not NULL, the operator $\bowtie\!\triangleleft_d$ returns all the n-cells as before, except the n-cell argument; if $n \neq d$ and the n-cell is not NULL, the n-cell imposes an additional adjacency/incidence restriction on the retrieved d-cells.

Let us consider again the mesh in Fig. 3(a) to illustrate how the mask operator works in conjunction with the AIF data structure:

1. $\bowtie\!\triangleleft_1(v_1,\text{NULL},\text{NULL}) = \{e_1, e_2, e_3\}$ directly returns all edges incident at v_1.
2. $\bowtie\!\triangleleft_2(v_1,\text{NULL},\text{NULL}) = \{f_1, f_2, f_3\}$ indirectly returns all faces incident at v_1. This requires an intermediate call to $\bowtie\!\triangleleft_1(v_1,\text{NULL},\text{NULL})$ to return all edges e_1, e_2, e_3 incident at v_1. Then, the operator $\bowtie\!\triangleleft_2(\text{NULL}, e_i,\text{NULL})$ is called for each edge e_i in order to compute faces incident on e_i and v_1.
3. $\bowtie\!\triangleleft_0(\text{NULL}, e_1,\text{NULL}) = \{v_1, v_2\}$ directly returns bounding vertices of e_1.
4. $\bowtie\!\triangleleft_2(\text{NULL}, e_1,\text{NULL}) = \{f_2, f_3\}$ directly returns faces incident on e_1.
5. $\bowtie\!\triangleleft_0(\text{NULL},\text{NULL}, f_1) = \{v_1, v_3, v_4\}$ indirectly returns all vertices bounding f_1. This requires an intermediate call to $\bowtie\!\triangleleft_1(\text{NULL},\text{NULL}, f_1)$ to first de-

termine all edges bounding f_1. Then, the operator $\blacktriangleright\!\triangleleft_0(\text{NULL}, e_i, \text{NULL})$ is called for each edge e_i in order to determine vertices bounding e_i and f_1.

6. $\blacktriangleright\!\triangleleft_1(\text{NULL}, \text{NULL}, f_1) = \{e_2, e_3, e_5\}$ directly returns all edges bounding f_1.

With the mask operator, there is no need to handle all the data structure constituents. It allows us to manipulate a mesh locally. Thus, its time performance holds independently of the mesh size. This is very important for handling large meshes, in particular in real-time mesh operations.

4 Data Structure Implementation

The AIF data structure represents a mesh (or cell complex) consisting of a set of cells (vertices, edges and faces). A C++ class codes each cell type, but all cell types could be implemented as a single C++ class as needed for n-dimensional meshes. This is possible because every n-dimensional cell is seen as a set of incident $(n+1)$-dimensional sets and a set of adjacent $(n-1)$-dimensional cells, i.e. the incidence and adjacency relations themselves. The AIF data structure is then as follows:

```
class Vertex {
   evector li;//incident edges
   Point *pt; //geometry
   Point *nv; //normal
}
 class Edge {
   Vertex *v1,*v2; //vertices
   fvector li;//incident faces
}
```

```
class Face {
  evector la; //adjacent edges
}
class Mesh {
  int id; //mesh id class
  vvector vv; //vertices
  evector ev; //edges
  fvector fv; //faces
}
```

Note that AIF data structure is not topologically-oriented provided that it does not include any oriented cells. It is geometrically-oriented by the vertex normal `nv` in the class Vertex. This normal vector is the arithmetic average of the normals to faces incident on such a vertex. Face normals are determined by topologically inducing the same or consistent orientation on them (e.g. clockwise orientation as shown in Fig. 3(c)). This requires traversing the frontier of any face consistently by alternately applying the operators $\blacktriangleright\!\triangleleft_0$ and $\blacktriangleright\!\triangleleft_1$ to the cells (edges and vertices) bounding such a face. The operator $\blacktriangleright\!\triangleleft_0$ gives us the next vertex in the frontier of a given face, and the operator $\blacktriangleright\!\triangleleft_1$ provides the next edge bounding the reference face. That is, by traversing a mesh with the help of the mask operator, it is possible to induce an orientation on a mesh.

5 Storage and Accessing Efficiency Comparisons

5.1 Storage Cost

Table 1 characterizes several data structures in terms of storage cost and accessing efficiency for a triangular mesh with n vertices. The "Mesh type" column

refers to the type of mesh supported, either triangular (△) or polygonal (any), while the "Non-manifold" column indicates whether a data structure supports non-manifold meshes or not. The "Bytes/△" column gives us the number of bytes per triangle. The variable k stands for the vertex degree, i.e. the number of edges incident at a given vertex.

Table 1. Data structures storage and accessing comparisons

Data structures	Mesh type	Non-manifold	Bytes/△	Not defined	Class
Triangle list	△	yes	18	6	C_2^9
Star-vertex	any	yes	10+4k	6	C_2^9
Progressive meshes	△	no	33	6	C_2^9
Tri-edge	△	no	35	6	C_2^9
AIF	any	yes	29+2k	0	C_4^9
PSC	△	yes	37+2k	0	C_4^9
Directed-edges	△	yes	44	3	C_2^9
Half-edge	any	no	46	0	C_2^9
FastMesh	△	no	53	3	C_2^9
Radial Edge	any	yes	56	0	C_2^9
Winged-edge	any	no	60	0	C_2^9

The Triangle List data structure is the most concise data structure because it only stores vertices and faces. Normally, it is used only for visualization purposes provided that graphical hardware is optimised for triangles.

The Star-Vertex data structure only stores vertices and their neighbouring vertices. Every neighbouring vertex comes with an index for navigation purposes (e.g. for traversing the vertices bounding a face counterclockwise). It is concise because it only needs 10+4k bytes per triangle. It also supports general meshes, but the adjacency information retrieval involves time-consuming operations because it does not include edges and faces explicitly. The absence of edges and faces implies that six adjacency/incidence relations are missing as illustrated by the "Not defined" column in Table 1.

Progressive Meshes [8] and Tri-edge [11] data structures are also more concise than the AIF data structure viewing that they spend 33 and 35 bytes per triangle, respectively. But, unlike the AIF data structure, they were designed only for triangular, manifold meshes.

On the other hand, an AIF mesh approximately requires 35, 37, 39, 41 bytes per triangle in main memory for k=3,4,5 and 6, respectively. In fact, according to the Euler formula for 2-dimensional triangular meshes, a mesh with n vertices has about m faces ($m = 2n$) and e edges ($2m = 3e$). So, assuming that floats and pointers are 4 bytes each, the corresponding runtime space cost is the following: $(34 + 4k)n = (12 + 4k)n = (6 + 2k)m$ bytes for vertex coordinates, $(24 + 24)e = 16e \approx 11m$ bytes for edges (references to vertices and faces), and $(34)m = 12m$ bytes for faces (references to edges).

Note that the AIF data structure does not include oriented cells (e.g. half-edges or directed edges). Therefore, it is more concise than traditional b-rep data structures (e.g. Winged-Edge [1]) and their oriented variants (e.g. Half-Edge [12], Directed-Edges [3], FastMesh [14], and Radial Edge [18]). At last, the PSC [15] data structure was designed only for triangular meshes and requires more memory space than the AIF data structure.

5.2 Retrieval of Adjacency and Incidence Information

The last two columns in Table 1 shows how many accesses are necessary to retrieve adjacency and incidence information from several data structures. Since we have three topological entities (vertex, edge and face), there are at most nine adjacency and incidence relations [19].

In Table 1, the "Class" column gives us the number of adjacency and incidence relationships represented in data structure explicitly. For example, a data structure C_4^9 has four out of nine adjacency and incidence relationships explicitly represented. In other words, we have direct access to them without additional processing overheads. The remaining five adjacency and incidence relations are represented implicitly viewing that all these data structures are complete, i.e. they represent meshes unambiguously. This means that we need inference mechanisms to retrieve such adjacency and incidence information, the so-called indirect accesses. It happens that these indirect accesses can be rather difficult because some topological cells are not in the structure. For example, the Star-Vertex has no edges and faces represented explicitly. Consequently, six adjacency and incidence relationships are missing (Table 1).

Looking again at Table 1, we see that the AIF and PSC [15] are the best data structures in terms of time efficiency. Of course, the time efficiency depends on the number of adjacency and incidence relationships explicitly stored in the data structure, and thus more space memory is needed. Both AIF and PSC data structures belong to the class C_4^9. According to Ni and Bloor [13], we can say that both data structures are optimal for meshes in the sense that a minimal number of direct and indirect accesses are required to retrieve adjacency and incidence information from them. However, the PSC [15] data structure only supports triangular meshes.

The other data structures in Table 1 belong to the class C_2^9 (two direct accesses and seven indirect accesses to retrieve adjacency and incidence information). For example, the FastMesh data structure has no vertex entities in the data structure. Thus, the three adjacency and incidence relationships involving vertices are missing, but they can be inferred with substantial computational cost anyway. The remaining six adjacency and incidence relationships involving edges and faces can be retrieved more easily.

5.3 Time Performance

The runtime performance tests were performed on a PC equipped with 1.6GHz Intel Pentium 4, 256MB of memory, an NVidia Riva TNT2 graphics card, and

running Windows 2000 OS. Table 2 shows some results for different AIF models (Fig. 2). The models are courtesy of Cyberware, Stanford University, Viewpoint Digital Inc., Avalon archive and 3D Cafe.

Loading an AIF model from a text file includes the mechanism to induce an orientation on the corresponding mesh. Rendering an AIF model involves two steps: (i) create its GL (Graphics Library) primitive; (ii) display such a GL primitive.

Table 2. Initialization and rendering times in seconds

Models	#v	#f	Loading disk	Orient	GL list	Display
Cow	2904	5804	0.180	0.060	0.050	0.020
CessnaNM	3745	3946	0.180	0.050	0.040	0.010
CessnaT	6795	13546	0.491	0.120	0.110	0.031
Dragon	25418	50761	1.622	0.701	0.431	0.220
Bunny	34835	69473	2.674	1.001	0.641	0.300
Horse	48485	96966	3.374	1.503	0.891	0.430
Venus	50002	100000	3.445	1.643	0.961	0.440

Looking again at Table 2, we see that rendering large manifold meshes (e.g. the venus, the horse, and the bunny depicted in Fig. 2) or non-triangular, non-manifold meshes (e.g. the Cessna pictured in Fig. 2(b)) is fast, even considering that GL rendering is done by software instead of hardware.

6 Conclusions and Future Work

This paper has introduced a new data structure, called AIF, together with a query operator in order to satisfy three major requirements, namely:

- *Generality*. AIF does not only support triangular meshes. It may accommodate manifold and non-manifold polygonal meshes, even in higher dimensions.
- *Conciseness*. Unlike other b-rep data structures, AIF is not oriented, i.e. it does not contain oriented cells. Thus, it is more concise than the traditional b-reps.
- *Responsiveness*. AIF was designed to ensure fast queries through a single indexed mask operator. It basically returns adjacency and incidence data.

In practice, the AIF data structure and its companion mask operator proved to be powerful in retrieving incidence and adjacency information. No oriented cells are needed as in conventional b-reps; hence, it is more concise than oriented data structures. In terms of information retrieval efficiency, its query complexity presents the best performance in the class C_4^9. In the near future, we hope that AIF data structure can address real-time application requirements as required, for example, in multiresolution and animation systems.

References

1. Baumgart, B. G.: Winged-edge polyhedron representation. Technical report, STAN-CS-320, Stanford University, 1972.
2. Brisson,E.: Representing geometric structures in d dimension: topology and order. Descrete & Computational Geometry, Vol.9, no.4, 1993, pp.387- 426.
3. Campagna, S., Kobblet, L., and Seidel, Hans-Peter: Directed Edges - A Scalable Representation for Triangular Meshes. JGT, Vol. 3. no.4, 1998, pp.1-12.
4. Carey, R., Bell, G., and Marrin, C.: The Virtual Reality Modeling Language ISO/IEC 14772-1, 1997. http://www.vrml.org/technicalinfo/specifications/
5. Garland, M.: Multiresolution modeling: Survey & future opportunities. In Eurographics - State of the Art Reports, 1999, pp.111-131.
6. Heckbert, P. S., and Garland, M.: Survey of Polygonal Surface Simplification Algorithms. Siggraph Course Notes 25, 1997.
7. Hoppe, H.: Progressive Meshes. Proceedings of Siggraph, 1996, pp.99-108.
8. Hoppe, H.: Efficient Implementation of Progressive Meshes. Computer & Graphics, Vol.22, no.1, 1998, pp.27-36.
9. Kallmann, M. and Thalmann, D.: Star-Vertices: A Compact Representation for Planar Meshes with Adjacency Information. JGT, Vol.6, no.1, 2001.
10. Lee, A., Dobkin, D., Sweldens, W. and Schrder, P.: Multiresolution Mesh Morphing. Proceedings of Siggraph, 1999, pp. 343-350.
11. Loop, C.: Managing Adjacency in Triangular Meshes. Tech. Report No. MSR-TR-2000-24, Microsoft Research, 2000.
12. Mäntylä, M.: An Introduction to Solid Modeling, Computer Science Press, 1988.
13. Ni, X., and Bloor, M. S.: Performance Evaluation of Boundary Data Structures. IEEE CG&A, Vol.14, no.6, 1994, pp.66-77.
14. Pajarola, R.: FastMesh: Efficient View-dependent Meshing. Proceedings of Pacific Graphics, 2001, pp.22-30.
15. Popovic, J., Hoppe, H.: Progressive Simplicial Complexes. Computer Graphics, Vol.31, 1997, pp.217-224.
16. Taubin, G., and Rossignac, J.: Geometric compression through topological surgery. ACM Transactions on Graphics, Vol.17, no.2, 1998, pp.84- 115.
17. Wavefront Technologies, Inc. Wavefront File Formats, Version 4.0 RG-10-004, first ed. Santa Barbara. CA. 1993.
18. Weiler, K.: The radial edge structure: a topological representation for non-manifold geometric boundary modelling. In Geometric modelling for CAD applications. Elsevier Science Publish. 1998.
19. Weiler, K.: Edge-Based data structure for solid geometric in Curved-Surface Environments. IEEE CG&A, Vol.5, No.1, 1985, pp.21-40.
20. Zorin, D., Schrder, P., and Sweldens, W.: Interactive Multiresolution Mesh Editing. Proceedings of Siggraph, 1997, pp.259-268.

High Dynamic Range Image Texture Mapping Based on VRML

Sung-Ye Kim and Byoung-Tae Choi

3D Graphics Research Team, Virtual Reality Research and Development Department,
Computer Software Research Laboratory,
Electronics and Telecommunications Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
{inside, btchoi}@etri.re.kr
http://insideblue.hihome.com, http://www.etri.re.kr

Abstract. In this paper, we describe a technique for representing and displaying high dynamic range image (HDRI) as a texture map in VRML structure. To do this, we designed and implemented extended texture nodes for supporting HDRI in VRML and developed HDRI-based mapping tool that can be used for authoring objects with HDR image as a texture data. Even though the authoring result of our tool is not identical with standard VRML format exactly, it is good enough to be used in virtual world. To verify this, we developed an extended VRML browser that supports a parsing and viewing of our new texture nodes related to HDRI and allows the exposure to be adjusted continuously and arbitrarily at the time of navigation.

1 Introduction

Recently, since a number of image-based techniques have begun to make appearance, it is possible to render realistic images with pretty low costs through the combination of computer vision and computer graphics technologies in modeling and rendering fields. However, 2D images as input data for an image-based technique intrinsically have limitations of representing all information being in the real world. Since the late of 1990's, information of real or natural lights has been used as source data for image-based techniques. Especially, such fields are called by an image-based lighting, and many researches[7][8] have been accomplished until now. High Dynamic Range Image is the image data type can represent real light in the world. The dynamic range of a scene is the ratio between its the brightest and darkest parts, and a HDRI is an image that has a greater dynamic range than LDR (Low Dynamic Range) that can be shown on a standard display device or be captured with a standard camera with a single exposure. Properties of a HDRI are as follows. The first, the pixel values of HDRI are directly proportional to the amount of light in the world corresponding to that pixel. Therefore it is possible to be represented the ratio of a bright and dark region precisely in HDRI. The second, various realistic optical effects such as a HDR fresnel, auto exposure, glow and vignette can be expressed. Because of these characteristics,

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 488–497, 2003.
© Springer-Verlag Berlin Heidelberg 2003

there have been a lot of researches related to application using HDRI. At present, various researches related to HDRI show the future course of realistic rendering in computer graphics. Tools[1] and plug-ins[3][4][5] for supporting HDRI have been produced by some researchers and developers. HDRShop[1] made by Paul Debevec is one of the most famous tools. This provides various HDR image processes and manipulations of HDRI such as loading, saving, editing, panoramic transformation, camera curve calibration, diffuse/specular convolution and assembling of HDRI using a series of images taken under different exposure. The LightGen[6] is an HDRShop compatible executable by Jonathan Cohen for generating a set of directional sources to approximate a light probe image. HDRI Bitmap I/O plug-in for 3D Studio Max and its tutorial are also available on website[3]. However, these only provide I/O operations for supporting HDRI in the existing rendering and modeling tools.

2 Texture Image Nodes Using HDRI

In VRML, Appearance node is used to describe characteristic of an object, and each node of VRML consists of some fields. One of fields of Appearance node is a texture field that specifies the type of image for texture mapping. In this section, we'll explain the design and implementation of our new texture image nodes for supporting HDRI as a texture image in VRML-like environment.

2.1 Representation of New Texture Image Nodes

Table 1 shows the types of texture that can be used as the texture field of Appearance node. VRML defines three texture types like the ImageTexture, the MovieTexture and the PixelTexture. To expand an image type of texture field, we added new texture nodes by the name of the *HDRImageTexture* and the *HDRICubeMapTexture* nodes. The *HDRImageTexture* node was added as a new node with the purpose to use HDRI as a texture image for texturing of objects in VRML scene. By using HDRI as a texture image, we can get some advantages of HDRI in itself. For example, we can adjust the exposure of an environment in virtual world freely. Table 2 shows the specification of the *HDRImageTexture* node. It seems like an ImageTexture node except an exposure field of SFFloat type. We can also create the *HDRICubeMapTexture* node based on above node of Table 2. Table 3 shows the specification of this node. Six HDRI maps are needed to do cube mapping.

2.2 Implementation of Our Texture Image Nodes

When we implement the *HDRImageTexture* node for our application, it has actually such the structure as Table 4. For the structure of a floating-point image data for a single *HDRImageTexture*, there is a pointer variable for image of uncompressed 32-bits floating-point value type, like floatRGBImage* floatImage. The structure of class of *HDRICubeMapTexture* node is similar to that of above

Table 1. Specification of Appearance node in VRML and image types of texture field in it. We added new texture nodes by the name of the *HDRImageTexture* and the *HDRICubeMapTexture* nodes.

Appearance {		
ExposedField SFNode	material	NULL
ExposedField SFNode	texture	NULL
ExposedField SFNode	textureTransform	NULL
}		
texture field:		
- ImageTexture		
- MovieTexture		
- PixelTexture		
- *HDRImageTexture*	(new)	
- *HDRICubeMapTexture*	(new)	

Table 2. *HDRImageTexture* node. This seems like an ImageTexture node except an exposure field. The difference between two nodes in url field is the fact that url field of our node will have HDRI file.

HDRImageTexture {			
ExposedField	MFString	url	[]
field	SFFloat	exposure	1.0
field	SFBool	repeatS	TRUE
field	SFBool	repeatT	TRUE
}			

Table 3. *HDRICubeMapTexture* node. This node have six url fields to represent a cube map and these urls are all HDRIs.

HDRICubeMapTexture {			
ExposedField	MFString	urlRight	[]
ExposedField	MFString	urlLeft	[]
ExposedField	MFString	urlTop	[]
ExposedField	MFString	urlBottom	[]
ExposedField	MFString	urlFront	[]
ExposedField	MFString	urlBack	[]
field	SFFloat	exposure	1.0
}			

node. Because *HDRICubeMapTexture* node should have six maps, a vector for tvFloatRGBImage, 'std::<vector tvFloatRGBImage*> floatImageVector', will be needed to store them. Fig. 1 shows the execution result of a scene using two texture nodes.

Table 4. Class structure for *HDRImageTexture* node. tvTexture is the name of a texture node that is offered in TOV library 1.0. We used TOV library to implement common VRML browser.

```
class HDRImageTexture : public tvTexture {
  // for VRML 2.0 fields
  private:
    tvStringVector* url;
    tvFloat exposure;
  // member data
  private:
    tvFloatRGBImage* floatImage;
    tvBool imageChanged;
}
```

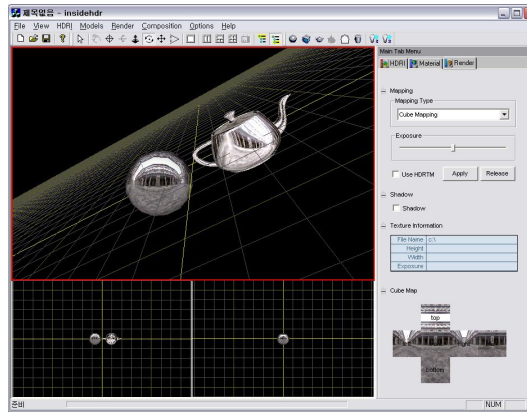


Fig. 1. Screen shot of a mapping using *HDRImageTexture* and *HDRICubeMapTexture* nodes. A sphere was mapped by cube map using *HDRICubeMapTexture* and a teapot was mapped by sphere map using *HDRImageTexture* node.

Table 5 shows the source of the scene using the *HDRImageTexture* and *HDRICubeMapTexture* nodes and the source file can be exported by our application. We also added texGen node for generating texture coordinate automatically. In Table 5, SPHERE_MAP of texGen node specifies the mode for a

Table 5. A scene source using *HDRImageTexture* and *HDRICubeMapTexture* nodes. Fig. 1 shows the execution result of this source. In second line, TOV toolkit V1.0 is the name of a rendering library for VRML scene and that was used in our application.

```
#VRML V2.0 utf8
#TOV Toolkit V1.0
Transform { translation -3 0 0
  children [
    Shape {
      appearance Appearance {
        texture HDRICubeMapTexture {
          urlRight "cmRight.tif"
          urlLeft "cmLeft.tif"
          urlTop "cmTop.tif"
          urlBottom "cmBottom.tif"
          urlFront "cmFront.tif"
          urlBack "cmBack.tif"
          exposure 1.0
        }
        texGen TexGen { mode CUBE_MAP }
      }
      geometry Sphere {}
    }
  ]
}
Transform { translation 3 0 0
  children [
    Shape {
      appearance Appearance {
        texture HDRImageTexture {
          url "insideHDRI0.tif"
          exposure 1.5
        }
        texGen TexGen { mode SPHERE_MAP }
      }
      geometry Teapot {}
    }
  ]
}
```

spherical texture mapping and CUBE_MAP specifies that for a cube mapping. As well known, cube mapping is view-independent. We also added the Utah Teapot node as geometric primitive.

3 HDRI-Based Application for Supporting HDRI Texture Mapping

We implemented a HDRI-based application for supporting the texture mapping using HDRI. This section is composed of two parts. Those are the authoring tool part for HDRI texture mapping with *HDRImageTexture* or *HDRICubeMapTexture* nodes and the extended VRML browser for supporting our new texture nodes.

3.1 insideHDR: HDRI-Based Authoring Tool

Fig. 2 shows the entire picture of our tool, insideHDR. The main processing of it is composed of three parts. See operations from the PROCESSING (1) to (3).

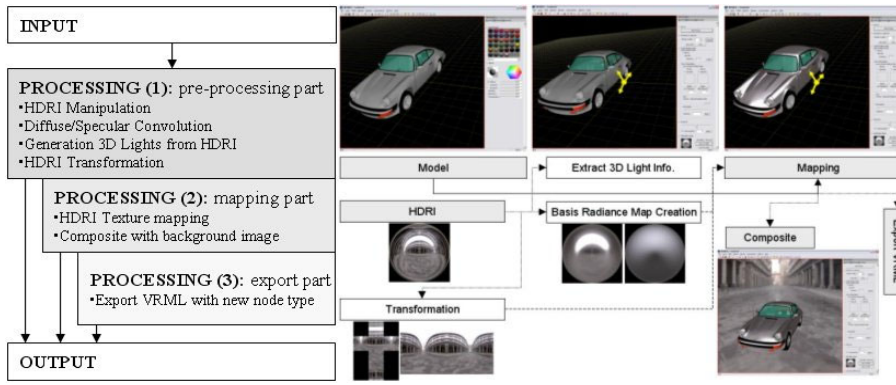


Fig. 2. Left picture shows the flow of the application from input to output. The processing part is composed of three parts. Right picture shows the snapshot at each step and the flow of operation.

The essential function of insideHDR is a texture mapping using HDRI and also we proposed a cube mapping using HDRI except basic mapping types were specified in VRML. Even though these aren't new research entirely, we applied these functions to VRML environment by creating new texture nodes. In relation to HDRI texture mapping, there was a research[8] by J.Cohen. His paper described the technique for representing and displaying high dynamic range texture maps (HDRTMs) using current graphics hardware by decomposing into sets of 8-bit textures HDRI. However, we don't use a decomposing method. We just create VRML-like nodes by applying tone mapping to HDRI and they are composed a scene graph. The insideHDR also provides some methods to manipulate HDRI such as loading, saving, diffuse/specular convolution, panoramic transformation, extract of light information and interpolation between specular radiance maps. These are same operations with those of HDRShop[1] by P.

Debevec and that of LightGen[6] by J. Cohen. In our tool, we represented the result of extracting a set of directional lights with user-friendly user interface. See Fig. 3.

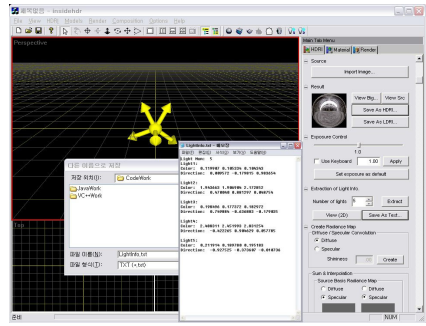


Fig. 3. Generation a set of 3D lights from HDRI. We can display the direction of lights with 3D arrows and 2D points and save the directions and colors of lights in text file. Above picture shows 3D representation of 5 lights extracted from HDRI on right side.

Below code is the function for applying new texture nodes to an object. We also provide common operations using ImageTexture and PixelTexture nodes of VRML. But such functions were omitted in below code.

```
void applyTexture()
  if (no texture data)
    if (CUBE_MAP)
      // 1.Create HDRI cube map node, HDRICubeMapTexture
      // 2.Define the name of the node to save as a file
      // 3.Save the HDRI data to HDRICubeMapTexture
    else
      // 1.Create HDRI texture map node, HDRImageTexture
      // 2.Define the name of the node to save as a file
      // 3.Save the HDRI data to HDRImageTexture
  else       // already have texture data
    if (CUBE_MAP)
      if (no change of mapping data type as HDRI->HDRI)
      else   // LDRI->HDRI
    else
      if (no change of mapping data type as HDRI->HDRI)
      else   // LDRI->HDRI
```

Fig. 4 shows the authoring of a car located in virtual showroom and the appearance of our tool. In the Fig. 4, the car model was mapped with *HDRIm-*

ageTexture of virtual show room. HDRI of the room was made with MAX5.0 manually.

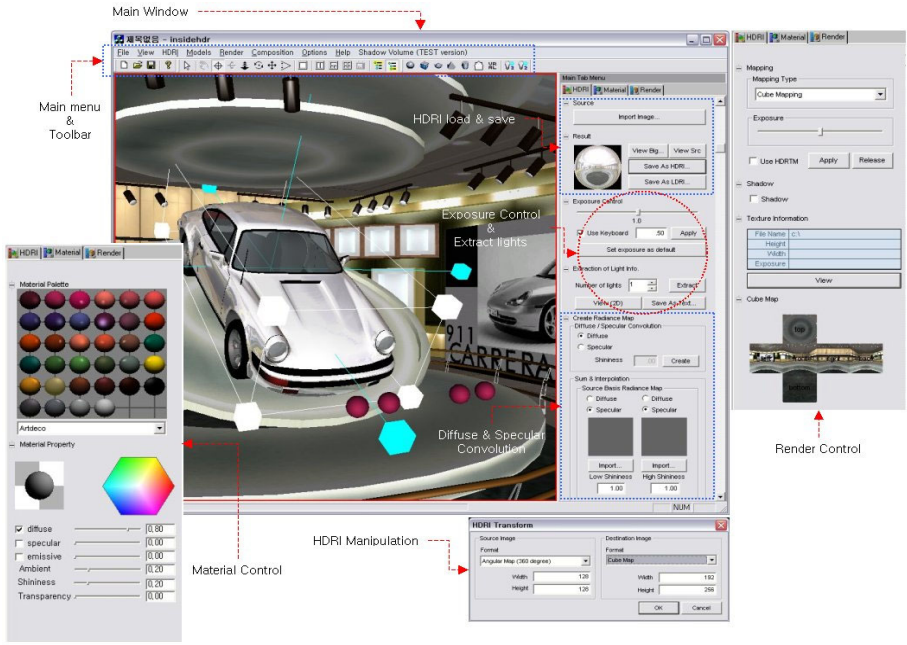


Fig. 4. This figure shows our HDRI-based authoring tool, insideHDR. This tool has various operations for manipulation of HDRI, changing of material of an object, and rendering using HDR texture image. This presents the authoring process of a car in virtual environment.

3.2 Extension of VRML Browser for Supporting New Texture Nodes

To use HDRI as a texture image, common VRML browser should have the routines for processing like parsing and displaying of the nodes related to HDRI. That is to say, some routines for parsing of our new nodes and displaying of floating-point typed image are needed in browser. Generally operation for display of floating-point typed image data is called as a tone mapping. By operating the routine, it is possible to use HDRI as texture image in our extended VRML browser. It may be very simple. Implementation of the extended browser is based on TOV library 1.0. TOV offers functions for parsing nodes in VRML2.0 specification and displays them. So, we can create new nodes by inheritance of tvTexture node that is a texture node in TOV 1.0 library.

4 Results and Future Works

We have demonstrated that high dynamic range images can be used as texture data in an extended VRML structure based on the Internet. The original goal of this paper is to develop the technique making possible use HDRI as a texture map in virtual space. So we designed new texture nodes related to HDRI in texture field of Appearance node in VRML and then developed HDRI-based mapping tool, insideHDR. By using the tool, we could create the object mapped with HDRI and save its authoring result as VRML-like formatted file. That is not identical with standard VRML but extended format. The viewer of our extended VRML browser based on OpenGL is used as a main window in insideHDR tool. Because it supports most operations as a web browser, as you can see in Fig. 6, we can also add JavaScript operations such as buttons for adjusting exposure and rotation of a model in our exported result. The insideHDR provides three main functions. The first, we can manipulate HDRI. For example, we can get different formatted HDRI, 3 dimensional light information and diffuse or specular radiance map from angular map Light Probe image and so on. The second, we can create VRML scene supporting HDRI as an image type of texture node. The final exported VRML file can be loaded and navigated by our extended browser. And then it is possible to adjust an exposure of that virtual environment in real time. In near future, because most of common browsers don't support HDRI type, we will also consider embedding of our work in the common VRML browser.

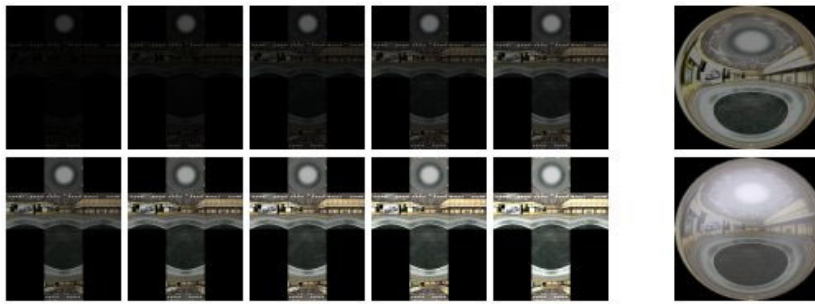


Fig. 5. (Left) A series of LDR cube maps images made under different exposure. We make HDRI by assembling them. (Right) Angular maps formatted HDRI created with left cube map HDRIs. (See Fig. 6 for the application of the usage of these HDRIs)

References

1. http://www.debevec.org/hdrshop/main-pages/tutorials.html
2. http://www.debevec.org/Probes

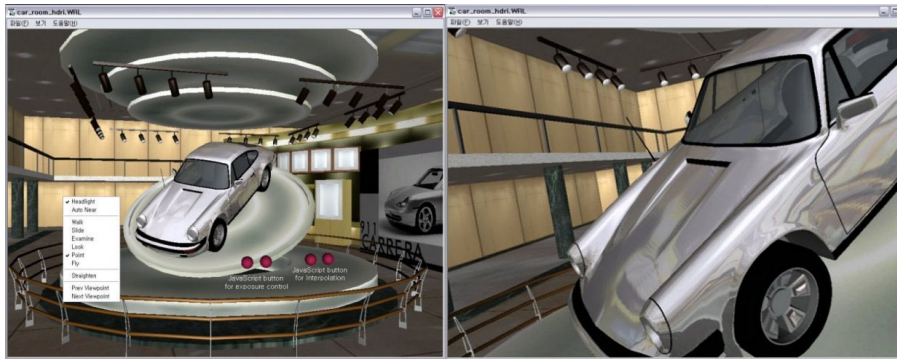


Fig. 6. Virtual showroom. In this room, a car was mapped with HDRI texture map of Fig. 5.

3. http://www.splutterfish.com/gabry_hdri_tut_tmp
4. http://www.lightwave3d.com/tutorials/fullprecision/index.html
5. http://www.ocf.berkeley.edu/ eliza/cs184
6. http://www.ict.usc.edu/ jcohen/lightgen/lightgen.html
7. Paul Debevec: Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. In Proceeding of SIGGRAPH'98, (1998) 189–198
8. Jonathan Cohen, Chris Tchou, Tim Hawkins, and Paul Debevec: Real-time High Dynamic Range Texture Mapping. In Eurographics Rendering Workshop, London, (2001)
9. Greg J. Ward: Real Pixels. Graphics Gems II, Edited by James Arvo, San Diego: Academic Press (1991) 80–83

CW Complexes: Topological Mainframe for Numerical Representations of Objects

Pascal Desbarats and Stefka Gueorguieva

LaBRI, Université de Bordeaux I
33405 Talence, France
desbarat@labri.fr, stefka@labri.fr

Abstract. Dimensionally non-homogeneous pointsets with internal structure are the focus in a consequent number of studies in both computational geometry and discrete (or digital) geometry, and lead to various practical applications in geometric modeling and computer imagery. Our motivation is to revisit the well known notion of algebraic topology, the cell CW complex, and to use it as an abstract framework for numerical representation of inhomogeneous objects. Two representational issues, respectively in non-manifold solid modeling and in discrete object boundary reconstruction, are discussed in illustration of this general setting.

1 Introduction

Mathematical fundamentals allow a rigorous characterization of the representational schemes and modeling operations of the continuously growing domain of objects in interest. With respect to geometric modeling and computational geometry for example, starting with the so called "manifold solids" and the corresponding topological data structures, manipulated with the set of the Euler Operators [45], more general classes of solids, presenting degeneracies, came up to view: "pseudo-manifolds" [24], "quasi-manifolds" [34, 10], and in general "non-manifolds" [60, 42, 20, 6, 19, 61, 38, 39, 5, 55]. An emerging fundamental concept of combinatorial topology, the cellular complex, becomes undeniabley a universal tool for construction and manipulation of dimensionally non-homogeneous objects with internal structures and produces the convergence of methods in generic domains such as the geometric modeling and computational geometry, and the computer imagery and discrete geometry [27, 22, 28, 15, 16, 23].

The aim of the present study is to revisit the "classic" cellular structure of the CW complexes [47, 44, 37] as a unified mainframe for solid and discrete object representation. The content of this work is the following. First, some basic notions are overviewed. Next, the mathematical framework utilization for two computer graphics and image processing applications is discussed: the topology-based geometric modeling and the boundary reconstruction of discrete objects.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 498–507, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 CW Complexes

The concept of *CW complex* is due to J.H.C Whitehead, 1953.
A Hausdorff space X is a *CW complex* with respect to a family of cells S providing that each cell $\sigma \in S$ has a continuous characteristic function, X has the weak topology with respect to S, and the cell complex (X, S) is closure finite. A CW complex X is *regular* if each closed cell is homeomorphic to a closed Euclidean n-cell. A CW complex is *normal* if each cell is a subcomplex.
Our interest is directed towards a certain type of CW complexes associated with the *semisimplicial complexes*.
A *semisimplicial complex* X consists of a sequence $\{X_n | n = 0, 1, \ldots\}$ of disjoint sets together with a collection of maps in each dimension n:
(1) $d_i : X_{n+1} \to X_n, i = 0, 1 \ldots n+1$, the ith $face$ operator ;
(2) $s_j : X_n \to X_{n+1}, j = 0, 1 \ldots n$, the jth *degeneracy* operator ;
such that the following *semisimplicial identities* are satisfied:
$d_i d_j = d_{j-1} d_i$, $i < j$; $\quad d_i s_j = s_{j-1} d_i$, $i < j$;
$d_i s_j = 1$, $i = j, j+1$; $\quad d_i s_j = s_j d_{i-1}$, $i > j+1$;
$s_i s_j = s_{j+1} s_i$, $i \leq j$.

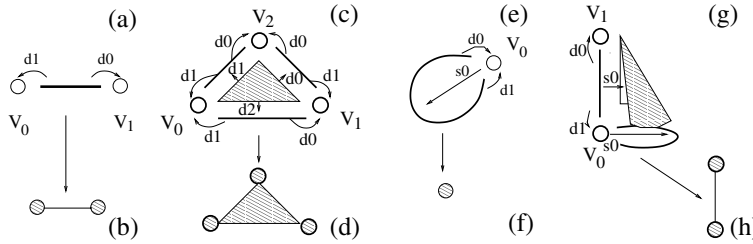


Fig. 1. Examples: (a) Nondegenerate $1D$ simplex and (b) its realization. (c) Nondegenerate $2D$ simplex and (d) its realization. (e) Degenerate $1D$ simplex and (f) its realization. (g) Degenerate $2D$ simplex and (h) its realization.

The elements of X_n are called the $n - simplexes$ of X. A simplex of form $s_i x$ is *degenerate*; one not of this form is *nondegenerate*.
The *geometric realization* $|X|$ of a semisimplicial complex X is a normal CW complex with one n-cell $|x_n|$ corresponding to each nondegenerate n-simplex x_n of X, and the faces of $|x_n|$ corresponding to the nondegenerate faces of x_n. See the illustrations on Fig.1. The *geometric realization* $|X|$, associated with each semisimplicial complex X could be considered [47] as a factorization of the semisimplicial complex. Unlike the case of a simplicial complex, some cells of $|X|$ need not to be homeomorphic to E^n. This property along with the semisimplicial construction *product*, *quotient* and *adjunction* will be used in the following to construct and to encode objects with degeneracies. Summary and extended discussion on the theoretical essential of CW complexes could be found in [8].

3 CW Complexes and Object Representations in Solid Modeling

3.1 Related Work

Combinatorial topology and cellular complexes emerged as a powerful background for numerous solid representations in a wide range of applications. See the examples in 2D and 3D mesh generation and refinement [11, 5] , boolean operations [48, 56, 57, 43, 58, 46], feature based modeling [54, 40, 51, 21, 3], multiresolution and object modeling [13, 29, 36, 41]

Pioneer researches in application of CW complexes in geometric modeling of area guide map are due to [59, 30]. The area guide map is considered as assembly of surface patches, each one being represented in terms of its singular points (peaks, passes and pits). Using Morse function and Reeb graph, each patch is represented as a CW complex. Further, surface patches are blended along common boundaries using homotopy between corresponding cellular characteristic maps. The final composite surface is restricted to differentiable manifold.

A significant contribution to cellular complex applications is made by [32, 33, 34, 35]. A wide set of combinatorial cellular structures has been proposed [35] to handle "irregular assemblies of regular cells" (semisimplicial sets [31], simploidal sets [17, 18]) and "irregular assemblies of irregular cells" (chains [10], generalized maps [34]). In the majority of works, a purely combinatorial approach is experimented and the reported [35] performance evaluation is analytic i.e. independent of actual implementation.

Simplicial decompositions are studied by [49, 50]. Modeling operations such as the boolean operations [48], the boundary evaluation and the extrusion [12] are elaborated. It is acknowledged however that the critical point of the proposed approach is the high representation fragmentation. Moreover, the underlying Winged Representation is restricted to a subclass of CW complexes, the regular CW complexes.

A great number of researches focalizes on geometric complexes, "Selective Geometric Complexes" [56], "Complete Geometric Complexes" [5], "Djinn Complexes" [46] and "Multi-Complexes" [13]. In spite of the further formalism definition, the descriptive domain remains restricted to solid representation and geometric modeling applications. There is a lack in representation unification for solids and discrete objects. In most cases, when an invalid geometric embedment or computation provokes degeneracies as for example the appearance of iD-cell, $i \geq 1$, embedded into a point, additional topological interpretation is required. Finally, no performance evaluation of the cellular structure is known even for common modeling operations such as the extrusion.

Our approach, in opposition with the above cited works, is in some way minimalist: to revisit the CW complexes as defined by [37] and to investigate to which extent this "minimalist" set of representation and manipulation tools contributes to the solutions of the above cited problems: to elaborate a unified representational frame, to encode object degeneracies with no complementary costs, and to implement a software kernel in order to evaluate cell structure performances.

3.2 *CW* Complexes and Solid Representation

Solid model in terms of CW complexes is a cellular structure of iD-cells, $0 \leq i \leq 3$. Model construction is a two step procedure. First, a list of principal cells belonging to the object is built, each iD principal cell being constructed in a recursive way in increasing order of dimension of the proper faces. The geometric embedment is defined for each proper face, and face and degeneracy operators are initialized while semisimplicial identities are ensured. Second, a merging operation is fulfilled in order to regroup cells with identical geometric embedment through adjunction and quotient semisimplicial constructions. In fact, the first phase corresponds to a semisimplicial complex construction and the second one, to its geometric realization. It should be emphasized that a iD degenerate cell is realized to a lower dimensional nondegenerate cell but owed to the corresponding sequence of degeneracy operators, the initial cell structure is maintained. For example, the geometric realization on Fig. 1(b) according to degeneracy operators could represent cell on either Fig. 1(a) or Fig. 1(g).
Some examples are given on Fig.2 and Fig.3. Next we will detail the model construction for the cube on Fig.2(a) and its evaluation is given in the first line on Table 1. The cube is constructed as a $2D$ quasi-manifold regular complex using the extrusion operation. Initially, the list of principal cells includes `12` $2D$-cells (each square is represented as 2 triangles). Each principal $2D$-cell has `6` proper faces, `3` $0D$-cells and `3` $1D$-cells. Working out the total for the whole complex, we have `36` $0D$-cells and `36` $1D$-cells. Further, the geometric realization is performed and the number of cells is reduced to `8` $0D$-cells and `12` $1D$-cells. The number of $2D$-cells remains unchanged as long as the final result is a $2D$ complex. The cost of the construction is given in number of geometric tests (point/point, point/line, etc.) with no accelerating technics. The CPU time for this construction for a Pentium III, 866MHz, RAM 256 MB, is quasi-instantaneous. The elaborated

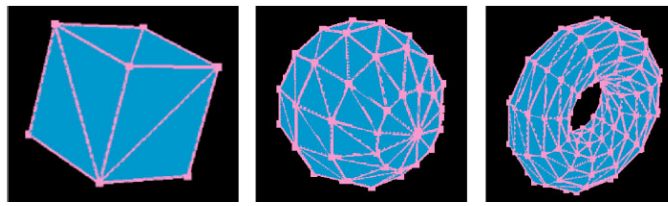


Fig. 2. Extrusion solid primitives: (a) Cube. (b) Sphere. (c) Torus.

evaluation on Table 1 accentuates the model fragmentation reduction in the special case of extrusion operation for CW complexes supporting solids on Fig.2 and Fig.3 and constructed as $2D$ quasi-manifold regular complexes.

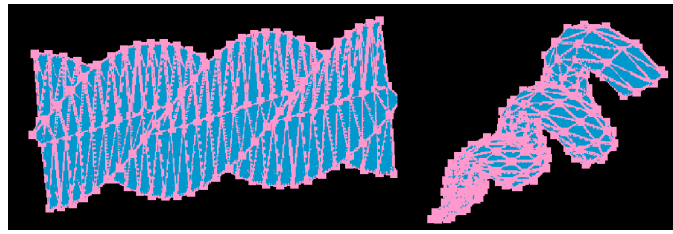


Fig. 3. Extrusion solids: (a) Star. (b) Spiral.

4 CW Complexes and Object Representations in Computer Imagery

4.1 Related Work

In opposition to the classic algebraic topology mainly considering sets with infinitely many elements, computer imagery manipulates sets with finite numbers of points. A first attempt to formulate consistent topology notions equivalent to the continuum notions of neighborhoods is due to [52]. An image is regarded as a graph, called *neighborhood graph*, whose nodes are the pixels and whose edges are linking the adjacent pixels to each other. This notion has the advantage to be intuitive. However, the definition of connectivity and boundary by means of the neighborhood graph fails owed to the so-called *connectivity paradox* and some other contradictions (the Jordan curve paradox, the infinitesimal thin object and the ambiguous boundary definition, [1, 2]).
Substential efforts to overcome these difficulties and further development of discrete topology have been made by [27, 25, 53, 22, 26, 28, 2, 15, 16, 23]. Appropriate definition and application of cellular complexes in computer imagery is presented by [27]. An *abstract cellular complex* is defined as $C = (E, B, dim)$, where E is a set of abstract elements provided with an assymmetric, irreflexive, and transitive binary relation $B \subset E \times E$, called *bounding relation* (or face relation), and with a dimension function $dim : E \rightarrow I$, from E into the set I of non-negative integers such that $dim(e') < dim(e")$ for all pairs $(e', e") \in B$. In $2D$ image processing, $2D$ elements are the surface elements associated with the notion of pixel. $0D$ and $1D$ elements are not represented in the hardware realizations of the image. It is proved that a finite topological space in which at least one smallest neighborhood consists of more than one element is an abstract cellular complex. The simplest open subset of an element e in a complex, called *open star* of e, consists of e itself and all elements bounded by e. An example is given on Fig.4(b). Using the definition of a discrete cellular complex, "spel-manifold" objects, and in general "kλ-manifolds" [23] could be described. Nevertheless, the underlying object definition is limited to the discrete representational domain.

Cellular complex applications in image segmentation are reported in [4, 7, 9]. Discrete objects are described in terms of nD-maps [34] and thus restrained to oriented nD quasi-manifolds. In relation with the cited researches, our approach supports a more general class of discrete objects, the non-manifolds. Thus in image segmentation for example, the region inclusion tree could be calculated allowing a $3D$ region reconsruction.

4.2 *CW* Complexes and Discrete Object Representation

A $3D$ discrete object is regarded as a grid of $3D$ discrete points (see Fig.4). This grid can be used to define the corresponding $3D$ CW complex as follows. A $3D$ discrete point corresponds to a $3D$-cell bounded by six $2D$-cells. Each $2D$-cell being limited by four $1D$-cells and finally, each $1D$-cell being bounded by two $0D$-cells. It should be pointed out that discrete points correspond to non-degenerate cells and the geometric embedment of each iD-cell, $0 \leq i \leq 3$, is explicitly encoded in the grid. For example, let us consider a $1D$-cell, denoted as e. The topological star of e is illustrated on Fig.4(b). The face operators of e encode the $0D$-cells belonging to the star of e and shown on Fig.4(c). The degeneracy operators of e point to the $2D$-cells highlighted on Fig.4(d). In fact,

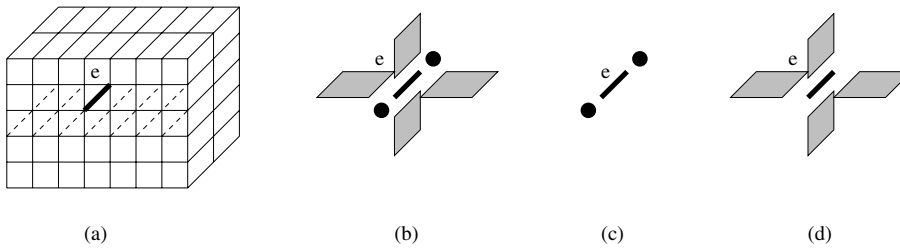


Fig. 4. $3D$ discrete image. (a) Grid. (b) Topological star of the $1D$ cell e. (c) Face operators. (d) Degeneracy operators.

the grid could be regarded as the list of principal $3D$-cells belonging to the object. Further, iD-cells, $0 \leq i \leq 2$, with identical embedment are joined up on the basis of quotient and adjunction operations. The result is a $3D$ CW complex corresponding to the initial object.

This approach has been implemented as a software kernel to reconstruct object boundary in discrete images. We are interested in boundary reconstruction of objects spanning pointsets structured as maximum isovalued connected sets. For this purpose, discrete image is segmented into regions using a homogeneity criterion [9]. The regions of interest are produced as grid of discrete points. Then, the CW complex corresponding to this grid is constructed. In practice, the explicit set of pixels and voxels belonging to the regions allows to replace adjacency determination by detection of a cell presence in a given discrete point. Next,

the inclusion region tree is computed. Finally a traversal procedure reconstructs region boundary.
For the discrete object given on Fig. 5(c), the brain is segmented into regions corresponding to the white matter, the gray matter, the corpus callosum and the hypothalamus. The model supports boundary reconstruction either of each region boundary separately or of the aggregate. The brain outer boundary is illustrated. Model evaluation in terms of total number of iD-cells is given on Table 1. The reconstruction procedure takes $1sec$ CPU time for a Pentium III, 500 MHz, RAM 128 KB.

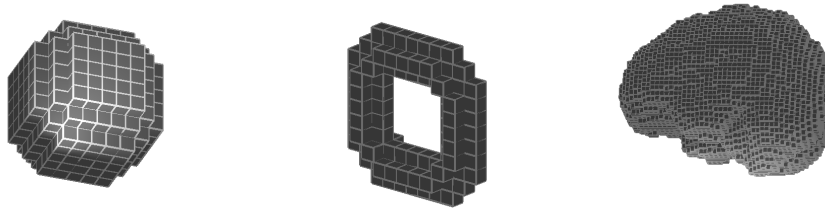


Fig. 5. Examples of object boundary reconstruction for: (a) Sphere. (b) Torus. (c) MRI brain image.

5 Conclusion

Along this paper we have described a framework, based on CW complexes, for the representation and manipulation of solid and discrete objects. The framework is more general than previous approaches with respect to two main points: it provides a unified representation of solid and discrete objects and it enables explicit object degeneracy encoding. We have shown examples illustrating its utility in both geometric modeling and computer imagery. Actual studies go further on cellular subdivision algorithm development and "split and merge"

(1) Solids	$0D$ cells	$1D$ cells	$2D$ cells	Number of tests	CPU time
cube	$36 \rightarrow 8$	$36 \rightarrow 24$	$12 \rightarrow 12$	1250	0.0000001 sec
sphere	$393 \rightarrow 63$	$393 \rightarrow 247$	$131 \rightarrow 131$	60709	2 sec
torus	$702 \rightarrow 96$	$702 \rightarrow 403$	$234 \rightarrow 234$	24677	4 sec
star	$1296 \rightarrow 224$	$1296 \rightarrow 1156$	$432 \rightarrow 432$	23301	21 sec
spiral	$1890 \rightarrow 288$	$1890 \rightarrow 1156$	$630 \rightarrow 630$	63649	35 sec
(2) Discrete objects	$0D$ cells	$1D$ cells	$2D$ cells	3D discrete points	
sphere	138	414	207	485	0.0000001 sec
torus	96	288	144	132	0.0000001 sec
brain	26470	79410	39705	25691	1 sec

Table 1. CW complex model evaluation

operations where the geometric insight of degeneracy cases, supported by the *CW* complexes, will facilitate the study of singular [14] cases and will improve their robustness and efficiency.

References

[1] E. Ahronovitz, J.P. Aubert, and C. Fiorio. Représentation topologique associée à l'analyse d'images. Technical Report 95040, Laboratoire d'Informatique, de Robotique et de Microlectronique de Montpellier, 1995.

[2] E. Ahronovitz, J.P. Aubert, and C. Fiorio. The star-topology: a topology for image analysis. In *5th Discrete Geometry for Computer Imagery, Proceedings*, pages 107–116, September 1995. Groupe GDR PRC/AMI du CNRS.

[3] R. Bidarra, K.J. Kraker, and W.F Bronsvoort. Representation and management of feature information in a cellular model. *Computer-Aided Design*, 30(4):301–313, 1998.

[4] J.P. Braquelaire and L. Brun. Image segmentation with topological maps and inter-pixel representation. *Journal on Visual Communication and Image Representation*, 9(1):62–79, 1998.

[5] P.R. Cavalcanti, P.C.P Carvalho, and L.F. Martha. Non-manifold modelling: an approach based on spatial subdivision. *Computer-Aided Design*, 29(3):209–220, 1997.

[6] G.A. Crocker and W.F.Reinke. An editable non-manifold boundary representation. *IEEE ComputerGraphics&Applications*, pages 39–51, March 1991.

[7] G. Damiand. *Définition et étude d'un modèle topologique minimal de reprsentation d'images 2D et 3D*. PhD thesis, Université Montpellier II, December 2001.

[8] P. Desbarats and S. Gueorguieva. Cw complexes: Topological mainframe for numerical representations of objects. Research report, LaBRI, UMR 5800, Université Bordeaux 1, 2003. RR-1291-03.

[9] Pascal Desbarats. *Structuration d'images segmentées 3D discrètes*. PhD thesis, Université Bordeaux I, 2001.

[10] E. Elter and P. Lienhardt. Cellular complexes as structured semi-simplicial sets. *International Journal of Computational Geometry &Applications*, 1(2):191–217, 1994.

[11] T. Fang and L.A. L.A.Piegl. Delaunay triangulation in three dimensions. *IEEE Computer Graphics and Applications*, pages 62–69, Septembre 1995.

[12] V. Ferrucci and A. Paoluzzi. Extrusion and boundary evaluation for multidimensional polyhedra. *Computer- Aided- Design*, 23(1):40–50, January - February 1991.

[13] L. Floriani, P. Magillo, and E. Puppo. Multiresolution representation of shapes based on cell complexes. In *Discrete Geometry for Computer Imagery*, 1999.

[14] A.T. Fomenko and S.V.Mateev. *Algorithms and Computer Methods for Three-Manifolds*. Kluwer Academic Publishers, 1997.

[15] J. Françon. Discrete combinatorial surfaces. *CVGIP : Graphical Models and Image Processing*, 57(1):20–26, 1995.

[16] J. Françon. On recent trends in discrete geometry in computer science. In *6th Discrete Geometry for Computer Imagery, Proceedings*, pages 3–16. Springer-Verlag Berlin Heidelberg, 1996. Groupe GDR PRC/AMI du CNRS.

[17] L. Fuchs. Une spécification formelle des modèles de courbes et de surfaces de forme libre. Phd thesis, Université Louis Pasteur, 1997. 2791.

[18] L. Fuchs and P.Lienhardt. Topological structures for d-dimensional free-form objects. In *Computer Aided Geometric Design*, 1997.
[19] S. Gueorguieva and D. Marcheix. Non-manifold boundary representation for solid modeling. In *Proc. of the International Computer Symposium*, 1994.
[20] E.L. Gursoz, Y. Choi, and B. Prinz. Vertex-based representation of non-manifold boundaries. In *Geometric modeling for product engineering.*, 1990.
[21] J.H. Hang and A.G.G. Requicha. Feature recognition from cad models. *IEEE Computer Graphics and Applications*, pages 80–94, March/April 1998.
[22] G.T. Herman. Discrete multidimensional jordan surfaces. *CVGIP : Graphical Models and Image Processing*, 54(6):507–515, November 1992.
[23] T.H. Herman. An abstract theoretical foundation of the geometry of digital spaces. In *Discrete Geometry for Computer Imagery*, pages 279–288. Springer-Verlag Berlin Heidelberg, 2002. LNCS 2301.
[24] Ch.M Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann, 1989.
[25] E. Khalimski, R. Koppermann, and P.R. Meyer. Computer graphics and connected topology on finite ordered sets. In *Topology and its applications*, pages 27–55, 1990.
[26] T.Y. Kong, A.W. Roscoe, and A. Rosenfeld. Concept of digital geometry. In *Special Issue on digital topology, Topology Applications*, pages 219–262, 1992.
[27] V.A. Kovalevski. Finite topology as applied in image analysis. *Computer Vision, Graphics and Image Processing*, 46:141–161, 1989.
[28] V.A. Kovalevsky. Digital geometry based on the topology of abstract cell complexes. In *Discrete Geometry for Computer Imagery.* Springer-Verlag Berlin Heidelberg, 1993. LNCS 2301.
[29] V. Kumar, D. Burns, D. Dutta, and C. Hoffmann. A framework for object modeling. *Computer-Aided Design*, 31:541–556, 1999.
[30] T.L. Kunii and S. Takahashi. Area guide map modeling by manifolds and cw-complexes. In *IFIP TC5/WG2 Working Conference on Geometric Modeling*, 1993.
[31] Veronique Lang. *Une étude de l'utilisation des ensembles semi-simpliciaux en modélisation géométrique interactive.* PhD thesis, Université Louis Pasteur de Strasbourg, 1995.
[32] P. Lienhardt. Subdivisions of surfaces and generalized maps. In *EUROGRAPHICS'89*, 1989.
[33] P. Lienhardt. Topological models for boundary representation: A comparison with n- dimensional generalized maps. *Computer-Aided Design*, pages 59–82, January/February 1991.
[34] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry &Applications*, 4(3):275–324, 1994.
[35] P. Lienhardt. Aspect in topology-based geometric modeling: Possible tools for discrete geometry? In *Proc. 7th DGCI, Montepellier, France*, volume 1347 of *LNCS*, pages 33–48, 1997.
[36] A. Losa and B Cervelle. 3d topological modeling and visualisation for 3d gis. *Computer&Graphics*, 23:469–478, 1999.
[37] A.T. Lundell and S. Weingram. *The Topology of CW Complexes.* Van Nostrand Reinhold Company, 1969.
[38] D. Marcheix and S. Gueorguieva. Topological operators for non-manifold modeling. In *Proc. of the Third International Conference in Central Europe on Comput er Graphics and Visualisation 95*, 1995.

[39] David Marcheix. *Modélisation géométrique d'objets non-variétés: construction, représentation et manipulation*. PhD thesis, Université Bordeuax 1, 1996.
[40] H. Masuda. Form-feature representation based on non-manifold geometric modelling. In *MICAD'92*, 1992.
[41] H. Masuda and R. Ohbuchi. Coding topological structure of 3d cad models. *Computer-Aided Design*, 32:367–375, 2000.
[42] H. Masuda and K. Shimada. A mathematical theory and applications of non-manifold geometric modeling. In *IFIP/GI Advanced Geometric Modeling for Engineering Applications*, 1990.
[43] A. Maulik. An efficient intersection algorithm for polyhedral cellular decompositions. In *ACM Symposium on Solid Modelling and Foundations*, 1991.
[44] J. Peter May. *Simplicial Objects in Algebraic Topology*. Chicago University Press, 1967.
[45] M. Mántylá. *Geometric and Solid Modeling: An introduction*. Computer Science Press, 1988.
[46] A.E. Middleditch, C.M.P. Reade, and A.J. Gomes. Set-combinations of mixed-dimension cellular objects. *Computer-Aided Design*, pages 683–694, 1999.
[47] John Milnor. The geometric realization of semi-simplicial complex. *Annals of Mathematics*, 65(2):357–362, March 1957.
[48] A. Paoluzzi, M. Ramella, and A. Santarelli. Boolean algebra over linear polyhedra. *Computer-Aided Design*, 21(8):474–484, October 1989.
[49] F. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.
[50] V. Pascucci, V. Ferrucci, and A. Paoluzzi. Dimension- independant convex-cell based hpc: Skeletons and product. Technical report, University La Sapienza , Roma, Italy, 1995.
[51] D.B. Perng and C.F. Chang. Resolving feature interactions in 3d part editing. *Computer-Aided Geometric Design*, 29(10):687–699, 1997.
[52] A. Rosenfeld and A.C. Kak. *Digital Picture Processing*. Harbour and Brace, 1982.
[53] A. Rosenfeld, T.Y. Kong, and A.Y. Wu. Digital surfaces. *CVGIP : Graphical Models and Image Processing*, 53(6):305–312, July 1991.
[54] J.R. Rossignac. Issues on feature-based editing and interrogation of solid models. *Computer&Graphics*, 14(2):149–172, 1990.
[55] J.R. Rossignac and D. Cardoze. Matchmaker: Manifold breps for non-manifold r-sets. In *ACM Solid Modelling'99*, 1999.
[56] J.R. Rossignac and M.A. O'Connor. Sgc: A dimension- independent- model for pointsets with internal structure and incomplete boundaries. Research report rc14340, IBM T.J. Watson Research Center, 1989. NY 10598.
[57] J.R. Rossignac and A.A.G Requicha. Constructive non-regularized geometry. *Computer-Aided Design*, 23(1):21–32, January/February 1991.
[58] N. Sapidis and R. Perucchio. Solid/solid classification operations for recursive spatial decomposition and domain triangulation of solid models. *Computer-Aided Design*, (10):517–529, October 1992.
[59] Y. Shinagawa, T.L. Kunii, and Y.L. Kergosien. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 1991.
[60] K. Weiler. The radial edge structure: A topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications, IFIP'88*, 1988.
[61] Y. Yamaguchi and F Kimura. Non-manifold topology based on coupling entities. *IEEE Computer Graphics&Applications*, pages 42–50, January 1995.

Approximating 3D General Sweep Boundary Using Depth-Buffer

Jaewoo Ahn and Sung Je Hong

Dept. of Computer Science & Engineering, POSTECH,
Pohang, Gyeongbuk 790-784, Korea
{jaewoo,sjhong}@postech.ac.kr

Abstract. This paper presents a practical technique for approximating the boundary surface of the volume swept out by three-dimensional objects using the depth-buffer. Objects may change their geometries and orientations while sweeping. The sweep volume is approximated as a union of volume elements, which are just rendered inside appropriate viewing frusta of virtual cameras and mapped into screen viewports with depth-buffer. From the depth of each pixel in the screen space of each rendering, the corresponding point in the original world space can be computed. Appropriately connecting these points yields polygonal faces forming polygonal surface patches approximately covering some portion of the sweep volume. Each view frustum adds one or more surface patches in this way, and these presumably overlapped polygonal surface patches approximately enclose the whole sweep volume. These patches may further be processed to yield non-overlapped polygonal surfaces as an approximation to the boundary of the original 3D sweep volume.

1 Introduction

Sweeps whose generating area or volume changes in size, shape, and orientation as they are swept and that follow an arbitrary curved trajectory are called *general sweeps* [6]. General sweeps of 2D shapes and 3D solids have attracted much research attention in various application fields, including computer graphics, CAD/CAM, and robotics, etc. Some of the works define, analyze and represent sweep volumes according to appropriate mathematical structures and criteria [1,4,5,12,13,17], and some adapt and utilize the sweep operation for solving real problems [7,9,10,11]. There are also works for efficient rendering of objects defined by general sweeps [16,18].

[17] represents the sweep of a moving polyhedron in terms of the sweeps of its faces and generates its sweep boundary as the union of the ruled surface patches generated by the edges and the developable surface patches generated by the faces. [18] demonstrates a ray tracing algorithm to display the general sweep volume of sphere moving along 3D curve trajectory while changing its radius dynamically. [9] attempts to simplify the expensive computation by sweeping three-dimensional objects in image space instead of in 3D object space. This approach represents 3D objects in image space as depth elements, called *dexels*,

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 508–517, 2003.
© Springer-Verlag Berlin Heidelberg 2003

and performs boolean operations on these dexels. While the use of dexels is sufficient for simple modeling and graphical display purposes, there are many cases that need the geometry of the resulting 3D object of a sweeping.

Due to inherent difficulties in computing the boundaries of these general sweeps, researches so far have placed restrictions on object shapes, shape changes, and sweep trajectories; nevertheless, they have involved a fair amount of symbolic or numerical computations that have limited their practical uses in graphics modeling systems as well as in many other applications which require fast sweep boundary computation. Although the technique presented here does not generate the exact object sweep boundaries, nor is complete yet, it suggests a way to get quite reasonable polyhedral approximations to them that are reasonably fast to be of practical uses. Related with the work in [2], which deals mainly with approximating 2D general sweep boundaries, see Figure 1, this paper may provide a suite of techniques for approximating 2D and 3D general sweep boundaries.

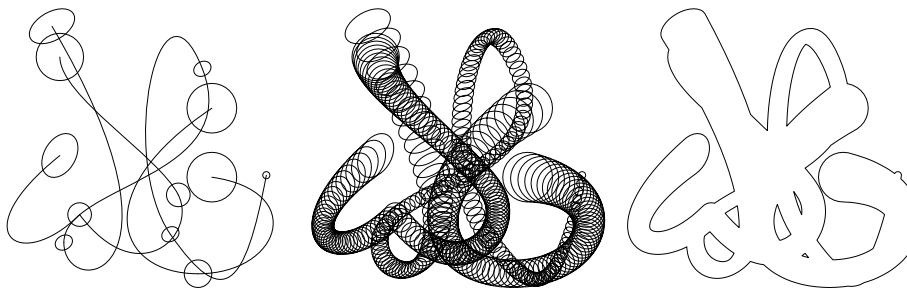


Fig. 1. An example of 2D general sweep and its boundary.

In the rest of the paper, we consider the problem of approximating sweep volumes of solid objects in a practical point of view, and then present a technique for approximating the boundary surface of the sweep volume using the depth-buffer, followed by a few illustrative examples. Discussions and current limitations on using the technique follow, and then we conclude the paper.

2 Approximating Sweep Volumes of Solid Objects

Given a description of an object whose geometry changes dynamically as a function of time, and a trajectory along which the object moves, we consider how to approximate the boundary surface of the volume that is swept out by the object while it moves along the trajectory. Let $S(t)$ describe the geometry of the sweeping object at time t, and $C(t)$ be the trajectory along which the reference point of the object traces, for $t \in [a, b]$. Without loss of generality, we may assume that $S(t)$ represents the set of points on the object with its reference point at $C(t)$ at time t. Further we may assume that $S(t)$ is connected, closed, and

bounded. We denote the volume swept out by $S(t)$, for $t \in [a, b]$, as $S(a, b)$. That is, $S(a, b) = \cup_{t \in [a,b]} S(t)$.

To approximate the sweep volume within a given error bound, the trajectory is subdivided into sufficiently small segments so that the changes in the object geometry and position are sufficiently small along each segment, and the object geometry is appropriately approximated as a polyhedron at each time instance of the subdivision of the trajectory. The sweep volume of a polyhedral object can be reduced to sweep volumes of its polygonal faces plus an instance of the object at any time during the sweep [17]. And the sweep volume of a polygonal face along a piecewise linear trajectory can be approximated as the union of volume elements, or convex hulls, of two consecutive corresponding face instances at ends of linear segments along the piecewise linear trajectory. This is a direct extension of the arguments in [2] from two-dimensional case into three-dimensional one.

As a whole, the sweep object is sampled at appropriate instances along the trajectory, and each instance is approximated as a polyhedron. Then, the sweep volume is approximated as the union of the polyhedron at the initial position and all the convex hulls of two corresponding faces of two consecutive polyhedra along the trajectory.

3 Approximating Boundary Surface of Sweep Volumes Using Depth-Buffer

Assume a sweep volume in space swept out by some solid objects along their own trajectories. Taking snapshots of the volume from virtual cameras around it would result in projected images of the volume in the imaging planes of the cameras. If a ray is cast from each of these cameras through the center of each pixel of the projected image of the volume in the imaging plane, it would hit the surface of the sweep volume in space. If we know the length of the ray from the camera to the first hit point, then we can figure out the coordinates of the point in space from the camera and its imaging plane geometry. In other words, this is just the point sampling of the surface of the sweep volume, using the ray casting method from cameras around the volume. By appropriately connecting these points in space, we construct polygonal faces, which form one or more polygonal surface patches and cover some portion of the sweep volume. If any of these polygonal faces goes beyond a given error bound from the actual sweep volume, its corresponding camera resolution may be enhanced to render a finer image of the sweep volume in the imaging plane, and the procedures above may be repeated again. By placing appropriate virtual cameras around the sweep volume, we can enclose the whole sweep volume, within a given error bound, inside a number of polygonal surface patches, which are presumably overlapped to cover the whole sweep volume without missing parts.

As described in the previous section, by sampling a sweep object at appropriate instances along its trajectory, and by approximating each instance as a polyhedron, its sweep volume may be approximated as the union of the polyhedron at the initial position and all the convex hulls of two corresponding faces

of two consecutive polyhedra along the trajectory. Then, the sweep volume assumed above can be approximately rendered inside the view frustum of each camera just by rendering a number of polygonal faces.

For the distance, or depth, information of each pixel of a rendering, we need to cast a ray from the camera to the center of the pixel in the imaging plane in space, to find the first hit point of the ray with the sweep volume. This process can be carried out quickly by utilizing a graphics hardware component which is supported by most of the modern graphics hardware systems, namely the *depth-buffer*, or the *Z-buffer* [6]. After rendering the sweep volume inside an appropriate view frustum, and mapping the image on an appropriate screen viewport, we examine the information of all the pixels from the frame-buffer and the depth-buffer memories of the graphics system, which let us know enough of the information that we may get from actual ray casting. Also by examining the frame-buffer, we can tell those pixels that are actually rendered by the sweep volume. The transformation from pixel coordinates of the buffers back into the world coordinates can be computed from the camera model used for the rendering and the resolution, or size, of the screen viewport. Here, we may implement and use a software depth-buffer, but since there are so many faces to render, hardware-accelerated rendering with depth-buffering is preferred. Note that we can easily enhance the result by just increasing the viewport size of cameras, or the depth-buffer resolution, which is increased either by using more hardware bits for storing depth values or by specifying the near-plane and the far-plane of each camera model as close as possible.

Given a screen viewport and a camera view frustum, the real coordinates of the closest point in space that causes the rendering of an image pixel can be computed by using the inverse of the transformation the graphics pipeline uses when rendering [6,14,19]. The boundary point of the sweep volume corresponding to each pixel can be approximated by applying the inverse transformation to the center of the pixel. Note that the depth-buffer is cleared, or set to the maximum distance value, before rendering. For those pixels whose depth values have not changed by the rendering, we consider there are no corresponding points in the sweep volume that get mapped to the center of them, at least from the viewpoint of the current camera. For the other pixels, there are surely a part of the sweep volume that gets mapped to the center of each of them, and their corresponding points surely belong to the boundary surface of the sweep volume, as far as the near-plane of the camera frustum does not clip the sweep volume. For two neighboring pixels, if their depth values are not close enough, they may belong to different parts of the sweep volume, even if their screen coordinates are neighboring. This closeness, however, does not seem to be measured and determined easily in advance, since it requires minute examination and analysis of the whole sweep volume. To be simple, we may just specify a threshold value, which is controllable for each camera, to determine whether two computed points are neighboring on the surface of the sweep volume when they are neighboring in the rendered image of the camera. Three or more neighboring points, thus obtained, may form a polygonal face on the boundary surface of the sweep volume.

From each camera and its rendering, we get a number of polygonal faces, which form one or more polygonal surface patches and cover some portion of the sweep volume. By placing a number of cameras appropriately around the whole sweep volume, we may get polygonal surface patches which are slightly overlapped with each other and cover the whole sweep volume together. Further, we may get a non-overlapped set of polygonal faces from these overlapped polygonal surface patches using, for example, the method in [8].

If the image size, or the resolution, of a rendering is not fine enough, there may be some spatial aliasing problem, in which several points belonging to different parts of the sweep volume form a polygonal face, resulting in losses of some detailed geometries of the sweep volume under the generated polygonal face. The image size of a rendering may be somewhat appropriately determined automatically in proportion to the projection area of the camera in world coordinates. The size may further be increased to yield more accurate result for each camera, especially for those cameras rendering complex or intersecting parts of the sweep volume.

The number of bit-planes used for depth-buffering usually exceeds 24 in most of the modern graphics hardware systems, which is practically much accurate. Thus, those computed polygonal surface patches almost exactly overlap with each other in the depth direction where they overlap. To fully utilize the depth-resolution of the graphics hardware for even better result, it is recommended to specify the depth range of each camera as compact as possible, which is set by the near and the far clipping planes of the view frustum of the camera.

Examples in this paper use ellipsoids as solid sweep objects, since they are easy to specify and tessellate [3], but any solid object which can be approximated as a polyhedron, and in which we can find corresponding points and faces of any two approximated neighboring instances along the trajectory, may be used. Even the shape of the sweeping object may change, as long as they satisfy the former two conditions. Perspective view frusta may also be used instead of orthographic view frusta, if necessary.

4 Examples

Figure 2 is a simple example illustrating the technique discussed in this paper. 2(a) shows a simple trajectory with a sweep object sampled at a few instances along the trajectory, and 2(b) shows the actual sweeping. 2(c) is the sweep volume whose boundary surface is to be approximated as a polyhedron. Each box in 2(d) represents the view frustum of a virtual camera, where far clipping plane is toward the sweep volume and mostly clipping the volume. There are 10 virtual cameras used to snapshot the sweep volume and to construct polygonal surface patches of 2(e) which overlap with each other and enclose the whole sweep volume. 2(f) and 2(g) are the same as 2(e), but generated with different resolutions of virtual cameras. 2(h) shows a polyhedral approximation to the sweep volume in 2(c) constructed thus.

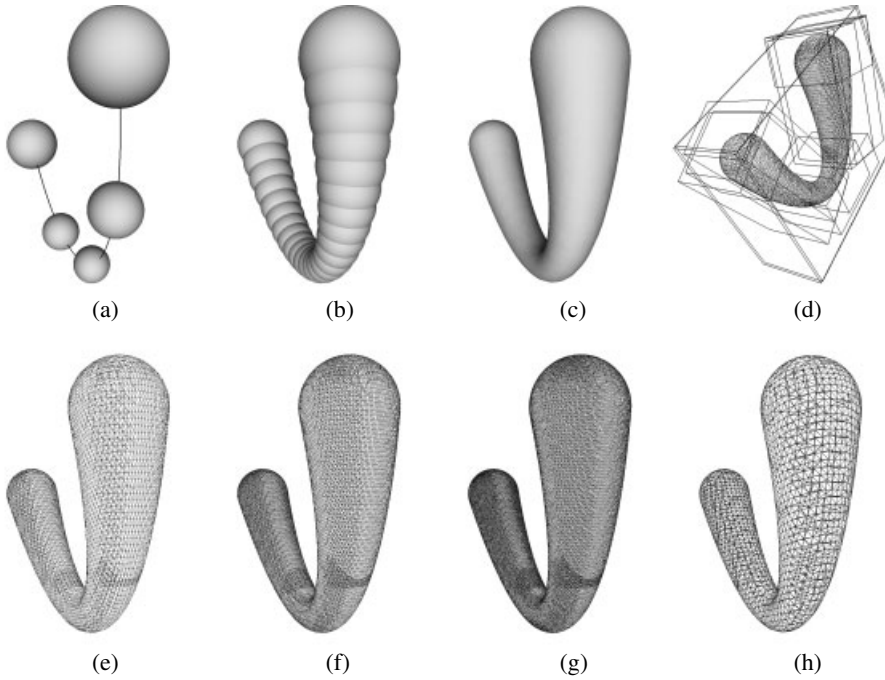


Fig. 2. A simple example of three-dimensional sweep.

Figure 3 is another example with intersecting parts. Rendering resolutions may be somewhat increased for those virtual cameras around the intersecting parts of the sweep volume, to cleanly and accurately depict the boundary surface of the sweep volume as in 3(c). Note the placement of virtual cameras used to snapshot the sweep volume, especially those around intersecting parts, as in 3(b). 3(e) is a close-up view around an intersecting part. In some cases, intersections may be just ignored to generate locally smooth surfaces as in Figure 4(f). That is, we consider intersecting parts separately, see 4(b) and 4(d). But global processing like polygon intersection is necessary for the final result.

Figure 5 is a somewhat intricate example, where 33 virtual cameras are manually placed around the sweep volume, as in 5(c), and Figure 6 is a somewhat more intricate example, where a number of virtual cameras are automatically placed around the sweep volume. The local reference frame of sweep object is as in 6(a) along with the trajectory, and these are used to generate the placement of virtual cameras in 6(c).

5 Discussions and Future Works

In this section, we discuss a few additional issues pertinent to the technique of previous sections and an implementation of the technique, and consider future works.

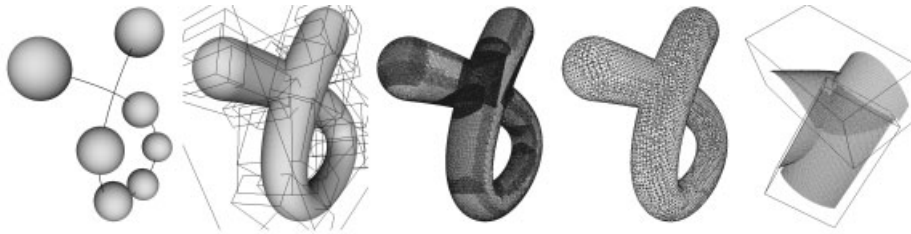


Fig. 3. An example of three-dimensional sweep with intersecting parts.

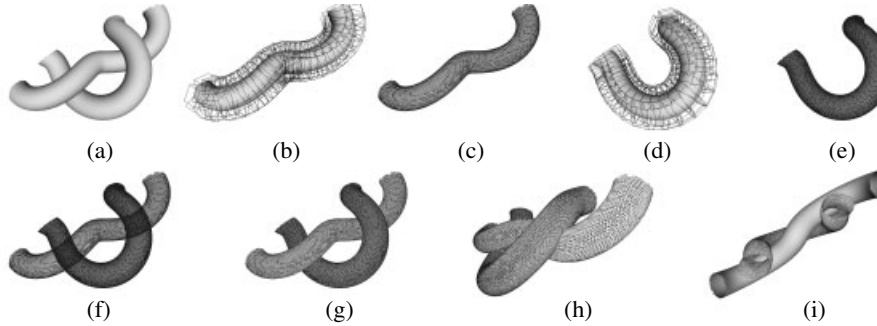


Fig. 4. Intersections may be ignored to generate locally smooth surfaces.

First of all, the frame-buffer is not necessary at all for renderings of virtual cameras. Only the depth information of each pixel from the depth-buffer is necessary for the construction. Thus, the frame-buffer may be disabled while the depth-buffer is enabled during rendering.

View culling may also greatly reduce rendering time. Only the sweep object instances or the volume elements lying inside the view frustum of, or visible from, each virtual camera, are rendered. Since there may be so many virtual cameras and object instances to consider, simple and fast culling would be better than complex and accurate one.

When placing a virtual camera around the sweep volume, some interior parts of the sweep volume may be rendered because of the clipping by the near plane of the camera view frustum. Some of these interior parts do not really belong to the boundary surface of the sweep volume, and the generated polygonal faces from these visible interior parts have wrong orientations. Thus, the virtual cameras should not be allowed to be placed in such a way that any of their near clipping planes clip parts of the sweep volume. After placing a camera appropriately, however, this may be easily avoided by adjusting the six clipping planes of the view frustum.

To reduce approximation errors, resolutions in the depth direction as well as in the horizontal and the vertical direction may be increased. These correspond to increasing the number of bit-planes for the depth-buffer and the rendering image

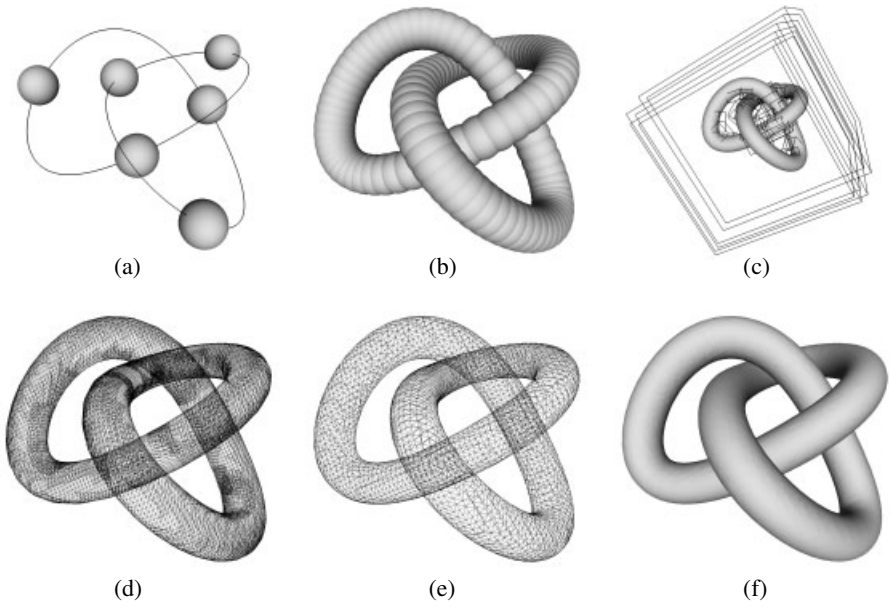


Fig. 5. An example of manual placement of virtual cameras (a trefoil knot).

resolutions of cameras. Adjusting camera placements or adding new cameras in appropriate places would also enhance the result.

Forming complete two-manifold meshes or to fit smooth surfaces out of overlapped polygonal surface patches enclosing the whole sweep volume may be done by referring the result of [8], etc. However, knowing the characteristics of the geometries under consideration, we may devise more suitable schemes for this. The characteristics are that only the neighboring meshes are overlapping with each other, that they almost exactly overlap in the depth direction where they overlap, and that they form a complete covering of a solid volume. Automatically and optimally placing virtual cameras around the sweep volume is not an easy work, but would be necessary in the end. These are considered as future works. For example, [15] considers the problem of placing cameras to view all visible surfaces in a scene. If these problems of automation and optimization are solved to some extent, the problem of finding general sweep boundary which has fundamental complexities and mathematical limits would be transformable to these problems, as far as approximation is concerned.

Reliability is one of the most fundamental difficulties in computational geometric operations, where finite-precision arithmetic is indispensable. After all, most geometric operations, including sweep boundary computation, have produced only limited results because of the complex mathematics involved. By using a technique similar to the one considered in this paper, more reliable geometric operations can be devised which are supported and accelerated by most of current computer hardware systems. Among these would be the problems of

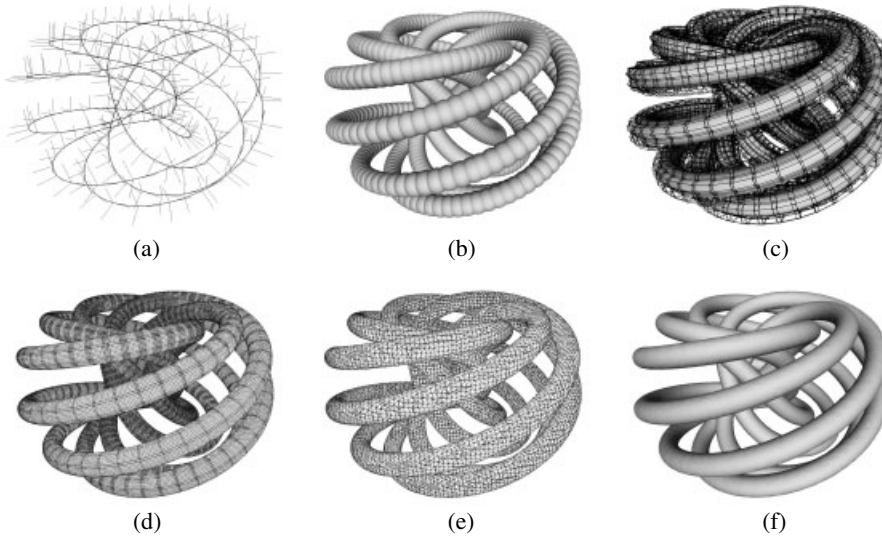


Fig. 6. An example of automatic placement of virtual cameras (a torus knot).

approximating the intersections between geometric objects, as well as approximating the boundary curves of two-dimensional general sweeps.

6 Conclusions

In this paper, we have considered a practical technique for getting a polyhedral approximation to the boundary surface of a three-dimensional general sweep volume. Computation can be accelerated by utilizing the depth-buffer component of hardware graphics system. By generating a number of overlapped polygonal surface patches enclosing the whole sweep volume, within a given error bound, and then generating a smooth polyhedral surface from these overlapped patches, we can quickly approximate the boundary.

This is a reliable and practical approximation technique applicable to the three-dimensional general sweep problem, with the approximation error and the computation time reducible simply at the cost of enhancing system environments or changing computation parameters. The approximation error mainly depends on the rendering accuracy, which is controllable by changing rendering resolutions either in the frame-buffer or in the depth-buffer, the number of instances sampled along the trajectory, and the accuracy in approximating each of these instances. The computation time can be greatly reduced just by upgrading the graphics hardware system, as well as by appropriately reducing the accuracy of the result by changing computation parameters. Moreover, sweep volumes may be much complex, and the approximation technique is still applicable.

References

1. K. Abdel-Malek, and H. J. Yeh, "Geometric Representation of the Swept Volume Using Jacobian Rank-Deficiency Conditions," *Computer-Aided Design*, 29(6):457–468, 1997.
2. J. Ahn, M.-S. Kim, and S.-B. Lim, "Approximate General Sweep Boundary of a 2D Curved Object," *CVGIP: Graphical Models and Image Processing*, 55(2):98–128, March 1993.
3. J. Ahn, "Fast Generation of Ellipsoids," *Graphics Gems V*, pp. 179–190, AP Professional, Boston, 1995.
4. D. Blackmore, M. C. Leu, and F. Shin, "Analysis and Modeling of Deformed Swept Volumes," *Computer-Aided Design*, 26(4):315–326, 1994.
5. D. Blackmore, M. C. Leu, and L. P. Wang, "The Sweep-Envelope Differential Equation Algorithm and Its Application to NC Machining Verification," *Computer-Aided Design*, 29(9):629–637, 1997.
6. J. D. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice, 2ed*, Addison-Wesley, Reading, MA, 1990.
7. M. Ganter, "Dynamic Collision Detection Using Kinematics and Solid Modelling Techniques," Ph.D Thesis, Dept. of Mechanical Engineering, Univ. of Wisconsin - Madison, August 1985.
8. H. Hoppe, T. DeRose, T. DuChamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," *Proceedings of SIGGRAPH 1992*, 26(2):71–78, July 1992.
9. K. C. Hui, "Solid Sweeping in image space - application in NC simulation," *The Visual Computer*, 10:306–316, 1994.
10. J. Kieffer, and F. L. Litvin, "Swept Volume Determination and Interference Detection for Moving 3-D Solids," *ASME Journal of Mechanical Design*, 113:456–463, 1990.
11. J. Korein, "A Geometric Investigation of Reach," An ACM Distinguished Dissertation 1984, The MIT Press, Cambridge, Massachusetts, 1985.
12. R. R. Martin, and P. C. Stephenson, "Sweeping of three-dimensional objects," *Computer-Aided Design*, 22(4):223–234, May 1990.
13. K. Sambandan, "Geometry Generated by Sweeps of Polygons and Polyhedra," Ph.D. Thesis, Cornell Univ., August 1990.
14. D. Shreiner, OpenGL ARB, and D. Schreiner, *OpenGL Reference Manual, 3ed: The Official Reference Document to OpenGL, Version 1.2*, Addison-Wesley, Reading, MA, 1999.
15. W. Stuerzlinger, "Imaging all Visible Surfaces," *Proceedings of Graphics Interface*, pp. 115-122, 1999.
16. W. P. Wang, and K. K. Wang, "Geometric Modeling for Swept Volume of Moving Solids," *IEEE Computer Graphics and Applications*, 6(6):8–17, 1986.
17. J. D. Weld, and M. C. Leu, "Geometric Representation of Swept Volumes with Application to Polyhedral Objects," *The International Journal of Robotics Research*, 9(5):105–117, October 1990.
18. J. van Wijk, "Ray Tracing Objects Defined by Sweeping a Sphere," *Proceedings of Eurographics '84 Conference*, Amsterdam, pp. 73–82, 1984.
19. M. Woo, J. Neider, T. David, D. Shriner, T. Davis, OpenGL ARB, and D. Shreiner, *OpenGL 1.2 Programming Guide, 3ed: The Official Guide to Learning OpenGL, Version 1.2*, Addison-Wesley, Reading, MA, 1999.

Morphology-Based 3D Volume Metamorphosis

Tong-Yee Lee, Chao-Hung Lin, and Wen-Hsiu Wang

Department of Computer Science and Information Engineering,
National Cheng-Kung University, Tainan, Taiwan, Republic of China
tonylee@mail.ncku.edu.tw
http://couger.csie.ncku.edu.tw/∼vr/3dmorph.html

Abstract. In this paper, we present a morphology-based scheme to perform 3D volume morphing. Our proposed scheme is a non-feature-based metamorphosis technique. This method avoids tedious feature specifications to control shape transitions. However, the proposed scheme yields a satisfactory morphing sequence for several examples evaluated in this paper. The proposed method consists of three steps: (1) create a morphological difference object volume (MDOV), (2) and perform both dilation-based distance transform, and (3) erosion-based interpolation on MDOV to accomplish volume morphing. Furthermore, with combination of a well-known principal axes registration (PAR) technique, the proposed method can smoothly morph two objects with different orientations. Finally, we present some interesting preliminary experimental results to evaluate the proposed method.

1 Introduction

Lazarus et al. [1] gave thorough surveys of previous work on 2D and 3D morphing problems. Techniques for 3-D object morphing generally divide into two categories: volume-based and mesh-based. In this paper, we will propose a morphology-based transform to establish volumetric metamorphosis. In case of the exploitation of the proposed method to 3D polygonal objects, we will voxelize given 3D objects before morphing. In the following, we will briefly review most related prior work in the category of volume-based morphing.

The proposed morphology-based transform scheme is a kind of the signed distance function and is employed to interpolate object shapes. Several previous efforts have exploited the signed distance function to execute interpolation between 2D contours [2,3]. A classic shaped-based scheme [4,5] is designed by encoding the segmented image with a signed Euclidean distance. The intermediate contour is created by the linear interpolation between corresponding distances of two given segmented images. However, this method does not consider the global geometric changes in shape of the object; therefore, it cannot deal effectively with objects with large offsets and holes. In contrast, our proposed method has been successfully applied to the biomedical reconstruction applications and has been experimentally proved to perform better than the shape-based scheme. For detailed comparisons and experimental evaluations, please refer to our previous

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 518–527, 2003.
© Springer-Verlag Berlin Heidelberg 2003

work [6]. In this paper, we will extend this work to handle 3D volume morphing. Payne et al. employed a shaped-based like scheme to transform 3D shapes [7], too. Cohen-Or et al. [8] also proposed to use a DFI (distance field interpolation) in combination of warping to transform from a 3D object to another. In fact, their DFI transform is the exact shape-based distance transform via a chamfer distance mask. Lerios et al. [9] described a powerful feature-based image morphing method to volumetric representation. Several "feature elements pairs" are used to define the transformation of two volumetric representations. Lee et al. [10] also extended this work to design several interesting operators such as tear and cut to 3D volume metamorphosis.

2 Methodology

2.1 The Basic Idea of Morphology-Based Volume Morphing

Among morphological operations [11], *dilation* and *erosion* are used to perform 3D volume morphing in this paper. Three-dimensional object volume morphing can be stated as follows. Given two objects X_0 and X_{n+1} on 3D volume space, we attempt to interpolate a sequence of intermediate objects (e.g., n intermediate objects, X_t and $1 \leq t \leq n$) so that the adjacent pairs in this sequence are very close to one another in geometrical appearance. The proposed method consists of three steps: (1) create a morphological difference object volume (MDOV), (2) and perform both dilation-based distance transform, and (3) erosion-based interpolation on MDOV to accomplish volume morphing. The basic idea of the proposed method in 2D can be schematized in Figure 1 and 2.

First, we overlap two input objects by aligning their centroids in volume space as shown in Figure 1. With this overlapping, we create a working object volume called morphological difference object volume (MDOV). In this figure, the boundaries of objects X_0 and X_{n+1} are represented by the dashed line and dotted line, respectively. On this MDOV, three kinds of regions are possible: I, II and III. Both regions I and II represent the morphological difference (MD) area between X_0 and X_{n+1} and III represents $X_0 \cap X_{n+1}$. Second, we apply dilation operation only to both regions I and II instead of the entire volume space. The dilation starts from the boundary of the region III toward boundaries of both regions I and II. To implement a dilation operation, a structure element or called convolution mask is always used [11]. See an example of dilation on the region I in Figure 2. In this figure, to start dilation on the region I, initially, each boundary pixel of the region III is assigned a structure element. Then, each mask begins to dilate toward the boundary of the region I, and the mask will update the value stored on the pixel. In the end of dilation, the value stored on each pixel P is called dilation-based *distance* that denotes the minimum number of dilations from the boundary of the region III to P. The distance stored on the boundary of the region III is all zero. After dilation task, on the regions I and II, if the distance of voxels is identical, we will say these voxels are on the same layer of the distance map. A voxel with distance D is defined to be located on the D^{th} layer of the distance map.

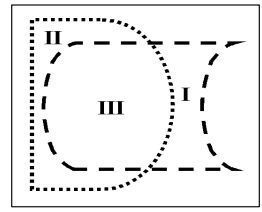


Fig. 1. Morphological difference image (MDI) between contours X_0 and X_{n+1} (i.e., X_0 defined by dashed line boundary and X_{n+1} defined by dotted line boundary). In this figure, region I $= X_0 - (X_0 \bigcap X_{n+1})$, II $= X_{n+1} - (X_0 \bigcap X_{n+1})$ and III $= X_0 \bigcap X_{n+1}$.

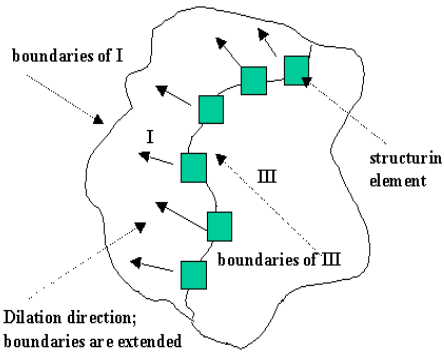


Fig. 2. During dilation, the structure elements will gradually (layer by layer) assign a distance to the region I from the boundary of region III toward the boundary of region I On the other hand, during erosion, the structure element will be placed on the boundary of I and then will gradually contract the boundary toward the boundary of region III.

Dilation creates 3D distance maps containing distance of MDOV. In the last step, the erosion will be operated on these maps to perform interpolation from region I or II to III. Assume we will totally interpolate n intermediate objects between two given 3D objects. Then we define E_k^n and $E_k^{n^*}$, called erosion factors, as follows:

$$E_k^n = \frac{k}{n+1}, \quad E_k^{n^*} = 1 - E_k^n, \quad 1 \leq k \leq n. \tag{1}$$

Where E_k^n is used for the region I, $E_k^{n^*}$ is used for the region II and k means the k^{th} object. Then, the interpolated object X_k is defined by equation (2).

$$\begin{aligned} X_k &= Erosion(X_0, E_k^n) + Erosion(X_{n+1}, E_k^{n^*}) \\ &= Erosion(\mathrm{I}, E_k^n) + Erosion(\mathrm{II}, E_k^{n^*}) + \mathrm{III}. \end{aligned} \tag{2}$$

The Erosion() procedure performs exact interpolation and it is accomplished by an erosion structure element. For the X_k, we apply this procedure to I and II regions with erosion factors E_k^n and $E_k^{n^*}$, respectively, and then we unite them with III region to obtain the object X_k. Erosion function takes two parameters: region I or II and an erosion factor E that is a real number varying from 0 to 1. The erosion starts from the boundary of I and II toward the boundary of III. Assume a voxel P is located on the boundary of I or II and P's distance is D (i.e., P is located on the D^{th} layer). Then, to accomplish Erosion(), the number of erosions for P is $Integer$(D*E), where $Integer()$ will return the maximum integer that is less than or equal to D*E. Under this situation, P will erode layer by layer in the sequence, D, D-1, ...D-$Integer$(D*E). After erosion, P is located on the (D-$Integer$(D*E))th layer. At each erosion, a voxel located on the N^{th} layer seeks the nearest voxels located on the (N-1)th layer for possible erosion. Recall in the first step that we align centroids of two objects in volume space to get a MDOV. Assume that we have two objects X_0 and X_{n+1} and their original centroids are O_0 and O_{n+1}, respectively. After interpolation we compensate for the effects of this object centralization by translating the interpolated object X_k^n(i.e, X_k) back to the correct position, computing the new centroid of X_k^n , O_k^n according to equation (3).

$$O_k^n = O_0 * E_k^{n^*} + O_{n+1} * E_k^n. \tag{3}$$

After finishing above step, we finally can obtain an intermediate X_k^n object. In the following subsections, we will fill the missing details and implementation details about dilation, and erosion.

2.2 Dilation-Based Distance-Maps

The dilation-based distance transform on MD area is accomplished by means of a structure element. We use a 3x3x3 cubic structure element to assign distance to MD area. To store distance value, we create two 3D array distance-maps: A_0 and A_{n+1} for region II (i.e., $X_{n+1} - (X_0 \bigcap X_{n+1})$) and region I (i.e., $X_0 - (X_0 \bigcap X_{n+1})$), respectively. Let ∂X_0 and ∂X_{n+1} denote the boundaries of X_0 and X_{n+1}. We initialize A_0 and A_{n+1} as follows: (1) initialize all elements of array A_0 and A_{n+1} to be -1, (2) set elements A_0(x,y,z) to be 0, if the voxel (x,y,z) of MDOV $\in X_0$ and (3), set elements A_{n+1}(x,y,z) to be 0, if the voxel (x,y,z) of MDOV $\in X_{n+1}$. To help easy implementation of dilation, we use a FIFO (first-in-first-out) queue data structure and each element M in this queue includes 6 fields (x, y, z, a, b, d) where (1) voxel's (x,y,z) coordinates, (2) a, b are two distance map identifiers (i.e., 0: A_0 or n+1: A_{n+1}) and (3) d is distance code stored at voxel (x,y,z). Note a voxel (x,y,z) will store its d at distance map a(x,y,z) (i.e., A_0 or A_{n+1} depends on the identifier a). Then, we execute the pseudo code as shown in Figure 3. Using this pseudo code, we simultaneously fill distance code layer by layer (i.e., controlled by an index) on 3D arrays A_0 and A_{n+1}. When a node N (x,y,z,a,b,d) is deleted from the queue L_D, we suppose now a 3x3x3 cubic is centered at voxel (N.x, N.y, N.z) of MDOV. Next, it will

```
3D_Dilation_Based_Distance_Transform( )
{
/* a. dilation queue L_D initialization */
  index ← 0  /* index denotes layer no. */
  for each voxel (x,y,z) of ∂X_0
    do we insert a new node M(x,y,z,0,n+1,0)
       into the dilation queue L_D;
  for each voxel (x,y,z) of ∂X_{n+1}
    do we insert a new node M(x,y,z,n+1,0,0)
       into the dilation queue L_D

 /* b. enter dilation process */
 while L_D ≠∅ (i.e.not empty)
 {
    do delete the first node of the queue L_D and denoted as N(x,y,z,a,b,d)
      /* c. the same dilation layer */
      if N.d = index
        /* d. check if N's 6-neighbors could be updated or not
              in the next dilation layer (e.q. dilation from N) */
           then for each six-neighbor P(x',y',z') of voxel (N.x,N.y,N.z) on MDOV
              /* e. check if it is inside another object; */
              do if P∈ X_b
               then if A_a(P) ≥ 0 and A_a(P)≤index+1
               /* A_a(P) denotes distance stored on distance map A_a(x',y',z'), where a
                is distance map id */
               /* there exists another shorter path to P in MDOV,
               therefore, we will not update A_a(P); */
                          then no update on A_a(P) and L_D
                          else insert point P into the tail of the list
                               L_D with (x',y',z', a, b, index+1 )
                               update A_a(P) ← index + 1
        /* f. start next dilation layer from node N; N was retrieved once in the previous
           layer, so we put N back */
        else index ← index + 1
             insert node N back to the first position of the list L_D
   }
}
```

Fig. 3. Dilation-based Distance Transform.

check its 6-neighbors at MDOV, say any its 6-neighbor, P(x',y'z'). If N.d+1 is less than A_a(P), we will update distance stored at map A_a(x',y'z') by N.d+1 and insert P (i.e., for the next layer) in the queue. If above condition is not true, it means there exists another shorter path than from voxel (N.x, N.y, N.z) to P(x',y'z'). Therefore, there is no update on A_a(P). The whole dilation will be repeated until there is no A_a(P) with -1 within morphology difference region (part e).

2.3 Erosion-Based Interpolation

After dilation step, there are two 3-D arrays created: A_0 and A_{n+1}. Next, we will create another 3D array called B. Recall that A_0 and A_{n+1} store distance code for the morphology difference (MDOV). This extra B array is used to store the number of erosions for the boundary of the morphology difference. In our implementation, B array is not a floating-point array but it is rounded integer array. We initialize B array in following steps: (1) initialize 3-D B array with ∞, (2) store -1 on each element of B for the morphology-difference defined by

$X_{n+1} - (X_0 \bigcap X_{n+1})$ and $X_0 - (X_0 \bigcap X_{n+1})$, and (3) store the distance code of a voxel (x,y,z) that is belonging to both ∂X_0 and ∂X_{n+1} on A_{n+1}(x,y,z) and A_0(x,y,z) to B(x,y,z). Next, we will execute the pseudo code as shown in Figure 4 to interpolate any in-between object X_k (i.e., X_k^n). The structure of this code is quite similar to that of dilation step. A similar queue called L_E is used, too. Initially, we insert boundary voxels of ∂X_0 and ∂X_{n+1} into this queue. Next, to interpolate object X_k, we calculate exact number of erosions termed $D_{erosion}$ and use $D_{erosion}$ to update nodes (x,y,z,a,b,d) in the queue and their corresponding B(x,y,z). Similar to pseudo code of dilation, a cubic structure element is used for the purpose of erosion. Therefore, in the pseudo code, when this 3x3x3 element is centered at voxel (N.x,N.y,N.z), this element will check its 6-neighbors, say any one termed P(x',y',z') for further erosions. In the current implementation, two cases are considered: (1) normal case and (2) special case. For the former, the legality of erosion depends on two conditions: (a) A_a(N.x,N.y,N.z) = A_a(P)+1; on distance map A_a,, both are located at the consecutive layers, and (b) N.d-1 > B(P). When both conditions hold, then we insert P with (x',y',z',a,b,N.d-1) into queue L_E. Otherwise, we do not need to insert P in L_E, since the consecutive erosions from P are totally included in the other longer and consecutive erosions. Therefore, we do not need to insert P in L_E, since we do not need to repeat erosions that will be included by the other node. For the latter case, if A_a(N.x,N.y,N.z) = A_a(P), then we will further check P's 6-neighbors, say any one termed Q. When all 6-neighbors satisfy $A_a(P) \geq A_a(Q)$, then we insert P with (x',y',z',a,b,N.d) into queue L_E. After above pseudo code, the interpolated contour X_k^n is composed of a set of voxels with zero distance on map B (i.e., we finish equation (2)). Finally, we will use equation (3) to translate X_k^n to corrected position and then finish interpolation.

In [8], Cohen-Or et al. claimed that an appropriate rotation transformation to original input source yields better morphing transition. In their approach, many features must be specified manually to find such a rotation transformation. However, for 3-D cases, this time-consuming effort could not make this approach practical. Alternatively, we adopted a method called principal axes registration (PAR) that usually is used in 2-D or 3-D image registration applications [12]. We will include this technique as an option for our 3D morphing. We summarize our adoption as follows. Since the 3D object is converted into binary discrete volumetric representation, we define $V(x, y, z) = 1$, if (x, y, z) is inside a 3D object; else $V(x, y, z) = 0$. Then, we execute the following five steps to find the rotation matrix for two 3D objects.

1. Compute the centroid of $V(x, y, z)$.
2. The eigenvectors of the inertia matrix $I_{3\times3}$ is the principal axes of the object $V(x, y, z)$. To compute each element of I, see [12].
3. Normalize eigenvector matrix of I and denote E, and e_{ij} denotes E's element at (i, j).
4. Rotate $V(x, y, z)$ with respect to the x, y, z axes, respectively.

```
3D_ Erosion_Based_Interpolation( )
begin
  /* a. erosion queue L_E initialization */
  for each voxel (x,y,z) of ∂X_0
    do  insert a new node M(x,y,z,n+1,0,0)
       into the erosion list L_E
  for each voxel (x,y,z) of ∂X_n+1
    do  insert a new node M(x,y,z,0,n+1,0)
       into the erosion list L_E
  /* b. to interpolate X_k object,we calculate the exact number of erosions,
     denoted as D_erosion and update all nodes in the queue and corresponding B(x,y,z) */
  for each node M in the list L_E
     do D_erosion ← B(x,y,z) * E_k^n    if b = 0   or
        D_erosion ←B(x,y,z) * E_k^n*  if b = n+1
        update the value d of M by D_erosion
        update B(x,y,z) by D_erosion

  /* c. erosion process */
  while L_E ≠ ∅
    do delete the first node of the list L_E and denoted as N(x,y,z,a,b,d)
     /* d. if required to be eroded */
     if ( N.d > 0 and N.d ≥ B(N.x,N.y,N.z)
      then for each six-neighbor P(x',y',z') of (N.x,N.y,N.z)
       /* check if N's 6-neighbors could be eroded or not */
       /* In the following, B(P) denotes B(x',y',z') and distance stored on
       A_a(x',y',z') is denoted as A_a(P)

            /* e. normal erosion case */
            do if A_a(N.x,N.y,N.z) = A_a(P)+1
                 then if N.d-1 > B(P)
                        then
                             update B(P) by N.d-1
                             insert point P into the tail of the list
                             L_E with (x',y',z' a, b, N.d-1 )
            /* f. special erosion case */
               if A_a(N.x,N.y,N.z) = A_a(P)
                 then if all the six-neighbor Q of P, A_a(P) ≥ A_a(Q)
                        then
                             update B(P) by N.d
                             insert point P into the tail of the list
                             L_E with (x',y',z',a,b,N.d )
  /* g. get result */
  Finally, the newly interpolated object's boundary consists
  of all voxels (x,y,z) with B(x,y,z) = 0.
End
```

Fig. 4. Erosion-based Interpolation.

5. We execute above steps for two input objects and then compute a rotation matrix $R = E_1 E_2^{-1}$, where E_1 and E_2 are normalized eigenvector matrixes of two objects.

In the 5^{th} step, the rotation R matrix rotates an object$_2$ from its original orientations to the orientations of another object$_1$. To perform 3D morphing, first we will use R to rotate object$_2$ to align with object$_1$. Then, we employed the proposed 3D morphing method to create the interpolated object$_t$. To achieve final morphing, we will further rotate object$_t$ by:

$$I * (1 - t) + R^{-1} * (t). \tag{4}$$

Where I is an identity matrix and R^{-1} is the inverse of matrix R. Here, object_t will be object_1 as t=0 and it will be object_2 as t=1. Then, we translate object_t to corrected interpolated centroid using equation (3). In contrast to Cohen-Or's [8], our adoption does not require the tedious feature specification. However, in the next section the experimental results show that our scheme with PAR yields nice shape transition.

3 Preliminary Results

To experimentally evaluate the proposed schemes, we execute five 3D morphing examples. In these examples, each 3-D object is voxelized as 192x192x192 binary volume data. Figure 5 shows an interesting morphing effect. One larger object, say L is torn into four smaller pieces, say s_1, s_2, s_3 ,and s_4. Using the proposed method, this effect can be easily achieved as follows: interpolate each (L , s_i) pair first and then combine these four interpolated results using set union operation to accomplish morphing. In contrast, for the feature-based schemes, it is not an easy job to specify appropriate features to achieve this tearing effect [10]. The second example is shown in Figure 6. In this example, a CT head will gradually be turned into a digit "5". We will point out that both Figures 5 and 6 can be also handled well using two 2-D version of the proposed scheme, too. Since in these two examples, both source and target objects have equal heights in the Z-axis, we can interpolate each contour pair slice by slice and then stack interpolated results to accomplish 3-D morphing. To the contrary, we must employed 3-D version to Figure 7 and Figure 8 instead, since their heights are different in the Z-axis. Both examples also yield very nice interpolations. Additionally, we include PAR option in the experiment of Figure 8. In this experiment, the cat and dock are originally in different orientations according to principal axes found by PAR. First, we find their rotation matrix R and then we will use equation (4) to gradually rotate the interpolated object. In this manner, in the course of interpolation, we can see the orientation of the cat is gradually rotated to that of dock. To demonstrate the importance of PAR, we conduct another experiments as illustrated in Figure 9 (a) and (b). In figure (a), since we do not apply PAR to volume data (Cat & Venus) first, the morphing sequence seems very awkward and unnatural. In contrast, if we perform PAR a priori, we can obtain better and smoother morphing sequence in figure (b). There are more results and animation sequences found in our research web:

`http://couger.csie.ncku.edu.tw/~vr/3dmorph.html`

4 Conclusion and Future Work

In this paper, we propose a morphology-based volume metamorphosis scheme. We successfully apply this scheme on several examples and illustrate several preliminary results. The proposed method does not require tedious feature specifications, but it generates satisfactory object transitions in our experiments. Finally, we should admit that not all morphing effects could be achieved by

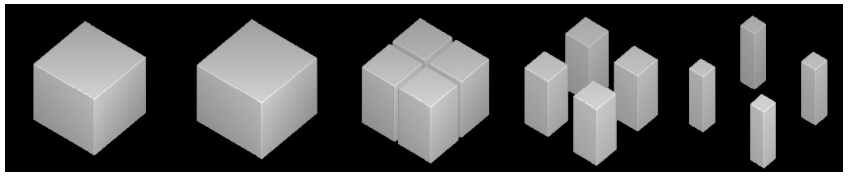


Fig. 5. One cube is torn into four small pieces.

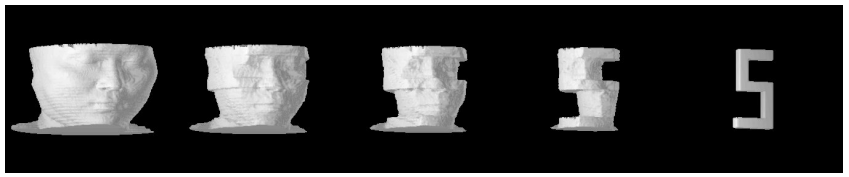


Fig. 6. CT head is transformed into a digital number "5".

the proposed scheme. For example, for two given models: baby head and adult head, we cannot guarantee the noses of both the baby and adult will be the exact correspondence during morphing. This is a common drawback of the non-feature-based morphing methods such as our method. In future, we plan to include the concept of feature control in our approach but we only require few features. In this manner, it will potentially lead to correct morphing results as well as to alleviate the time-consuming manual efforts.

Acknowledgement

This project is supported by National Science Council, Republic of China, Taiwan, under contract NSC-91-2213-E-006-078.

References

1. Lazarus, F., and Verroust, A. "Three-dimensional Morphing: a survey," The Visual Computer, Vol. 14, 1998, pp. 373-389.
2. Levin, David, "Multidimensional Reconstruction by Set-valued Approximation," IMA J. Numerical Analysis, Vol. 6, 1986, pp. 173-184.
3. Rossignac, Jarek and Anil Kaul, "AGERLs and BiPs: Morphing as a Bezier Curve in the Space of Polyhedra, " in Proceedings of Eurographics'94, Sept. 12-16, 1994, pp. 179-184.
4. Sai Prasad Raya, Jayaram K. Udupa, "Shape-based Interpolation of Multidimensional Objects," IEEE Trans. on Medical Imaging, Vol.9, No.1, 1990, pp. 32-42.
5. Gabor T. Herman, Jingsheng Zheng, Carolyn A. Bucholtz, "Shaped-based Interpolation," IEEE Computer Graphics & Applications, May 1992, pp. 69-79.
6. Tong-Yee Lee, Wen-Hsiu Wang, "Morphology-based Three-dimensional Interpolation," IEEE Transactions on Medical Imaging, July 2000, Vol.19, pp. 711-721.

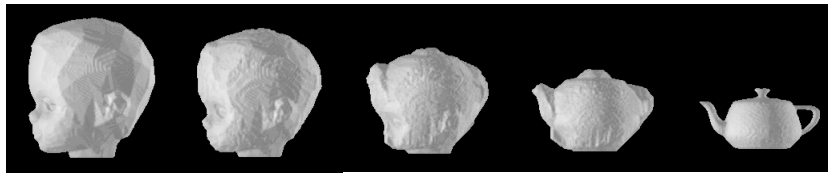
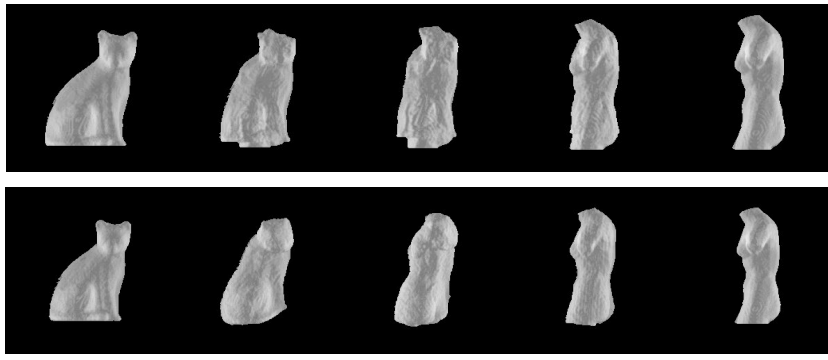


Fig. 7. A baby head is transformed into a teapot.



Fig. 8. A cat is transformed into a duck.



(a)

(b)

Fig. 9. Cat vs. Venus morphing sequence; (a) without PAR and (b) with PAR.

7. Payne, Bradley A. and Arthur W. Toga, "Distance Field Manipulation of Surface Models," IEEE Computer Graphics & Applications, Vol. 12, No. 1, 1992, pp. 65-71.
8. Cohen-Or, Daniel, David Levin and Amira Solomovici, "Three-dimensional Distance Field Morphing, " ACM Transactions on Graphics, Vol. 17, No. 2, 1997.
9. A. Lerios, C. Garfinkle and M. Levoy, "Feature-based Volume Metamorphosis," in Proceedings of ACM SIGGRAPH'95, 1995, pp. 449-456.
10. Tong-Yee Lee, Y-C. Lin, Leeween Lin, Y.N. Sun, "Fast Feature-Based Metamorphosis and Operator Design," Eurographics'98, International Journal of the Eurographics Association, Computer Graphics Forum, Vol.17, No.3, 1998, pp. C-15-30.
11. Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison-Wesley Publishing Company, Inc., 1992.
12. Louis K. Arata, Atam P. Dhawan, Joseph P. Broderick, Mary F. Gaskil-Shipley, Alejandro V. Levy, and Nora D. Volkow, "Three-dimensional Anatomical Model-based Segmentation of MR Brain Images Through Principal Axes Registration," IEEE Transactions on Biomedical Engineering, Vol.42, No.11, 1995, pp. 1069-1078.

Self-correctional 3D Shape Reconstruction from a Single Freehand Line Drawing

BeomSoo Oh[1] and ChangHun Kim

Dept. of Computer Science and Engineering, Korea University,
1, 5-ka, Anam-dong, Sungbuk-ku, Seoul 136-701, Korea
{obs, chkim}@korea.ac.kr

Abstract. The goal of sketch reconstruction is to take an inaccurate, 2D edge-vertex graph (i.e., sketch drawing) as input and reconstruct a 3D shape as output. However, traditional reconstruction methods based on image regularities tend to produce a distorted 3D shape. In part, this distortion is due to the inherent inaccuracies in the sketch, but it also relates to the failure to accurately distinguish between important and less important regularities.
We propose a new self-correctional reconstruction algorithm that can progressively produce refined versions of sketch reconstructions. The algorithm corrects the shape and the drawing simultaneously using geometric error metrics. The proposed algorithm can minimize the distortion of the shape by adding 3D regularities to the image regularities. The self-correctional algorithm for minimizing the distortion of sketch reconstruction is discussed, and the experimental results show that the proposed method efficiently reconstructs more accurate 3D objects than previous ones.

1 Introduction

During the conceptual design stage of products, designers tend to draw their basic ideas mainly on papers with pencil. The method of representing three-dimensional information by using a line drawing is easy to input geometrical information. Once the 3D shape is obtained, it can be manipulated/modified, and further detail can be sketched in to obtain more detailed and accurate object. The approach provides designers with the means to convey their ideas to a CAD system.

Figure 1 shows the concept of the self-correctional 3D reconstruction system. The input to the system is an inaccurate sketch drawing (Figure 1a). Previous methods reconstructed distorted 3D object directly from the inaccurate freehand sketch drawing as shown in Figure 1b. Proposed method refines the sketch drawing and 3D object simultaneously by minimizing the energy function derived from the geometric constraints. As we can see in Figure 1c,d, the proposed method facilitates much better corrections.

In earlier work [6], Lipson et al. proposed an approach to optimization-based reconstructing a 3D object from the sketch drawing using image regularities. In

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 528–538, 2003.
© Springer-Verlag Berlin Heidelberg 2003

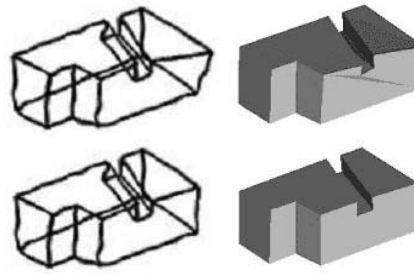


Fig. 1. Self-correctional sketch reconstruction. (a) Input drawing, (b) Distorted reconstruction (c) Refined drawing, (d) Corrected reconstruction

their research, the reconstruction process is based on identifying and formulating geometrical regularities. The algorithm is capable of reconstructing objects with considerable fault tolerance.

However, the reconstruction results appear to correctly reflect the concept of the depicted object but tend to produce a somewhat distorted 3D shape [6]. In part, this distortion is due to the inherent inaccuracies in the sketch, but it also relates to the failure to accurately distinguish between important and less important regularities. In addition, the reconstruction process is far more prone to errors when the object involves curved faces. This can be attributed to the fact that the majority of regularities used are applicable only to straight-line segments.

To overcome these difficulties, we have developed an approach based on progressive self-correctional reconstruction. This has allowed us to robustly estimate the relative positions of vertices of an object. By examining the projective geometric constraints between the 3D object space and the 2D sketch plane, we first propose the objective function to reconstruct the most plausible object. Next, we determine whether the sketch drawing is inaccurate or not. Finally, we minimize the distortion of the reconstruction results by optimizing the objective function. Main features of the algorithm are as follows.

- Rough 3D reconstruction: The algorithm is required to arrive at a rough 3D reconstruction. The reconstruction shown in Figure 1b,d reflects the correct notion of the object, though not strictly accurately.
- Self-Correctional Reconstruction: The procedure for identifying the most probable solution must be able to tolerate the inaccurate sketches. However, sometimes the sketch is too inaccurate to reconstruct plausible object. The algorithm is required to reconstruct a 3D object roughly, and it can adjust a sketch if there are obvious faults in sketch as shown in Figure 1.

2 Background and Related Work

A sketch drawing lacks an entire dimension. Therefore, additional methods must be used to extract the missing dimension. Wang et al. [11] and Lipson [5] surveyed detailed works on 3D object reconstruction from a single projection.

There are two stages in reconstructing the 3D geometrical information from a 2D sketch drawing. At the face identification stage, the edge-vertex graph is generated by analyzing strokes in the sketch drawing. Then, we should identify 2D actual faces of the target object [8,9]. At the object reconstruction stage, the 3D object is restored by means of the geometrical/topological relationship among 2D identified faces, edges and vertices [6].

This paper discusses mainly about the restoration of 3D object. Marill [7] suggested an optimization-based reconstruction by means of the minimum standard deviation of angles (MSDA). His method inflates the flat sketch into three-dimensional objects by minimizing the objective function derived from MSDA at junctions. However, this technique yields implausible reconstruction for non-orthogonal shapes in experiments. Braid and Wang [1] refined Marill's approach by improving the optimization procedure with conjugate gradients. Leclerc and Fiscler [4] identified all the non-self-intersecting closed circuits of edges. Their method, however, cannot be applied to the concave faces and to the ambiguous sketch drawings. In addition, although they amended Marill's method by using face planarity, their method limited the type of objects. Lamb [3] implemented interactive reconstruction system based on the line labeling. The algorithm enables users to avoid ambiguity, and it identifies principal axis and symmetry of face.

Shpitalni and Lipson [9] efficiently identified actual faces by using the maximum rank equation and the face adjacency theory. However, their method requires the large search domain of the minimal faces. Lipson and Shpitalni [6] reconstructed the 3D object with flat and cylindrical faces by formalizing various image regularities. Their technique yielded the plausible reconstruction for various objects including manifold or non-manifold.

Despite many methods proposed, it is still difficult to develop a practical reconstruction system for the following reasons: (i) They use only the 2D image regularities. (ii) In addition, there are no method to refine the distortion of the reconstructed results and the sketch drawings.

3 Self-correctional 3D Shape Reconstruction

3.1 Problem Statements

- Goal : The goal of the sketch reconstruction is to restore the original 3D object using the information derived from the projection only.
- Input drawing : A sketch drawing is a single 2D projection of a 3D object that is manifold or non-manifold. The input projection represents the wireframe model of a general object, and it consists of only a single 2D line drawing, which is a graph of connected entities.

– Output reconstruction : The 3D objects can be manifold or non-manifold objects depicted in the sketch drawing. However, any information about the 3D object itself as well as its type or its position to the viewpoint is not provided for the reconstruction system. The projection is drawn from the general viewpoint that reveals all edges and vertices.
– Processing requirement : A 2D sketch might be a projection of an infinite number of possible 3D objects. The reconstruction must produce the most plausible 3D object described with a given projection, that human observers are most likely to select. In addition, the procedure to identify the most probable solution must be able to tolerate faults due to an inaccurate sketch drawing.

3.2 Objective Energy Function

The goal of sketch reconstruction is to obtain a 3D structure of an object depicted in the sketch drawing. We find a topological information K, a set of vertex positions V, and the sketch drawing S defining a 3D object $M = (K, V, S)$ that minimizes the energy function

$$E(K, V, S) = \\ E_{vertex}(V, S) + E_{edge}(K, V, S) + E_{face}(K, V) \tag{1}$$

We assume that topological information K can be obtained by identifying 2D actual faces of the 3D object [9,8]. To estimate the plausible positions for the vertices, we derive three constraints from the relationship between an object space and the projection plane. In the reconstruction process, each vertex should satisfy three constraints. Three constraints are as the followings:

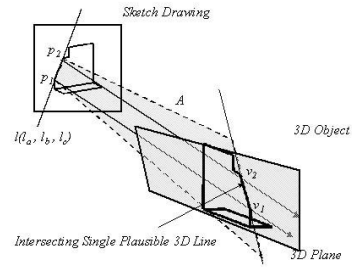


Fig. 2. Geometric constraints between the 3D object space and the 2D sketch plane

– $E_{vertex}(V, S)$: The vertex constraint means that each vertex v of a 3D object should be transformed into the corresponding 2D point p of the sketch drawing by the projection matrix T. That is, each vertex v satisfies $Tv - p = 0$.

- $E_{edge}(K, V, S)$: Given a 2D line l ($\overline{p_1 p_2}$), we can generate a 3D plane A. Then, the corresponding end-vertices of a 3D line L ($\overline{v_1 v_2}$) should lie on a plane A. The edge constraint can be defined by summing the distance from each vertex to the plane. However, there are infinite plausible lines that satisfy the vertex constraint and the edge constraint.
- $E_{face}(K, V)$: The face constraint means that 3D vertex v should lie on the 3D plane (face). With the constraints on vertex, edge and face, we can narrow down the optimal 3D line among the infinite plausible lines as shown in Figure 2.

To estimate the progressive positions for the vertices, we generate $(i+1)^{th}$ partial refined object(M_{i+1}) by optimizing the energy function derived from the previously reconstructed partial refined object(M_i) , the projection matrix(T_i), and current refined sketch drawing(S_{i+1}). The energy function must be optimized for the coordinates of vertices and points.

A vector V containing the x, y, and z coordinates of the vertices can represent a 3D configuration of an object. An energy function $E(K, V, S)$ can be computed for any 3D configuration by evaluating three constraints above. The manipulation process of the V for the best reconstruction is the matter of a full 3n-dimensional nonlinear optimization, where n is the number of vertices of current partial object. We use Brent minimization algorithm [2]. Since the minimization actually needs much time, we reconstruct a 3D structure of an object face by face each time.

Although we optimize the nonlinear function by dividing it into smaller units, the initial estimation of solution is the main issue in the nonlinear optimization. We use ADG as an initial estimation to reconstruct an initial object. Consequently, we use the $(i-1)^{th}$ partial solution as an initial estimation of the i^{th} solution. This initial estimation is useful to restore a 3D structure of an object efficiently.

3.3 Distortion Minimization

By examining the relationship between 3D object and sketch drawing, we propose the objective energy function derived from three constraints to reconstruct the most plausible object. Nevertheless, the reconstruction results tend to produce a somewhat distorted 3D object due to the inherent inaccuracies in sketch drawing. Because sketch drawing is itself inaccurate generally, we cannot determine whether the optimization will succeed or fail. Only we can do is that we should start optimization process from good initial guess so as not to get into local minimum.

However, sometimes the sketch is too inaccurate to reconstruct plausible object. The algorithm is required to reconstruct a 3D object roughly, and it can adjust a sketch if there are obvious faults in sketch. In experiments, we discover that although we cannot determine the success of reconstruction, we can determine the sketch is inaccurate or not after reconstruction. This capability is used to refine sketch more accurately.

Table 1. The comparison of the input drawing and the refinement of the synthetic sketch drawing ($E(K, V, S)$)

Figure 3	(a)	(b)	(c)
Input drawing	4.84377	21.46840	65.30986
Refined drawing	1.19213	4.73310	12.48470

Assume that we have synthetic object, and its various projections. We evaluate the value of the objective energy function for synthetic object and reconstructed objects with same projections. The reconstructed objects are very different according to the projections (sketch drawings); however, its 3D configuration is acceptable for us to perceive from the projections. We use the evaluated value of the objective function as the estimation of the accuracy of sketch drawing. If the value is above threshold, we determine that the sketch drawing is inaccurate.

To determine whether the sketch drawing is obvious accurate or not, we introduce threshold ϵ. If the evaluation of the objective function is more than as Equation 2, we determine that the sketch drawing is inaccurate seriously. However, it is difficult to determine proper value of threshold ϵ, currently we assume that $\epsilon = 3.0$. This threshold is not absolute value, however, we get stable self-correction with that value experimentally.

$$\epsilon > E(K, V, S) \tag{2}$$

Currently, it is difficult to which point in the sketch drawing is inaccurate more than other points. We assume that user specify the point that is obvious inaccurate. If the inaccurate point is specified, it is simple task to refine that point more accurately. We minimize the objective function with respect to the vertices of a 3D object and an inaccurate point. This means that the specified point can effect on the projection matrix T, therefore, the objective function.

Figure 3 shows various inaccurate sketch drawing generated from synthetic object, we add some noise to the sketch drawing in one point. In the process of sketching, it is common to draw inaccurate point as left column in Figure 3. However, the reconstruction result is very inaccurate. Figure 3 and Table 1 show that our self-correction process is well suited for small noise. If the noise is large than about 30 pixels, the correction of sketch is still inaccurate. By applying our self-correction approach, however, we get more accurate sketch drawing.

As we mentioned, the capability of sketch refinement comes from the constraints used in the objective energy function. We introduced threshold ϵ to determine whether the sketch drawing is obvious inaccurate or not. In most freehand sketches, the evaluated value of the objective function is large than ϵ, that is, the sketch drawings are inaccurate. It is actual nature of the sketch that makes it a fast communication language for conveying conceptual ideas.

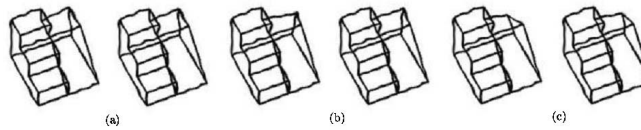


Fig. 3. Self-correction of the noisy synthetic sketch drawing. (a) 14 pixels noise ,(b) 28 pixels noise, (c) 42 pixels noise

3.4 3D Regularities

We introduce some geometrical constraints of 3D regularities that are used into the objective energy function to reconstruct objects more accurately.

$$E(K,V,S) = E_{vertex}(V,S) + E_{edge}(K,V,S) + E_{face}(K,V) + E_{3Dregularities}(K,V) \tag{3}$$

$$E_{3Dregularities}(K,V) = \alpha_{face_parallelism} + \alpha_{face_orthogonality} + \alpha_{radius_equality} \tag{4}$$

– Face parallelism / orthogonality: A parallel / orthogonal pair of planes in the sketch plane reflects parallelism / orthogonality in space. The term used to evaluate is

$$\alpha_{face_parallelism} = \sum_{i=1}^{n} \{cos^{-1}(n_1 \cdot n_2)\}^2 \tag{5}$$

$$\alpha_{face_orthogonality} = \sum_{i=1}^{n} \{sin^{-1}(n_1 \cdot n_2)\}^2 \tag{6}$$

where, n_1 and n_2 denote all possible pairs of normal of parallel / orthogonal faces.

– Radius equality: All radiuses of quadric surfaces should be equaled each other.

$$\alpha_{radius_equality} = \sum_{i=1}^{n} \{sin^{-1}(d_1 \cdot d_2)\}^2 \tag{7}$$

where, d_1 and d_2 are distance from center of curve to the end-vertices.

We can find parallel or orthogonal faces by using ADG (angular distribution graph) that identifies prevailing axis system. First, we find each edges prevailing axis. Then, all faces contain at most two prevailing axes. If two faces containing two axes have the same axes, and then they are parallel faces, else they are orthogonal faces. In addition, we assign high weight to the regularity of face planarity to reconstruct the most plausible solution with reality.

Table 2. Evaluation of the refinement of the freehand sketch drawing ($E(K, V, S)$)

Figure 4	(a)	(b)	(c)
Input drawing	22.60258	16.90177	39.10252
Refined drawing	4.26594	2.74861	2.97916

Table 3. Evaluation of the sketch refinement ($E(K, V, S)$)

Figure 5	(a)	(b)	(c)
Input drawing	31.61546	18.75655	13.31084
Refined drawing	1.56168	2.78907	4.40385

4 Experimental Results

4.1 Results

To estimate the efficiency of the proposed algorithm, we applied the method to various sketch drawings on a PC with Pentium III processor (600 MHz).

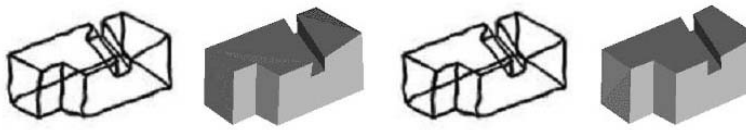
We use the synthetic sketch with some noise. We use images with 512×512, therefore, the threshold we assumed ($\epsilon = 3.0$) is quite small quantity.

If user specify the point that is obviously inaccurate, we can refine that point as shown in Figure 3. Here, we try to refine the sketch drawing without manual intervention. Of course, we assume that we reconstructed the most plausible object by applying the progressive reconstruction algorithm already. The process of minimization is a full 5n-dimensional nonlinear optimization problem, where n is the number of points or vertices. To reduce computation, we refine sketch drawings point by point.

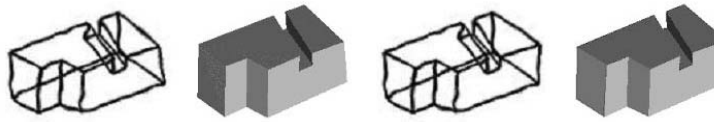
Figure 4 shows that our self-correction process copes with freehand sketch. Table 2 indicates that if the sketch drawing has much noise, the value of the objective function is more severe. From this point of view, the sketch of Figure 4a is more accurate than the sketch of Figure 4b. Because freehand sketch it self is inaccurate, thhis situation may occur in freehand sketch although we add more noise to the sketch of Figure 4b.

4.2 Discussions

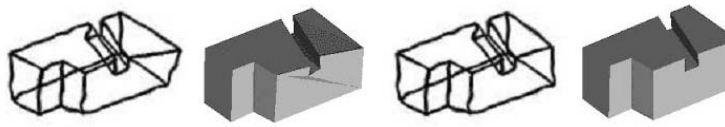
Figure 5 shows the results of sketch refinement, however, it is difficult for human beings to differentiate the original sketches from the refined sketches at a glance. We refine the sketch by using our progressive reconstruction algorithm. We evaluate the reconstruction results of original / refined sketch drawings by comparing the objective energy function. Table 3 shows that the refined sketches are somewhat accurate. In experiments, we find that our self-correction process is well coped with obvious faults as depicted in 6.



(a) 14 pixels noise & self-correction



(b) 28 pixels noise & self-correction



(c) 42 pixels noise & self-correction

Fig. 4. Self-correction of freehand sketch with noise

To evaluate the effect of 3D regularities, we check the 3D error and the 2D error. We define 3D error as the distance between the depth of reconstructed object's vertices and the real depth of synthetic object's vertices, and we define the 2D error as the sum of regularities proposed by Lipson et al. [6]. When 3D regularities and quadric face regularities are used to improve the shape (after 20 iteration), they can perturb the error curve as demonstrated by the sudden spike (Figure 6). However, they improve significantly the shape of an object with more iteration. Figure 6 shows that the constraints of 3D regularities improve the shape of reconstructed object significantly. When 3D regularities and quadric faces regularities are used in curved object, they reduce 2D error as well as 3D error significantly in case of the quadric object. However, the error in curved objects still more significant than that in polyhedral objects because most of regularities are derived from 2D planar configuration.

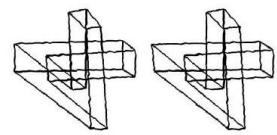


Fig. 5. Results of the sketch refinement

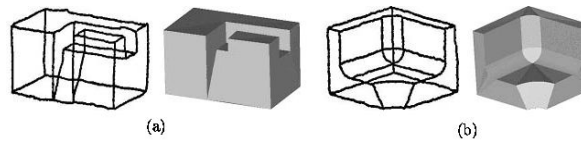


Fig. 6. Results of 3D Regularities. (a) Face regularities (polygonal object) (b) Face regularities & radius regularities (quadric object)

5 Conclusions

We developed a self-correctional reconstruction algorithm that can progressively produce refined versions of sketch reconstructions. By examining the projective geometric constraints between the 3D object space and the 2D sketch plane, the algorithm uses iterative refinements to correct the shape and the drawing simultaneously using geometric error metrics. This can facilitate much better corrections.

Future works are as follows. First, we are interested in refining sketch drawing more quickly without manual intervention. Second, we will use quadric models to make our approach more general. Finally, as an application of the sketch reconstruction, we will reconstruct the object and the scene from a real photograph or a set of photographs. By using the progressive sketch reconstruction algorithm, we will generate photo-realistic models with texture, and estimate projection matrix.

Acknowledgments. This research was supported by University IT Research Center Project

References

1. L. Braid and P.Wang. Three-dimensional object recognition using gradient descent and the universal three-dimensional array grammar. In SPIE, volume 1607, pages 711.718, 1991.
2. R. P. Brent. Algorithms for Minimization without Derivatives. Prentice Hall, Englewood Cliffs N. J., 1973.
3. D. Lamb and A. Bandopadhay. Interpreting a 3d object from a rough 2d line drawing. In Proceeding of Visualization 90, pages 59.66, 1990.
4. Y. G. Leclerc and M. A. Fiscler. An optimization based approach to the interpretation of single line drawings as 3d wire frames. Int. J. of Computer Vision, 9(2):113.136, 1992.
5. H. Lipson. Computer Aided 3D Sketching for Conceptual Design, PhD Thesis. Israel Institute of Technology, Israel, 1998.
6. H. Lipson and M. Shpitalni. Optimization based reconstruction of a 3d object from a single freehand line drawing. Computer Aided Design, 28(8):651.663, 1996.

7. T. Marill. Emulating the human interpretation of line drawings as three-dimensional objects. Int. J. of Computer Vision, 6(2):147.161, 1991.
8. B. S. Oh and C. H. Kim. Fast reconstruction of 3d objects from single free-hand line drawing. LNCS, 2059:706.715, May 2001.
9. M. Shpitalni and H. Lipson. Identification of faces in a 2d line drawing projection of a wireframe object. IEEE Trans. Pattern Analysis & Machine Intell., 18(10):1000. 1012, 1996.
10. M. Shpitalni and H. Lipson. Classification of sketch strokes and corner detection using conic sections and adaptive clustering. Trans. of the ASME. J. of Mechanical Design, 119(2), 1997.
11. W. Wang and G. Grinstein. A survey of 3d solid reconstruction from 2d projection line drawings. In Computer Graphics Forum, pages 137.158, 1993.

3D Primitive Reconstruction Using the Line Segment with Single Image

Sung-Dong Cho, Seung-Taek Ryoo, Tai-Ho Choi, and Kyung-Hyun Yoon

Department of Image Engineering Graduate School of
Advanced Imaging Science, Multimedia and Film
ChungAng University, Seoul, Korea
{sodeng, bluelancer, thchoi, khyoon}@cglab.cse.cau.ac.kr

Abstract. This paper presents a 3-D reconstruction method IBM(image based modeling) of an image that does not contain any camera information. This system adopts a 3D reconstruction method based on a model. Model-based 3D reconstruction recovers an image using the geometric characteristics of a pre-defined polyhedron model. It uses a pre-defined polyhedron model as the primitive and the 3D reconstruction is processed by mapping the correspondence point of the primitive model onto the picture image. Existing model-based 3D reconstruction methods were used for the reconstruction of camera parameters or error method through iteration. However, we proposed a method for a primitive model that uses the segment and the center of the segment for the reconstruction process. This method enables the reconstruction of the primitive model to be processed using the minimum camera parameters (e.g. focal length) during the segment reconstruction process.

1 Introduction

Image-based modeling is a method of auto-recovering 3D information from a single or a number of images to enable the user to perform 3D modeling. Image-based modeling requires the intrinsic and extrinsic parameters to be present for the 3D recovery process. The algorithm for the image-based modeling method using through the camera parameter is prone to errors caused by the relationship between the input images or the characteristics of the image. This problem includes the problem of camera calibration, obtaining the geometric relationship of the camera between two or more images, and the problem of the correspondence point to apply these geometric relationship.

The camera parameter information obtained through the calibration process, is used to determine the location and direction of the object as well as for the recovery of the correspondence point. Therefore, the calibration of the obtained image is a very important process in the 3D recovery process. The camera calibration process is done by calculating the camera parameter value of the correspondence point or by using the relationship of the geometric information transformed on the projective space [4].

If the vanishing points information of 3 parallel points are available on an image, the camera calibration can be processed [5]. Vanishing point is an information that can easily be acquired from the image and is easy to manipulate for more accuracy in the

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 539-548, 2003.
© Springer-Verlag Berlin Heidelberg 2003

camera calibration process. However, it also becomes a liability in the reconstruction process since only planes vanishing point on it can be reconstructed. Pollefeys[1] suggested a method for auto-calibration through automatically extracting corresponding points from a sequence of densely acquired images. This method requires a dense image sequence but does not require any user input and performs a more accurate calibration. However, because the camera parameter values are not fixed the parameter values must be updated for each image. Also, because the corresponding points are extracted using the relationship between the images, no geometric relationship exists between the images. This means that there is no restriction on the type of model to be recovered, but also that the characteristics of the model to be recovered cannot be expressed accurately. Debevec [3][6] used a different method of a user-input based geometric approach. This system combines the image-based method and the geometric based method for modeling the virtual environment. This method adopts a method of getting the line and point of the pre-defined model as user input. Because this method uses a pre-defined model, the recovered result can be used as 3D model information.

We propose a method for 3-dimensionally reconstructing an object using a line that includes the midpoint information from a single image. A pre-defined polygon is used as the primitive and the recovery is processed from a single image.

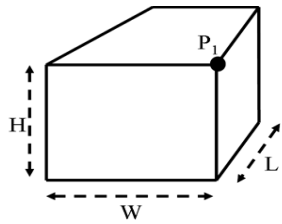
Existing methods using a single image used the vanishing points or error function for the iteration. However, this method was inefficient in that it did not fully utilize the polygon. A pair of parallel line must exist for the vanishing point and the vanishing point must cross at right angles for the vanishing point information to be valid. A more accurate result can be obtained using the error minimization method through iteration, but the characteristics of the object is not considered in the definition stage. The system used in this study approaches the primitive through the lines consisting of the points instead of the traditional vanishing point and the error minimization method, to maintain the characteristics of the primitive.

2 Definition and Expression of the Primitive

The recovery is processed by mapping a pre-defined primitive on the image. The geometric characteristics of the primitive are used for the recovery process and the recovered 3D model keeps its primitive characteristics. The predefined primitive should be easy to use. In this section, the method used for defining the primitive and the expression of the parameters and the recovered line.

The primitive is defined as the local coordinate LC(X,Y,Z) and dimension vector λ that has the three vectors X,Y, and Z as the axis. The local coordinate LC becomes the coordinate that defines the primitive and each point on the primitive can be expressed as a one dimensional function as shown in equation (1). For example, considering a primitive of the polygon, by defining the three normal of each surface of the polygon as X, Y and Z axis, the 8 points on the polygon can be expressed as a one-dimensional matrix for the dimension vector λ , as shown in [Figure 1].

$$\lambda = (W, H, L),\ M_i \text{is } 3 \times 3\ m\text{atrix},$$
$$P_i = M_i \times \lambda \quad \text{..... (1)}$$



$$P_1 = \begin{pmatrix} W \\ H \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \lambda$$

[Figure 1] Representation of points

The recovery process used in this paper, deals with the problem of solving the local coordinates X, Y, and Z and the dimension vector λ. To recover these parameter values, there is a restriction on defining the primitive. First of all, a line containing its center point is used as the information for recovering each parameter. Therefore, the line and its center point must be extracted from its primitive. Secondly, more than two lines and mid-points are required for the recovery of the local coordinate of the primitive, and more than the number of dimension vector lines are required for the recovery of the dimension vector λ.

The minimum number of lines for the 3D recovery, N, can be acquired using the following equation.

N=max(λ number of elements, 2)

3 Recovery of the Line Using the Midpoint Information

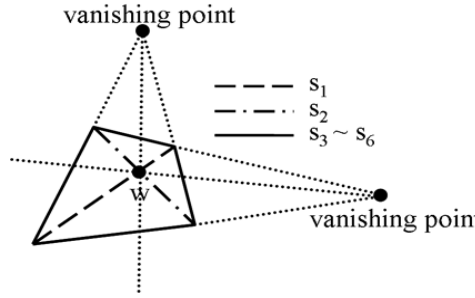
In this section, we will take a look at the problem of 3-dimensionally recovering a line from the midpoint of the line and introduce two different methods used for extracting the midpoint information. In the other studies on line recovery, they used the correspondences between the images or the camera calibration method to acquire the 3D vector or surface information. However, the method that we proposed in this paper only requires the focus length of the camera as a parameter, and uses the midpoint of the line for the 3D recovery process. This reduces the process of camera calibration and allows the user to recover the image using a single image.

3.1 Midpoint Enformation Extraction from a Primitive

Extracting the line and the midpoint information from a primitive can be defined differently by the characteristics of the primitive. In other words, it is hard to find a general method but can be applied on any primitive from which the midpoint can be extracted. In this section, we will take a look at the rectangle used for constructing a 3D model.

As shown in [Figure 2], a line s_i ($i \in [1,6]$) with 6 midpoints can be extracted from a rectangle. The crossing point of the two diagonal lines of a rectangle divide the lines

(s_1,s_2) into half and the crossing point becomes the center of gravity of the rectangle. Therefore, the two diagonal lines are determined by the four points of the rectangle and the crossing point becomes the midpoint of the two lines. This straight line divides the lines that have not been used for the creation of vanishing point. If they are parallel on the image, the vanishing point is not created and makes the lines that have not been used also parallel. The midpoint information of the inner lines of the rectangle can be acquired through these two methods. This midpoint information is used for the 3D recovery process discussed in the following section and the result image of curved model is shown in [Figure 10]. This model is the result acquired by defining and 3-dimenstionally reconstructing the cylinder. The two diameters used as the input for the bottom of the cylinder, intersect the circle and divides the diameters into half. Therefore, these two lines can be 3-dimensionally recovered, and the local coordinate LC and size vector λ can be calculated.



[Figure 2] Line segments with center points

3.2 3D Reconstruction of Line Segment

In this paper, line recovery uses the two points creating a line and the line midpoint. [Figure 3] shows the relation between the extracted line(p1,p2), midpoint of the line(c) and the line recovered line(Q1,Q2) in space. By using p_1, p_2, c on the image plane the direction vector $\vec{L}$ of the line Q_1 ,Q_2 can be recovered in space.

The point (p1,p2,c) projected onto the surface and the point on space (p1,p2,c) and (Q1,Q2,C) have the same meaning on the projected space and exist on the same surface. This is because p1,p2, and c are values acquired through the projection of points Q1,Q2, and C that are on the line. The line on the image (p1,p2) and midpoint (c) are used for the recovery process explained below..

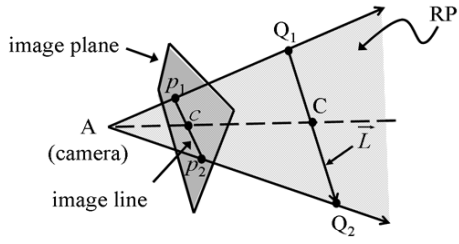
The plane created by A(camera), p1, p2 can be seen as plane RP. The recovered lines Q1, Q2 exist on plane RP and is bisected by the line connecting camera A and midpoint C. All lines Q1, Q2 existing on plane RP all have the same direction $\vec{L}$. we can calculate the direction vector $\vec{L}$ of lines Q1, Q2 in 2-D plane RP. As shown in [Figure 4] the line passing through the camera and point c becomes the v axis and to make u,v plane exist on plane RP, a u,v,n coordinate system is configured using midpoint c as the origin. Equation(2) shows the u,v,n coordinate system transformation matrix F..

$$F = [R \mid T], \quad R = \begin{pmatrix} u \\ v \\ n \end{pmatrix}, \quad T = \begin{pmatrix} -c_x \\ -c_y \\ -c_z \end{pmatrix} \quad \cdots\cdots (2)$$

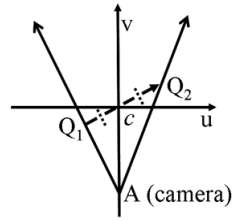
The points transformed to u,v,n coordinate system are p'1,p'2, A', shown in Equation(3), and the line created from A' to p'1,p'2 is $\overline{p'_1, p'_2}$. Points Q1, Q2 exist on line $\overline{p'_1, p'_2}$ and the direction vector $\vec{L}$ can be calculated by using the slope $\nabla\overline{p'_1}, \nabla\overline{p'_2}$ of $\overline{p'_1, p'_2}$ shown in Equation(4)

$$\begin{aligned} p'_1 &= F\ p_1, \quad p'_2 = F\ p_2, \quad A' = FA \\ \overrightarrow{p'_1} &= p'_1 - A', \quad \overrightarrow{p'_2} = p'_2 - A' \end{aligned} \quad \cdots\cdots (3)$$

$$\begin{aligned} &\vec{L'}(u,v,0) = \left(\frac{-A'_v \times 2}{\nabla\overline{p'_1} - \nabla\overline{p'_2}}, \ \frac{-A'_v \times 2}{\nabla\overline{p'_1} - \nabla\overline{p'_2}} \times \nabla\overline{p'_1} + A'_v, \ 0 \right) \\ &\vec{L} = F^{-1}\vec{L'} \end{aligned} \quad \cdots\cdots (4)$$



[Figure 3] Relationship between line segments and center point



[Figure 4] line segment on uv plane

4 3-D Primitive Reconstruction from Recovered Lines

If the camera calibration process is applied using the primitive, the camera is used for the 3D recovery process of the line using the midpoint information. Finally, the line is used to add the local coordinate LC and dimension vector λ for the recovery of the whole primitive. The primitive is defined as a 1D function for the LC coordinate, which makes it possible to acquire each point of the primitive by acquiring the LC and λ values. LC is used to determine the coordinate of the object. Its value should be determined to minimize the dimension of the dimension vector.. For example, if one of the 3 axis of LC is set to normal in a rectangle, the other two axis will exist on the plane of the rectangle and the size vector λ can be expressed 2-dimensionally. If at least one axis does not match the normal of the rectangle, the dimension of the

dimension vector λ must be 3. These elements should be defined when defining the primitive. The distance length information of the recovered line are used to determine the size of the LC coordinate direction. In the case of the rectangle, the two diagonal lines are projected onto the X and Y axis, to use their respective size as a dimension vector for each axis.

In case of a hexahedron, the X, Y, and Z coordinate of each face are set for the normal vector of the recovered face, that is adjacent, to be a different axis. The error rate is equally distributed over each face and recovered 3-dimensionally to make the adjacent faces to be rectangular.

$$LC\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{L1 + L2}{\|L1 + L2\|} \\ \frac{L1 - L2}{\|L1 - L2\|} \\ x \times y \end{pmatrix} \quad \cdots\cdots (5)$$

$$\lambda' = (L1 \bullet LCx\ ,\ L1 \bullet LCy,\ 0)$$

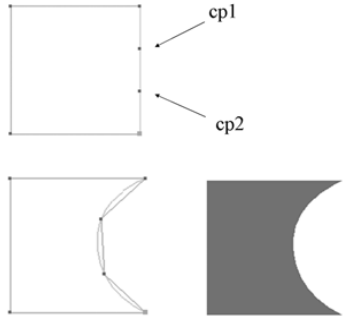
λ represents the dimension vector when the distance between the camera and the midpoint of the primitive model is 1. The point reflected onto the plane must be maintained regardless of the recovered primitive model and the distance from the camera. Therefore, λ should be scaled according to the distance from the camera. Assuming that ω represents the distance between the camera and the midpoint of the primitive model, λ' can be defined as λ*ω.

$$\lambda = \lambda' * \omega \quad \cdots\cdots (6)$$

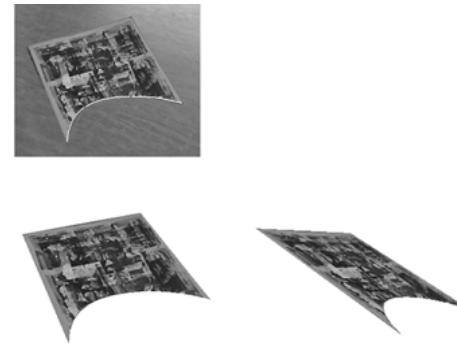
5 Definition and Reconstruction of 3-D Curved

The straight lines of the primitive are transformed into curved lines through user input upon the recovery. The curved line is restricted to a single plane. Even with such restriction, the user input is limited to an image plane which makes the 3D curve hard to control. Therefore, the plane that the curve exists is defined when the primitive is defined. The B-Spline curve is used in this study with 2 control points(cp1,cp2) to minimize the use of user input. The starting point and end point of the line are each recovered 3-dimenstionally when recovering the primitive. The control points are added inside the straight lines to create the curve. The two control points are controlled on the screen for the matching with the outline. The curve plane should be located so that the start and end point are included in the process of defining the primitive. After the curve plane has been defined, the user input control points(cp1,cp2) must be projected onto the curve points. The B-Spline curve is constructed using the projected control points (cp'1,cp'2) and the start and end point of the recovered primitive. [Figure 5] shows the process of defining the curvature through defining a line of the rectangular primitive as a curve and using the control

points. [Figure 6] is the result of 3-dimensionally restructuring the primitive in [Figure 5].



[Figure 5] Curved primitive



[Figure 6] Reconstruction of curved primitive

6 Result

We have performed a 3D recovery of an object using the primitive. The primitive is used to recover 3D information from a single input image. When the user maps each point of the primitive onto the image, the line segments including the midpoint information is extracted by its geometric characteristics.

[Figure 7] is a hexahedron that is most suitable for use in recovering the object from a single image. The normal of each face is perpendicular and exists in pairs of four which makes it suitable for use in a 3D reconstruction. The result shown in [Figure 7] has been reconstructed by getting 8 points of the hexahedron as user input and using the four points of a single face through the process explained in chapter 4.

[Figure 8] is a primitive of a roof shape and its size vector consists of four elements. Two elements have been used for the height and length of the roof.

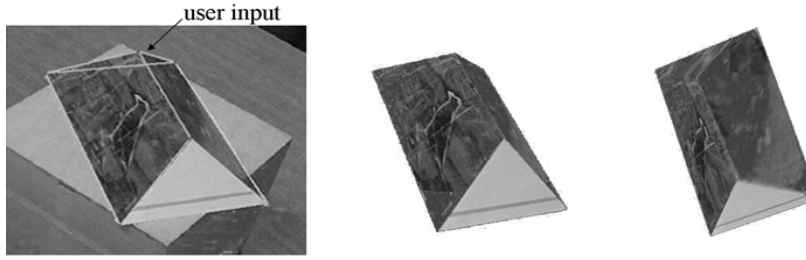
[Figure 9] used the cylinder instead of the straight primitive in the definition process. The user uses the lines used for the recovery of the circle and the cylinder. The two diameters in the bottom part intersect at the center of the circle and bisect each other.

[Figure 10] shows the result after the curve has been added as input. To recover the tile-roofed shape, primitives of roof shape and hexahedron have been reconstructed. It has been restructured by mapping each point of the primitive defined as a straight line. By using the information of each point of the restructured model, the plane with the curve is defined as described in part 5.



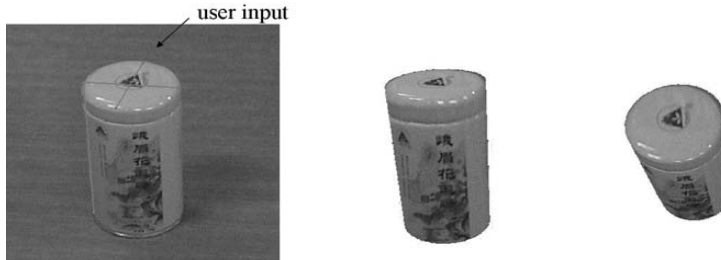
(a) user input (b) 3D reconstruction and rendering

[Figure 7] Hexahedron primitive model



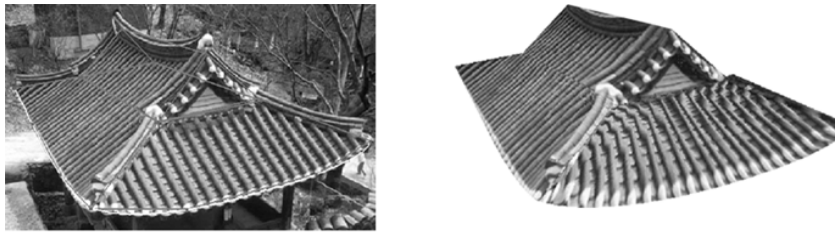
(a) user input (b) 3D reconstruction and rendering

[Figure 8] Pyramid primitive model



(a) user input (b) 3D reconstruction and rendering

[Figure 9] Cylinder primitive model



(a) user input (b) 3D reconstruction and rendering

[Figure 10] Tile-roofed primitive model

7 Conclusion

We have extended information required for the recovery process through enhancing the model-based 3D reconstruction method. It only requires minimum focal length and divides the information required for the camera calibration and the recovery of the primitive. Only the focal length is calculated in the camera calibration process, which makes the camera calibration and 3D reconstruction process of the model possible using a single image. The mid point information is newly adopted in the recovery of the primitive to maintain the characteristics of the recovered model and increase the types of primitive to be recovered.

In case of recovering a pre-defined polygon, the characteristics of the polygon were not efficiently utilized in previous studies. The vanishing points information that have been generally used is easy to calculate but always requires more than two parallel lines and more than three vanishing points. The lack of these two elements would require iteration, which is time-consuming and restricts the types of model that can be recovered. It also makes the number of vanishing point vary according to the location or direction of the model. This means that the information required for the recovery is variable and that the shape and form of the object to be recovered becomes the main issue in the recovery. We have adopted a method that utilizes the characteristics of the object itself instead of the variable information available. The line extracted from the characteristics of the polygon and the midpoint of this line are used for the recovery. This method makes the acquisition of various information possible which restricts the types of polygon to be recovered. It is not affected by the location or direction of the object itself. It also makes the recovery of objects from which the vanishing points cannot be extracted possible, such as a ellipsoid or cylinder. The curves are recovered by adding user input to each line of the recovered object.

Further research should concentrate on enabling the user to pre-define the primitive containing the midpoint information and also on defining the relationship between the extended primitive. Further research is also required on the recovery of a curved line. The plane where the curved line exists must be restricted in the stage where the primitive is defined for the recovery of the curved line, and future work should try to minimize such restriction. Also, because the variation on the curved line occurs

independently, the shape of the model itself is also restricted. Therefore, further research should be conducted on combining a number of lines to form a curved line.

Acknowledgement : This work was supported by the Basic Research Program of the Korea Science and Engineering Foundation (No. R01 -2000-000-00274-0(2002))

Reference

[1] M. Pollefeys, R. Koch and L. Van Gool, "Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters," *Proc. ICCV'98* (international Conference on Computer Vision), pp.90-95, Bombay, 1998

[2] B.S. Villa Alvarez, P.C. Pinto Carvalho and Marcelo Gattass, "Insertion of Three-Dimensional Objects in Architectural Photos*," Proc. WSCG2002*, pp. 17-23, 2002.

[3] D. Jelink and C.J. Taylor, "Reconstruction of Linearly Parameterized Models from Single Images with a Camera of Unknown Focal Length," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 23(7), 2001

[4] P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. SIGGRAPH '96*, pp. 11-21, Aug. 1996.

[5] B.Caprile and V.Torre, "Using Vanishing Points for Camera Calibration," *Int'l J. Computer Vision*, vol. 4, no.2, pp. 127-140, Mar. 1990

[6] Camillo J. Taylor and David J. Kriegman, "Structure and Motion from Line Segments in Multiple Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11), 1995

[7] Sabry F. El-Hakim, "3D Modeling of Complex environments," *SPIE Proceedings*, Vol 4309, Videometrics VII, 2001

[8] Marta Wilczkowiak, Edmond Boyer and Peter Sturm, "3D Modelling Using Geometric Constraints a Parallelepiped Based Approach," *ECCV*, vol. 4, pp. 221-236, 2002

[9] Cipolla R., Boyer E., "3D Model Acquisition from Uncalibrated Image," *IAPR Workshop on Machine Vision Application*, pp. 559-568, 1998

[10] Lieboqitz D. and Zisserman A., "Metric Rectification for Perspectie Images of Planes," *CVPR*, pp. 482-488, 1998

[11] Z. Zhang, "Image-based Geometrically-correct Photoreallistic Scene/Object Modeling (IBPhM): A review," *Proc. Of the Asian Conference on Computer Vision* (ACCV), 1998

[12] D. Liebowitz, A. Criminisi A., and Zisserman A., "Creating Architectural Models from Images," *EUROGRAPHICS '99*, 18(3), 1999

[13] Armstrong M., Zisserman A. and Beardsley P, "Euclidean Structure from Uncalibrated Images," *Proc. Britich Machine Vision Conference*, pp. 509-518, 1994

CIGRO: A Minimal Instruction Set Calligraphic Interface for Sketch-Based Modeling

Manuel Contero[1], Ferran Naya[1], Joaquim Jorge[2] and Julián Conesa[3]

[1] DEGI - ETSII, Universidad Politécnica de Valencia, Camino de Vera s/n,
46022 Valencia, Spain
{mcontero, fernasan}@degi.upv.es
[2] Engª Informática, IST, Av.Rovisco Pais
1049-001 Lisboa, Portugal
jorgej@acm.org
[3] DEG, Universidad Politécnica de Cartagena, C/ Dr. Fleming
30202 Cartagena, Spain
julian.conesa@uptc.es

Abstract. Although CAD systems have evolved considerably in functionality, expressiveness and modeling power over the last decades, their user interfaces are still tied to legacy principles and are not suited to the initial stages of product development. They exhibit steep learning curves, cumber-some and overly structured dialogues, including hundreds of commands. While much of this functionality may be required by the sheer complexity of the tasks these systems are designed to help, we believe the user interface could benefit from simpler paradigms based on sketching and drawing to reduce unneeded complexity, especially in the conceptual design phase. In what follows, we present the CIGRO system that provides a reduced instruction set calligraphic interface to create polyhedral objects using an incremental drawing paradigm evocative of paper and pencil drawings. Users draw lines using an axonometric projection, which are automatically beautified and connected to existing elements of the drawing. Such line drawings are then converted to a three-dimensional model through a reconstruction process guided by an axonometric inflation method.

1 Introduction

In spite of large strides made by commercial CAD applications in terms of descriptive power, flexibility and functionality, their user interfaces are still by and large constrained by the WIMP (Windows, Icons, Menus and Pointing) paradigm, which severely hinders both ease of learning and ease of use. Recently, work on sketch-based modeling has looked at a paradigm shift to change the way geometric modeling applications are built, in order to focus on user-centric systems, rather than systems that are organized around the details of geometry representation. Some of this work stems from early attempts of shape interpretation in Computer Vision. While most of the activity in this area in the past has been focused in off-line algorithms, the growing focus on sketches and modeling has brought forth a new emphasis on approaches geared towards interactive applications. To this end, the aim of our research is to

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 549-558, 2003.
© Springer-Verlag Berlin Heidelberg 2003

develop expeditious ways to construct geometric models. We want to generate solid and surface models from two-dimensional freehand drawings, using a digitizing tablet and a pen, an approach we have termed calligraphic interfaces. These rely on interactive input of drawings as vector information (pen-strokes) and gestures, possibly coupled with other interaction modalities. We want to keep the number of low level interactions minimal, as well as the command set (number of different gestures/strokes the user has to enter). The present text describes work at the user interface towards integrating a three-dimensional reconstruction approach into an interactive working environment, through sketch input. This environment differs markedly from previous approaches in that the speed of execution and timely feedback are more important than the ability to produce models from vectorized bitmaps in one pass as typical of previous efforts in computer vision.

2 Related Work

The new generation of calligraphic applications uses gestures and pen-input as commands [1], [2], [3]. In contrast to conventional drawing applications, the stylus can also be used to enter continuous-mode sketches and freehand strokes. Thus, there is a growing research interest on using freehand drawings and sketches as a way to create and edit 3D geometric models. Within this research area we can distinguish two approaches. One method relies on gestures as commands for generating solids from 2D sections. The second approach, derived from computer vision, uses algorithms to reconstruct geometric objects from sketches that depict their two dimensional projection.

An example of *gestural modeling* is Sketch [4]. The geometric model is entered by a sequence of gestures according to a set of conventions, regarding the order in which points and lines are entered as well as their spatial relations. Quick-Sketch [5] is a computer tool oriented to mechanical design, consisting of a 2D drawing environment which is based on constraints. From these it is possible to generate 3D models through modeling gestures. Teddy [6] allows free form surface modeling using a very simple interface of sketched curves, pockets and extrusions. Users draw silhouettes using a series of pen strokes and the system automatically proposes a surface using a polygonal mesh whose projection matches the object contour. GIDeS [7] allows data input from a single-view projection. In addition the dynamic recognition of modeling gestures provides users with contextual menus and icons to allow modeling using a reduced command set.

The second approach, which we call *geometric reconstruction*, uses techniques based on computer vision to build three-dimensional geometric shapes extracted from two-dimensional images, representing some kind of projection. The systems we surveyed use two main techniques. The first is based on Huffman-Clowes labeling scheme [8], [9]. The second approach treats reconstruction as an optimization problem [10]. This enables us to obtain what from the point of view of geometry is unrealizable: a three-dimensional model from a single projection. However, from the psycho-logical point of view it is well-known that humans can identify 3D models from 2D images, using a simple set of perceptual heuristics. Thus geometric reconstruction,

when cast as a problem of perception, can be described in terms of mathematical optimization. To this end reconstruction applications have been developed by authors such as Marill [11], Leclerc [12], Fischler and Lipson [13]. Other examples are Digital Clay [14] that supports basic polyhedral objects combined with a calligraphic interface for data input, uses Huffman-Clowes algorithms to derive three-dimensional geometry and Stilton [15] where a calligraphic interface is directly implemented in a VRML environment, and the reconstruction process uses the optimization approach based on genetic algorithms.

3 The CIGRO System

In contrast to surveyed work, our application provides and integrated 2D-3D environment, where users sketches and immediately can switch the point of view and see the corresponding 3D model. This real time sketching experience is supported by implementing a minimal gesture alphabet, automatic line drawing beautification and a fast and robust axonometric inflation engine.

The objective of our system is provide support for the preliminary phases of design, where not it is not necessary to build complex CAD models but expressing our visual thinking in a fast way. In this context, engineers usually employ sketches as their natural tool to express design concepts. So we have chosen a compromise between geometric complexity and fast interaction. For this reason, at this moment only polyhedral models are supported, in order to implement a fast axonometric inflation algorithm to reconstruct them.

Previous reconstruction based applications, include a preliminary offline 2D reconstruction stage, where the input sketch is adjusted before to proceed with 3D reconstruction. To implement a true interactive sketching system we have developed an automatic online 2D reconstructor that operates in real time. So, whenever the input sketch is changed, either because new edges are added or deleted, the sketch is adjusted, and automatically a 3D model is built and offered to the user for review. The only constraint on drawing is that the reconstruction procedure requires a single orthogonal axonometric projection of the model as input. These are the single-view representations most commonly used in engineering drawings of three-dimensional parts.

3.1 Gesture Analyzer

Gesture analyzer processes strokes generated by the user directly on the screen of a Tablet-PC or LCD tablet as shown in Fig. 1. The user interface is designed to minimize (we call it minimalist interface) the interaction with the system in an attempt to approximate it to the traditional use of pen and paper. We want user concentrates in sketching instead having to navigate through menus or searching for icons in toolbars. For that reason application automatically generates the corresponding three-dimensional geometric model, and it does not need any additional information from the user. The system updates the 3D model as the user refines the geometry, so users

can see the evolution of the model, whenever any change in the input sketch takes place.

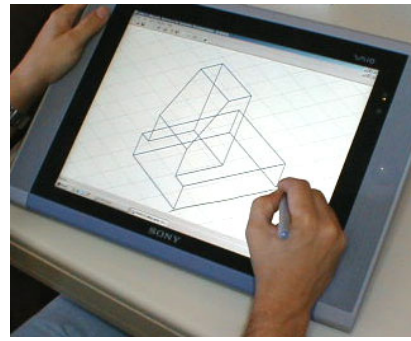


Fig. 1. Example showing CIGRO operation

The gesture set used by the interface is reduced to the following commands: *new edge*, *new auxiliary edge* and *remove edge* (auxiliary or not). These gestures are very similar to those used when drawing on paper. The application is, therefore, in charge of recognizing the type of stroke drawn by the user, and to try and capture the designer's intention (parallelism, perpendicularity, proximity, close points).

The application provides both real geometry and auxiliary lines. This emulates an extended practice for making sketches. First, the user draws a set of auxiliary lines (see Fig. 1.a to 1.e) to define the main geometric features of the object, and then, using this skeleton as a drawing template the designer refines the sketch, making pressure with the pencil, and drawing over the previous template (Fig. 1.f). Our application takes into account the pressure made on the pencil. Of this way it distinguishes auxiliary constructions from real geometry lines, by applying a pressure level threshold (configured by user). Auxiliary strokes tend to serve as references (constraints) for the intended model geometry strokes. This is implemented in the application using the Wintab API (http://www.pointing.com), an open industry interface that directly collects pointing input from a digitizing tablet and passes it in a standardized fashion to applications. This API allows the possibility to retrieve the pressure the user applies at each point of the stroke over the tablet. This information is used by the application to distinguish between those auxiliary entities (less pressure) from those of the model (higher pressure). Raw strokes provided by Wintab API are processed by the CALI library [3] that provides some components to develop calligraphic interfaces. It is based on a recognizer of elemental geometric forms and gestural commands, that operates in real time using fuzzy logic. The recognized gestures are inserted in a list order by degree of certainty, and returned to the application. CALI recognizes the elemental geometric shapes, like triangles, rectangles, circles, ellipses, lines, arrows, etc, and some gestural commands, such as delete, move, copy, etc.

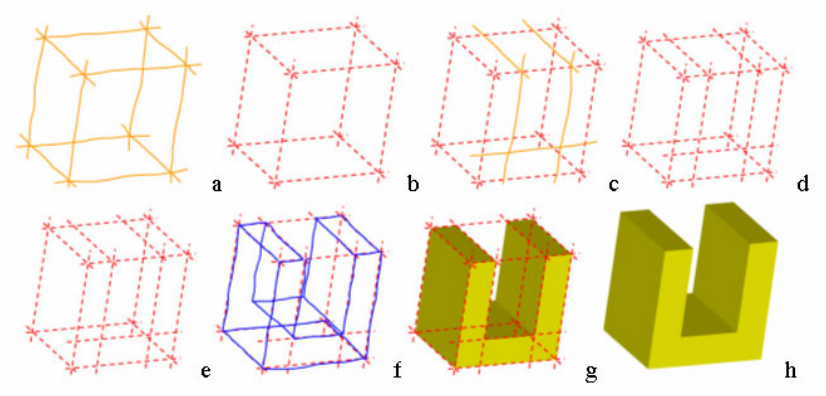


Fig. 2. Example showing sequence of actions extracted from CIGRO. Orange color in **a** represents raw strokes corresponding to auxiliary lines. Cyan dashed lines in **b** and **d** represent adjusted auxiliary lines by gesture analyzer. Blue raw strokes in **f** correspond to real geometry that is snapped to auxiliary lines in **e** that provides the solid model in **h**. All of these representations can be switched on or off by means of a visualization toolbar

At the current development level, CIGRO application only supports sketched segments which can be recognized as entities of "line" type or as a gestural command of "delete" class. In this way, the application analyzes the typology of the sketched entity, and if it corresponds to a line or to the command "delete", the entity is processed.

Users are allowed to generate faces drawing a sequence of straight edges. It is not relevant in which order edges are drawn, since the three-dimensional reconstruction method only looks at connectivity and perceptual constraints. In this way we try to afford a freedom in sketching mimicking that of pencil and paper.

Edges and segments can be removed using a scratching gesture. This allows not only to correct errors but also to draw more complicated shapes from □simpler□ forms, an idiom familiar for draftspeople used to sketching. When drawing a scratch gesture, the application detects the edge(s) that the user wants to delete as those intersecting the smallest quadrilateral enclosing the scratching gesture (see Fig. 3.d and 3.e). This can be used to incrementally derive more complex shapes from simpler ones, as illustrated in Fig. 3, where users can construct shapes by either adding new edges or removing existing ones. The interface cooperates with the user by not requiring that valid models are present at all times. It rather allows geometric models which are consistent only at the face level.

3.2 Line Drawing Beautification and Snaps

Previous reconstruction based applications usually include an offline 2D reconstruction stage, where the input sketch is adjusted. In our system we propose an online 2D reconstruction [16]. Online reconstruction provides an immediate feedback to the user, because it operates as the user draws the sketch, and it offers better integration in the calligraphic interface.

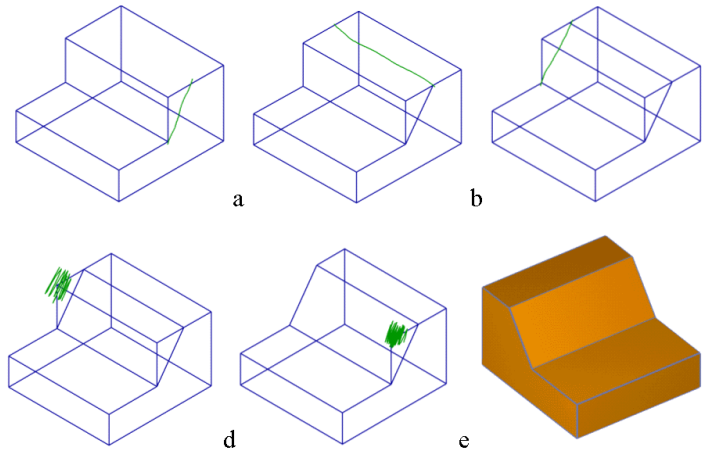


Fig. 3. Refining geometry from simple shapes

Our concept of 2D reconstruction is similar to drawing beautification proposed by Igarashi [6], [17]. The aim of this stage is adjust drawing entities provided by the CALI recognizer to be used at the stage of 3D reconstruction. To obtain functional input data for 3D reconstruction, we need to cleanup input data and adjust edges to make sure they meet precisely at common endpoints to get geometrically consistent figures which can then be used for generating 3D models at the next stage. The "beautification" process has to filter all the defects and errors of initial sketches which are inherent to their inaccurate and incomplete nature. At present, the stage of 2D reconstruction receives as input data either geometric shapes of type "line" or "auxiliary line". In order to provide an adequate database for the 3D geometric reconstructor, the application provides support for the following drawing aids: *automatic line slope adjustment*, *vertex point snap* and *vertex on line snap*. The first drawing aid consists of checking whether the new line is parallel to any of the principal axes of the sketch, considering a *slope tolerance*. In the case that the straight line is nearly parallel to one axis, then we adjust one or both endpoints so that the resulting line is now precisely parallel line to one of the three main axes. The second analysis looks for vertices close to the line endpoints, again taking into account a *vertex proximity tolerance*. In the case there are several such vertices, we select the one with the closest to that line endpoint. For endpoints of the new line which do not lie close to a model vertex, the system analyzes whether the points are close to an existing edge, accounting for a given *edge proximity tolerance*. If several edges match this criterion, we select the edge which lies closest to the given endpoint. Snapping capability provides a continuous feedback to the user, because it is performed in real time. Other previous systems perform these analysis offline, when the user has finished his sketch, before to launch 3D reconstruction. We provide a general tolerance control to soften the beautification action, because some users prefer a less automatic drawing control. After this stage all the 2D image data are stored in a database composed by a list of vertices and a list of edges.

3.3 Axonometric Inflation Engine

In order to provide a real time experience, we have implemented a fast reconstruction algorithm whose clearest antecedents are presented in [18] and [19]. Other reconstruction strategies based on optimization or labeling algorithms have been discarded because they are not fast enough. For this reason we have restricted the shape domain supported by CIGRO to *quasi-normalon* objects. We use an extension to 3D of Dori's definition of a *normalon* [20], as "a polyhedral object having the property that the angle between any two of its adjacent edges is 90°". Quasi-normalon typology corresponds to objects that can be reduced to normalon by deleting all edges non parallel to the three main directions without loosing any vertex on model.

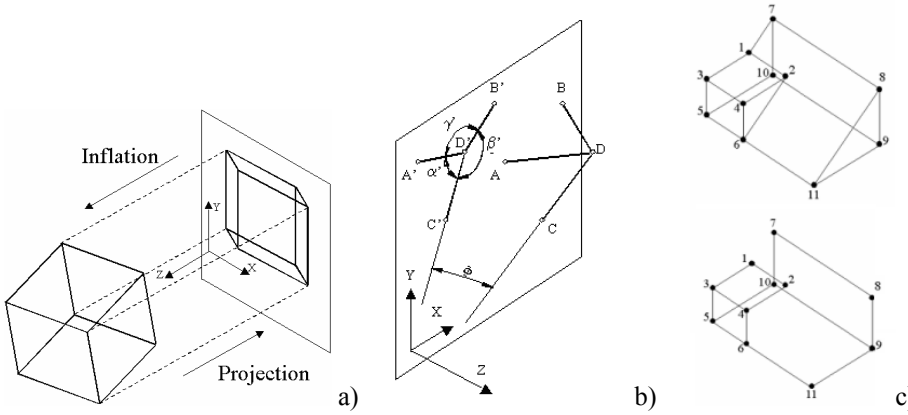


Fig. 4. Inflation method (a), relationship between angles in model and projection (b) and quasi-normalon concept (c)

Let us suppose an orthogonal projection of an orthogonal corner like A'B'C'D' (see Fig. 4.b). To reconstruct the model (i.e. to determine x, y and z coordinates of vertices A, B, C and D) the Cartesian coordinate system associated to inflation is used to trivially obtain x and y coordinates of all four vertices ($x_A = x_{A'}$, $y_A = y_{A'}$, ...) and z coordinate of central vertex D can be arbitrarily fixed without loss of generality. Then, axonometric inflation takes profit from formulations that determine the angle between every edge (CD) and its own projection (C'D') when three orthogonal edges are connected to the same central vertex (D). Next, relative z coordinates of neighbor vertices (A, B, C) are obtained from z coordinate of central vertex (D) in the following way:

$$z_C = z_D \pm L_{C'D'} \cdot \tan(\operatorname{asin}(\sqrt{\operatorname{cotg}(\alpha) \cdot \operatorname{cotg}(\beta)})) \quad (1)$$

Where $L_{C'D'}$ is the length of line segment C'D' (CD edge projection), and z_C and z_D are the respective z coordinates of the lateral and the central vertex that determine the reconstructed edge. To determine z coordinates of all vertices in a normalon polyhedron, a Kruskal algorithm is used to obtain a spanning tree formed by all edges connecting the successive central vertices. The lateral vertex connected through the longest edge is defined as the central vertex for the new corner; because it is assumed

that longer edges are less prone to error than shortest do (i.e. the dimensional relative errors are minor, and long lines are more accurately drawn in hand drawing). The process is repeated until all vertices in the model have been determined. In case where converting any of the present lateral vertices in central ones would generate a circuit (i.e. if all lateral vertices have been previously visited) the branch is abandoned and the longest lateral edge not yet explored is used to begin a new branch. Additionally, particular cases must be studied to prevent numerical inconsistencies during the process (for instance due to trigonometric operations).

The algorithm requires three orthogonal edges concurring in every central vertex. Nevertheless, information of faces is not required and the restriction applies to central vertices only. The valence (the number of edges concurring in a vertex) of lateral vertices is irrelevant. Consequently, the approach will work if a spanning tree can be obtained where all central vertices have a valence of three, and, thus, all vertices of other valences can be determined as laterals. Of this way, axonometric inflation can be applied to the *extended* class of polyhedrons whose spanning tree is normalon. In those quasi-normalon models (see Fig. 4.c), the temporary elimination of all line-segments that are non-parallel to any of the three main directions determines an equivalent normalon if no vertex is deleted and the graph remains connected.

Sometimes, in spite of getting an equivalent normalon, axonometric inflation cannot be applied if certain vertices are not accessible through some valid spanning tree. This is the case when some junctions are only connected to other junctions of valence less than three. For instance, in Fig. 5, vertex 3 is only connected to vertex 4, which cannot be central because its valence is two. The same happens to vertices 6 and 7. When the valence of a central vertex is minor than three, equation (1) cannot be used. Nevertheless, the assumption of the model to be a normalon has already been made. Hence, adding fictitious line-segments is coherent with the assumption and solves the problem. Those fictitious line-segments are defined of unit length and oriented in accordance with those main directions still not present in the vertex. In Fig. 5, fictitious edges 4-F1, 12-F2 and 6-F3 allow vertices 3, 6 and 7 to be determined. When vertices of valence greater that three appear, the approach still will work if a valid spanning tree can be obtained; i.e. a tree where those vertices are not central. When this is not possible, we have confined ourselves to obtain one of the potential models, by randomly choosing three of the edges that converge in the vertex

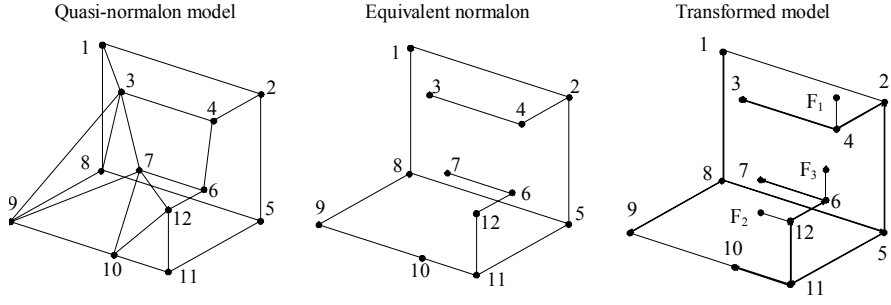


Fig. 5. Transformation of quasi-nomalon model

4 Conclusions and Future Work

While much work remains to be done, preliminary results garnered from experimenting with the current prototype are very encouraging. The calligraphic approach to modeling allows users to focus on the drawing task rather than on the subtleties of geometric representations and dialogues. This seems to offer great advantages as testified by the lesser number of steps as well as time needed to complete the drawing tasks, relative to conventional modeling systems. The results obtained so far, constitute a strong incentive to extending and improving our approach towards more sophisticated, yet natural methods to modeling with computers.

In order to reach our final goals it would be interesting to integrate geometric reconstruction as presented here with more conventional sketching application such as GIDeS [7] to endow our application with a richer set of modeling primitives. Another challenging direction of work consists on incorporation curved shapes and surfaces to our reconstruction methods. This may require non-trivial extensions to our modeling approach, although much of the intended functionality may be accomplished through a suitable combination of constraints and interfacing a solid modeling kernel which we have not done so far.

Acknowledgments

This work was supported in part by the Portuguese Science Foundation grant 34672/99, the European Commission project IST-2000-28169 and the Spanish Generalidad Valenciana grant CTIDIB/2002/51.

References

1. Rubine, D.: Combining gestures and direct manipulation. Proceedings ACM CHI'92 Conference Human Factors in Computing Systems (1992) 659-660
2. Long, A.C., Landay, J.A., Rowe, L.A., Michiels, J.: Visual Similarity of Pen Gestures. Proceedings of Human Factors in Computer Systems (SIGCHI), (2000) 360-367
3. Fonseca, M., Jorge, J.: Experimental Evaluation of an On-Line Scribble Recognizer. Pattern Recognition Letters, **22** (12), (2001) 1311-1319
4. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: SKETCH: An interface for sketching 3D scenes. SIGGRAPH'96 Conference Proceedings (1996) 163-170
5. Eggli, L., Hsu, C., et al.: Inferring 3D Models from Freehand Sketches and Constraints. Computer-Aided Design, **29** (2), (1997) 101-112
6. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. ACM SIGGRAPH99 Conference Proc. (1999) 409-416
7. Pereira, J., Jorge, J., Branco, V., Nunes, F.: Towards calligraphic interfaces: sketching 3D scenes with gestures and context icons. WSCG'2000 Conference Proc. Skala V. Ed. (2000)
8. Huffman, D.A.: Impossible Objects as Nonsense Sentences. In: Meltzer B., Michie D. (eds.) Machine Intelligence, No. 6, Edimburgh UK. Edinburgh University Press (1971) 295-323
9. Clowes, M.B.: On Seeing Things. Artificial Intelligence, **2**, (1971) 79-116

10. Wang, W., Grinstein, G.: A Survey of 3D Solid Reconstruction from 2D Projection Line Drawing. Computer Graphics Forum, **12**(2), (1993) 137-158
11. Marill, T.: Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects. International Journal of Computer Vision, **6**(2), (1991) 147-161
12. Leclerc, Y., Fischler, M.: An Optimization-Based Approach to the Interpretation of Single Line Drawing as 3D Wire Frames. Int. Journal of Computer Vision, **9**(2), (1992) 113-136
13. Lipson, H., Shpitalni, M.: Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing. Computer Aided Design, **28**(8), (1996) 651-663
14. Schweikardt, E., Gross, M.D.: Digital Clay: deriving digital models from freehand sketches. Automation in Construction, **9**, (2000) 107-115
15. Turner, A., Chapmann, D., and Penn, A.: Sketching space. Computers & Graphics, **24**, (2002) 869-879
16. Oh, B.S., Kim, C.H.: Progressive 3D Reconstruction from a Sketch Drawing. In Proceedings of the 9th Pacific Conf. on Computer Graphics and Applications, (2001) 108 -117
17. Igarashi, T., Hughes, J.F.: A Suggestive Interface for 3D Drawing. 14th Annual Symposium on User Interface Software and Technology, ACM UIST'01, Orlando, Fl., (2001) 173-181
18. Kanatani, K.: The Constraints on Images of Rectangular Polyhedra. IEEE Transactions on Pattern Analysis and Machine Intelligence, **8** (4), (1986) 456-463
19. Lamb, D., Bandopadhay, A.: Interpreting a 3D Object from a Rough 2D Line Drawing. Proceedings of Visualization'90, (1990) 59-66
20. Dori, D.: From Engineering Drawings to 3D CAD Models: Are We Ready Now?. Computer Aided Design, **27** (4), (1995) 243-254

MOTRICO Project: Geometric Construction and Mesh Generation of Blood Vessels in Coronary Bifurcation

Francisco J. Seron, Elsa Garcia, and Jorge del Pico

Engineering Research Institute of Aragon,
Grupo de Informatica Grafica Avanzada, Centro Politecnico Superior
Universidad de Zaragoza, Maria de Luna 1,
50018 Zaragoza, Spain
seron@posta.unizar.es, http://giga.cps.unizar.es

Abstract. The MOTRICO project plans the development of an advanced environment that will offer computer assistance for cardiac therapy and diagnosis, which would be useful in the hemodynamics units of those hospitals that have access to the instrumental techniques of Angiography and Intravascular Ultrasounds. The initial work presented in this paper will describe modeling and finite element mesh generation of two anatomically realistic models of the human left coronary artery bifurcation. The first computational geometric model has been developed on the basis of real anatomical information; the second one has a coronary vessel segment developed on the basis of the information obtained by means of the fusion of angiograms and intravascular ultrasound images(IVUS).

1 Introduction

The MOTRICO project plans the development of an advanced environment that will offer computer assistance:

- For constructing an anatomically realistic model of segments of the human vascular system. This three-dimensional geometric model is generated on the basis of the information obtained by means of the fusion of angiograms and intravascular ultrasound images(IVUS).
- For simulating blood flow through arteries in order to calculate the wall shear stress distribution in these arteries. Therefore, development of non-linear models of the material that will represent the behaviour of the biological soft tissues (hyperelastic models, reologic behaviours, two-phase matched models, etc.)will be necessary.
- For allowing the user to interact with the system as friendly, quickly and intuitively as possible using virtual and augmented reality techniques for the visualisation of 3D data.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 559–568, 2003.
© Springer-Verlag Berlin Heidelberg 2003

1.1 The Coronary Tree

The first thing that needs to be taken into account in order to model and to mesh the vascular system is that his structure is similar to a tree: the heart pumps blood through a complex network of smaller and smaller branching vessels with the purpose of distributing it to different organs. The term "tree" is often used to designate this arterial network because its structure is produced by the division of a single trunk, called the parent vessel, into two or more child vessels. This is the main reason that coronary arteries are called "The Coronary Tree".

As for the number of child vessels, it is interesting to consider that according to a study on the branching characteristics of human coronary arteries all branching sites encountered were arterial "bifurcations", where the parent vessel undergoes a dichotomous division [1]. To which might be added that, according to another study of human coronary artery geometry, the left main artery terminated in a trifurcation in only seven out of thirty cases [2].

Thus it follows from all this that the key to coronary tree mesh generation must surely be bifurcation mesh generation.

2 Geometric Model Construction of the Left Coronary Bifurcation

2.1 Geometry and Morphology

The root of the Left Coronary Tree is the left main coronary artery (LM or LMCA). This artery arises from the aortic sinus of Valaslva and it covers around 1 cm before it branches into two slightly smaller arteries: the left anterior descending coronary artery(LAD) and the left circumflex coronary artery(LCX).As far as morphology is concerned, there is a large body of literature that deals with quantification of coronary diameters using angiography and IVUS images [3] [4]. Moreover, Zamir and Chee (1987) carried out measurements of lengths and diameters in a total of 1614 vessel segments in two human hearts [5]. Regarding branching characteristics, two principles suffice to model the human coronary network and to determine diameters of child vessels and branch angles: the principle of minimum pumping power and the principle of minimum volume [6] [7]. In addition, there are several studies on coronary branching sites [1] [2] [8].Finally, it is important to mention that the blood vessel wall consists of three layers: the intima, the media and the adventitia. The intima is the innermost layer, the media is the middle one and the adventitia is the outermost layer. The thickness of the media layer is between two and three times the thickness of the adventitia.

Dimensions in Lengths, Diameters, Branching Angles and Thickness of the Bifurcation Model. According to the geometry information described in this section, an idealized geometric model of the LM-LAD-LCX bifurcation is modeled (See figure 4) on the following dimensions: The lengths are 0.9, 2.7 and 1.06 cm for the LM, LAD and LCX, respectively. The vessel of the LM artery has

an average diameter of 3,8 mm. The LAD and LCX vessels have 3,1 and 2,7 mm for initial diameters and 2 and 2,3 mm for final diameters, respectively. The total branch angle is 76 and the angles between the LM-LAD and the LM-LCX are 116 and 168. It is easy to observe that these dimensions in angles and diameters are in keeping with the principles cited in this section. The thicknesses are 0.1, 0.3 mm for the adventitia and media layers, respectively. The thickness of the intima layer is considered null because it consists only of the endothelium.

2.2 Geometric Modeling Techniques

The following operations have been performed in order to construct the bifurcation geometric model :

- A semi-circle with the LMCA's diameter is created and the bifurcation trajectory is described using three cubic NURBS curves.
- Three NURBS surfaces are created by extruding the semi-circle along each of the NURBS curves.
- In order to eliminate the hole formed where the three surfaces meet in a complex way, they must meet at a point. However, this can pose a problem, because simply joining them at the central point causes overlapping edges. To solve the problem, knots of multiplicity 3 are created. The multiknots produce a discontinuity that helps align the surfaces, so continuity is maintained across the multiknot curves as if they were junctions (See figure 1).
- Similar points in each curve are aligned by activating the techniques called "snap to grid" and "snap to point" [9].

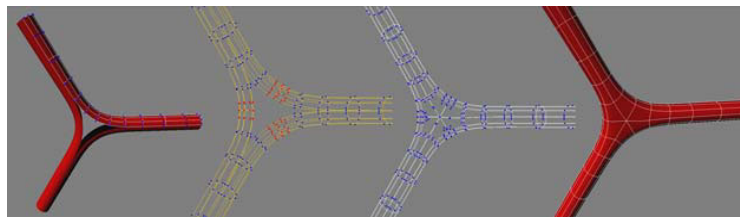


Fig. 1. Alignment by snapping.

- Branching angles and vessel diameters are modified according to the angles cited in the section Sect. 2.1.

The following operations have been performed in order to construct the geometric model of the vessel segments of the idealized arteries [10]:

- The two semi-circular curves of the final cross section in each branch are extruded along a NURB curve that the trajectory of each artery describes in space.

- Cross section diameters are modified according to each individual artery.
- The NURBS surfaces of each branch are deformed with the purpose of obtaining the correct curvature.

These operations have been performed for each of the surfaces that separate vessel wall layers with different diameters. All surface patches are saved as IGES surfaces. Afterwards, different subvolumes required for mesh techniques are composed from these IGES surfaces using boolean operations. Finally, the four models (bifurcation and three vessels) are joined as a single model.

2.3 3D Reconstruction of a LAD Vessel Segment

In the idealized bifurcation model, a LAD vessel segment has been replaced with a 3D reconstruction of the same segment (See figure 4).

This segment has an average length of 1,35 cm. This reconstruction of the LAD segment has been obtained from the sequence of IVUS images and angiographies. Both methods (IVUS and angiogram) provide a lot of information on the internal and the external shape of the coronary vessels.

The vessel model is reconstructed using deformable models and compounding methods. Deformable models are very well suited for lumen and vessel wall detection, as they allow modeling of the vessel via an elastic dynamic model that adjusts to the image features for reconstructing the vessel. Once the vessel boundaries have been detected, compounding methods are used to interpolate IVUS data [11].

This technique is complemented by a 3D reconstruction of the vessel using biplane angiography support. IVUS images are located in space thnaks to a 3D reconstruction of the catheter trajectory on the basis of the registration of the catheter in two views of angiograms before and after the pullback of the IVUS catheter. Note that this spatial curve represents the trace of the centers of the IVUS images. Each IVUS plane is placed in space in order to allow later reconstruction of vessel tortuosity [12].

3 Finite Element Mesh Generation

Coronary vessels could be defined using their cross sections and the trajectory they describe in space. This is the reason that they could be considered to be sweepable volumes, that is, a blood vessel could be meshed using sweeping algorithms. The sweeping algorithms take the 2-D quadrilateral mesh from one cross section surface (source) and project it through the vessel to another cross section surface (target).

3.1 Mesh Generation of Blood Vessels

In the first place, it is necessary to subdivide the vessel into subregions with different physical characteristics suitable for finite element analysis. The vessel

decomposition has as many subvolumes as materials (adventitia, media and intima layers and lumen of the artery). As far as the geometry is concerned, it is possible to distinguish two types of subvolumes:

Ring-Shaped. This type ranges the adventitia, media and intima layers of the artery wall. The method employed for meshing the source surface of this type of subvolumes is called *Hole* (See figure 2). A polar coordinate-like mesh with the singularity removed is produced with this method [13].

Cylinder-Shaped. This ranges the lumen of the artery. The technique used to obtain an unstructured mesh on the 2D cross section of this type of subvolumes is called *Paving* (See figure 2). The Paving technique introduced by Blacker and Stephenson presents a method for forming complete rows of elements starting from the boundary and working inward [14].

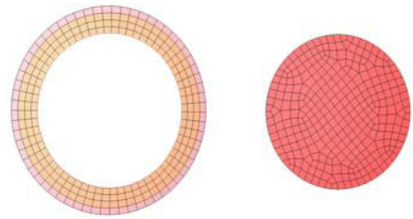


Fig. 2. Bidimensional meshes generated by hole (left) and paving (right).

3.2 Mesh Generation of Bifurcations

A bifurcation is not a sweepable volume because its volume has two sweep directions, one for the trunk and another for its branch. This constraint has been lifted by decompositing the initial volume into sweepable subvolumes. Therefore, the main difference between meshing an isolated vessel and a bifurcation is, in essence, the initial decomposition of the volume. The bifurcation model has been subdivided into three subvolumes according to the surfaces that contain the bifurcation axis and are perpendicular to the bifurcation plane. After that, each of the subvolumes is decomposed by different materials such as blood vessels (See figure 3). The mesh generation techniques employed in each of the three subvolumes have been *Hole* and *Paving* for meshing the source surface of the cross section of adventitia and media layers and of the artery lumen, respectively and *Sweeping* to generate the three-dimensional mesh by projecting the 2D paving and hole meshes through each of the three subvolumes. Finally, the three submeshes of domains with the same material are joined adequately as a single mesh has been generated.

Mesh Generation of Coronary Tree. As a consequence of the decomposition method of the bifurcation and in order to preserve the connectivity of the mesh nodes between bifurcations and artery vessels of the coronary tree, the initial volume of vessels has been divided in half along its axis. Each subvolume will be decomposed and meshed just like the bifurcation subvolumes. No other mesh techniques are required for generating the total left coronary tree mesh, as the process described here is sufficient for this purpose.

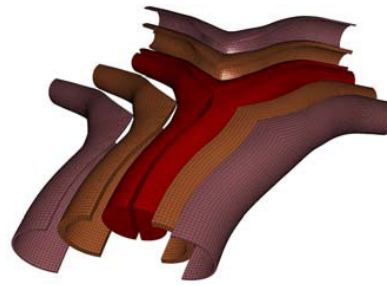


Fig. 3. Mesh Generation of the LM-LAD-LCX bifurcation.Tridimensional meshes of each of three subvolumes.

4 The Problem of Bifurcations

Both the geometric model and the finite element meshes entail certain constraints and problems.

4.1 Constraints Related to Geometric Modeling

Surface Continuity of Geometric Models. An important characteristic of the geometry of coronary arteries is the absence of corners and peaks on their surfaces, so the main constraint related to geometric modeling of coronary vessels is the smoothness and continuity of the boundary surfaces. A frequently used branch-modeling method is to construct the geometric models of trunk and branch separately and to intersect them afterwards. This method was dismissed because of the corners resulting from such intersection.

Initial Decomposition of the Domain. Some constraints related to the later descomposition of the domain have been assessed before geometric modeling of the bifurcation, in order to model the bifurcation surfaces already divided into as many parts as necessary. Such constraints are mainly related to the quality of the mesh elements. In keeping with this, several possibilities of bifurcation

modeling were considered. One of them was to preserve the triangular-shaped hole formed where the three semicylinder-shaped surfaces meet (See Sect. 2.2) and to mesh it as another subvolume. But this option was dismissed because of the low quality of the mesh produced for this subvolume.

4.2 Constraints Related to Mesh Generation Techniques

Hole. In this method, the number of intervals in the azimuthal direction is controlled by setting the number of intervals on the inner and outer bounding loops of the surface. The number of intervals must be the same on each loop. There are usually problems with the correspondance between mesh nodes on the inner and outer boundaries [13].

Paving. The paving boundary must always contain an even number of nodes. This is a necessary condition when generating an all-quadrilateral mesh [14].

Sweeping. To maintain the structured mesh in the sweep direction, sweeping algorithms require that the linking surfaces (those that connect the source to the target) be mappable or submappable. This constraint limits the number of solids that can be meshed with these algorithms. They specifically exclude solids with imprints or protrusions on the linking surfaces as bifurcations. There is an algorithm called grafting that lifts this constraint on linking surfaces. This algorithm has three major steps: meshing of the trunk, modification of the base surface mesh at the graft surface, and meshing of the branch. However, it is not employed in this case because of the decomposition of materials required by the finite element method and because the solids that generally benefit from the grafting algorithm have one major sweep direction which is perpendicular to the other one, that is, solids with corners [15].

5 Geometric Models and Meshes Obtained

Two geometric models of the human left coronary artery bifurcation have been produced. The first computational geometric model has been developed on the basis of real anatomical information; the second one has a coronary vessel segment developed on the basis of the information obtained by the fusion of angiograms and intravascular ultrasound images (See figure 4). A total of amount of 244844 mesh nodes has been created for each of geometric models. Five meshes have been generated for each geometric model(See figure 5):

- The first mesh contains the adventitia layer, which is formed using Hole and Sweeping mesh techniques and is an hexahedral mesh (See figure 6).
- The second one is a bidimensional mesh of quadrilateral elements produced when the first mesh is generated. It covers the outermost surface of the adventitia layer.

- The third mesh contains the media layer and is generated in the same way as the first one (See figure 7).
- The fourth one is a quadrilateral mesh produced as a result of generating the third mesh. It covers the intima layer.
- The fifth one is the lumen mesh of the bifurcation. The paving method is employed to generate the quadrilateral mesh of the cross section surfaces and Sweeping to form the hexahedral mesh through the lumen (See figure 8).

6 Acknowledgement

This work has been partly financed by the Spanish "Ministerio de Ciencia y Tecnologia" (contracts number TIC 2000-1635-C04-01)

References

1. Zamir, M., Chee, H.: Branching Characteristics of Human Coronary Arteries. Can. J. Physiol. Pharmacol., Vol. 64.(1986) 661–668
2. Brinkman, A.M., Baker, P.B., Newman, W.P., Vigorito, R., Friedman, M.H.: Variability of Human Coronary Artery Geometry: An Angiographic Study of the Left Anterior Descending Arteries of 30 Autopsy Hearts. Can. J. Physiol. Pharmacol., Vol. 22.(1994) 34–44
3. Nissen, S.E., Gurley, J.C., Grines, C.L., et al.: Intravascular Ultrasound Assessment of Lumen Size and Wall Morphology in Normal Subjects and Patients with Coronary Artery Disease. Circulation, Vol. 84.(1991) 1087–1099
4. De Sheerder, I., De Man, F., Herregods M.C., et al.: Intravascular Ultrasound versus Angiography for Measurement of Lumen Diameters in Normal and Diseased Coronary Arteries. Am. Heart J., Vol. 127.(1994) 243–251
5. Zamir, M., Chee, H.: Segment Analysis of Human Coronary Arteries. Blood Vessels, Vol. 24.(1987) 76–84
6. Changizi, M.A., Cherniak, C.: Modeling the Large-Scale Geometry of Human Coronary Arteries. Can. J. Physiol. Pharmacol., Vol. 78.(2000) 603–611
7. Murray, C.D.: The Physiological Principle of Minimum Work applied to the Angle of Branching Arteries. J. Gen. Physiol., Vol. 9.(1926) 835–841
8. Hutchins, G.M., Miner, M.M., Boitnott, J.K.: Vessel Caliber and Branch-Angle of Human Coronary Artery Branch-Points. Circulation Research, Vol. 38.(1976) 573–576
9. Bayne, J., et al.: SOFTIMAGE XSI version 2.0, Modeling and Deformations. Avid Technology (2001)
10. Seron, F.J., Garcia, E., Sobreviela, E.J., del Pico, J.: Proyecto Motrico. El Problema de la Generacion de Mallas. Actas, XX Congreso Anual de la Sociedad Espagnola de Ingenieria Biomedica, Zaragoza, Spain (2002) 301–304
11. Pujol, 0., Radeva, P.: Lumen Detection in Ivus Images using Snakes in a Statistical Framework. Actas, XX Congreso Anual de la Sociedad Espagnola de Ingenieria Biomedica, Zaragoza, Spain (2002) 129–132
12. Rotger, D., Radeva, P., Mauri, J., Fernandez-Nofrerias, E.: Internal and External Coronary Vessels Images Registration. Lecture Notes in Artificial Intelligence, Vol. 2504, Springer-Verlag, Castell de la Plana (2002) 408–418

13. Blacker, T.D., et al.: CUBIT Mesh Generation Environment, vol. 1: User's Manual. SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico (1994)
14. Blacker, T.D., Stephenson, M.B.: Paving: A New Approach To Automated Quadrilateral Mesh Generation. International Journal For Numerical Methods in Engineering, Vol. 32.(1991) 811–847
15. Jankovich, S.R., Benzley, S. E., Shepherd, J.F., Mitchell, S.A.: The Graft Tool: An All-Hexahedral Transition Algorithm for Creating a Multi-Directional Swept Volume Mesh. Proceedings, 8th International Meshing Roundtable, South Lake Tahoe, CA, U.S.A. (1999) 387–392

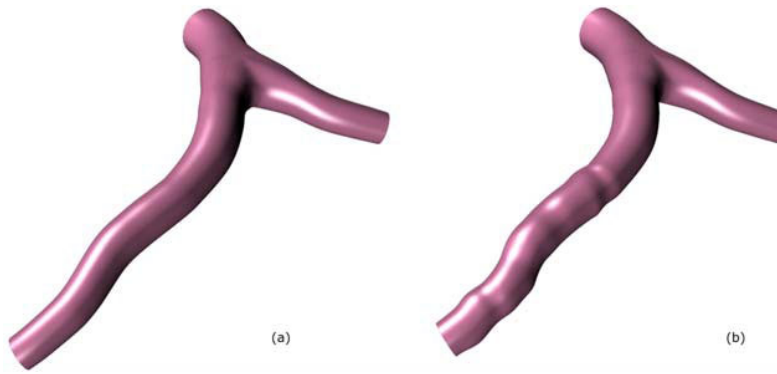


Fig. 4. Geometric models of the human left coronary artery bifurcation.(a) The idealized geometric model. (b) The model with the LAD vessel segment developed by means of angiograms and IVUS.

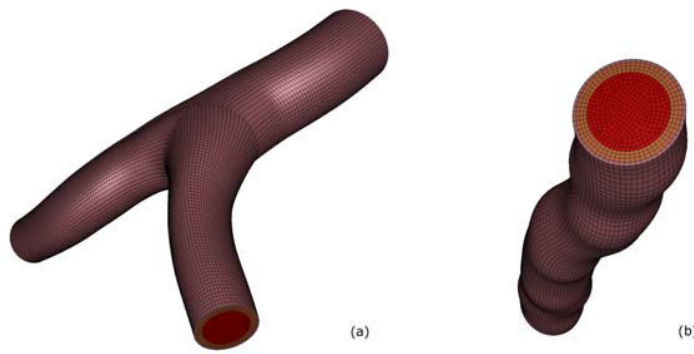


Fig. 5. Five meshes of the geometric models. (a) Meshes of the bifurcation. (b) Meshes of the LAD vessel segment.

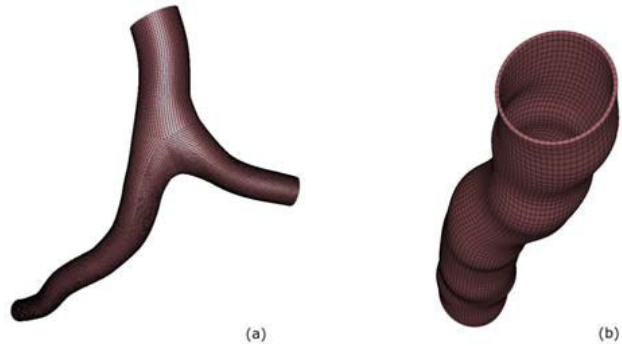


Fig. 6. The meshes of the adventitia layer. (a) Meshes of the idealized model. (b) Meshes of the LAD vessel segment.

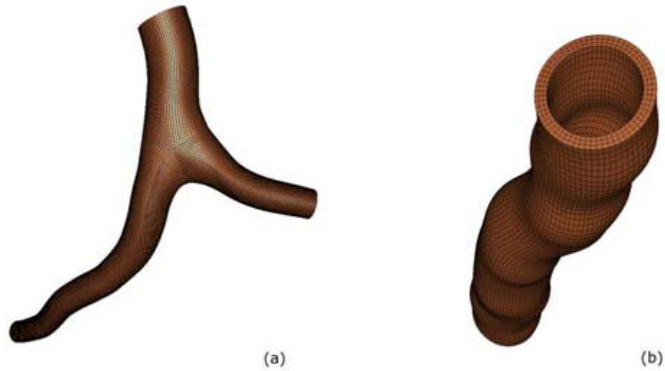


Fig. 7. The meshes of the media layer. (a) Meshes of the idealized model. (b) Meshes of the LAD vessel segment.

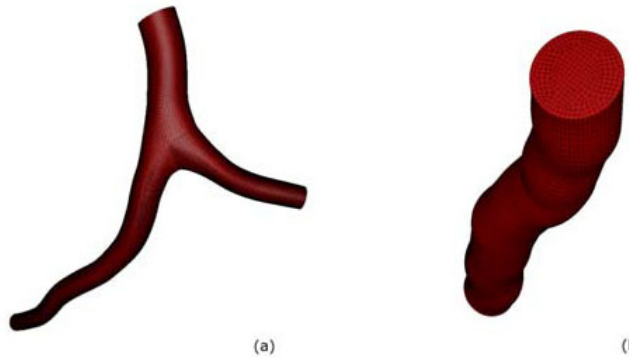


Fig. 8. The meshes of the lumen. (a) Meshes of the idealized model. (b) Meshes of the LAD vessel segment.

Analysis Tool for Cataloguing Textile and Tile Pattern Designs

Margarita Valor, Francisco Albert, José María Gomis, and Manuel Contero

DEGI, Universidad Politécnica de Valencia, Camino de Vera s/n,
46022 Valencia, Spain
{mvalor, fraalgi1, jmgomis, mcontero}@degi.upv.es

Abstract. This paper presents an analysis tool, part of an integrated management system for pattern design in the textile and tile industries, that provides automatic cataloguing capabilities based on the application of the scientific theory of symmetry groups. To do this, a process of analysis is performed which starts from an initial digitized image of the decorative element, which in turn is subjected to a number of segmentation and labelling operators that allow to detect the objects present in the image. These objects are vectorized, compared, and their isometries obtained; subsequently they are grouped and the isometries of the groups of objects detected. Finally, a composition analysis is carried out that, on the basis of the repetitions and symmetry axes existing in the design, provides the fundamental parallelogram and the plane symmetry group. This paper summarizes the results obtained from processing 95 pattern designs using the analysis tool developed by the authors.

1 Background

It is a common practice in the design departments of the textile and tile industries to use old design patterns to create new ones. These companies often keep samples of fabrics and tiles –in some cases, more than 100 years old– which constitute a valuable source for pattern design. Some European regions have a rich tradition in these industries, where specialized museums and design institutions store a great historical patrimony.

To effectively use these resources, it is necessary to digitize these funds and make this information accessible by means of graphic design applications. In this context, we propose to use the symmetry group theory to provide a framework for both analysis and synthesis of new pattern designs.

Applying the symmetry group theory for the analysis of textile and tile pattern designs brings about three different but related issues: symmetry detection, recognition of similar geometrical shapes, and generation of symmetry groups. The detection of symmetries is an issue that has been studied since the 70s, when artificial vision started to develop. Following a chronological order and taking into account their relevance, it is worth mentioning the following works as the basis of automatic symmetry detection: [1], [2], [3], [4]. All these works detect and determine rotational and

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 569-578, 2003.
© Springer-Verlag Berlin Heidelberg 2003

reflectional symmetries on a separate plane geometrical form. If we consider that the design patterns and tile designs respond to plane images in which there is a regular and systematic repetition of geometrical shapes, the knowledge of the isometries of the different geometrical shapes forming the image is an important and necessary data but not decisive to determine the isometries of the complete design. Anyway, the algorithms developed in the works above mentioned may serve as the basis to develop a method for determining the plane symmetry group from the analyzed design, i.e., for determining which isometries characterize the design pattern.

Other issues related to symmetry detection are: orientation of a geometrical shape and recognition of similar geometrical shapes. These aspects are necessary in our approach. On one hand, to permit us to distinguish similar geometrical shapes in a design, and, on the other hand, to determine the different orientations of a geometrical shape in the design [5], [6], [7].

The last important issue related to our approach is the symmetry group theory. Some interesting references can be found in [8], [9], [10], [11], [21]. Based on these references, this work uses recent developments on symmetry detection and geometrical form recognition to analyze and synthesize more complex forms like design patterns or tile designs. The knowledge of the relationships between shapes allows to carry out a geometrical shape reconstruction process.

Table 1. Abbreviations and symbols used in text

Psg	point symmetry group	FP	fundamental parallelogram
PSG	plane symmetry group	P	skewed parallelogram
RE	reflection axes	RE	rectangle.
RAD	reflection axes with displacement	ROE	rhombus formed by two equilateral triangles
RC-X	rotation centre of X-order	RO	rhombus
S	Square		
ER	Dash-point line	CDR-3	Equilateral triangle
ERD	Broken line	CDR-4	Square
CDR-2	Circle or rhombus	CDR-6	Hexagon

2 Design Patterns and Tile Designs: Identification of PSG

Design patterns and tile designs have in common to be the result of a systematic repetition of one given geometrical shape. However, they present some inherent characteristics with respect to the repeated shape [11]. In the case of a design pattern, the repeated geometrical shape does not involve any constraint with respect to its contour shape, since the result is a set of independent geometrical forms more or less close to each other. In the case of tile designs, the repeated form necessarily requires a given shape so as not to allow gaps or overlapping between the different geometrical forms. "Geometrical Form" here means what is perceived or seen. One geometrical form is determined by its visual elements: shape, size, colour and texture.

In spite of these formal differences between design patterns and tile designs, their classification in terms of compositive syntax, is similar and in accordance with the symmetry group theory. The importance of this theory lies on the fact that all design patterns and tile designs can be classified according to the PSG to which they belong.

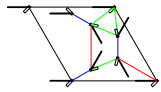
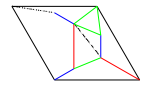
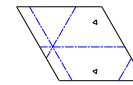
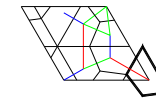
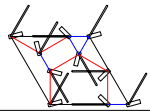
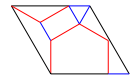
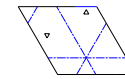
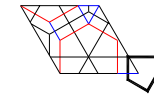
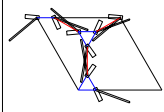
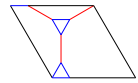
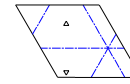
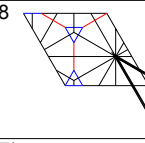
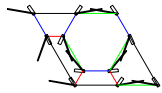
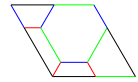
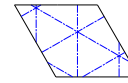
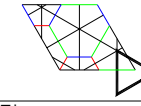
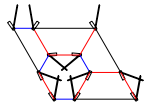
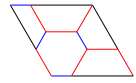
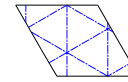
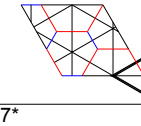
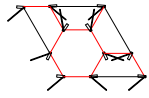
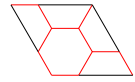
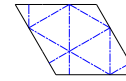
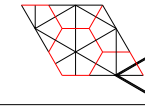
The specific aspects to identify one PSG are [12]: FP type, existing isometries, geometrical properties and minimal region. These specific aspects allow to determine the similarities and differences among the PSG, and have been used to identify the PGS in the cataloguing tool proposed in this paper. In addition, the classification proposed in [8] has been used for periodical patterns (PP), design point patterns (DPP) and isohedral tile designs (IH). With these classifications, cataloguing is limited to isohedral tile designs and periodical tile designs. Two tables have been developed, one table when the group is cyclic and another table when the group is dihedral. In both tables, the first data to know is the psg type of the geometrical form repeated in the design. Table 2 illustrates the cyclic group cataloguing.

The input data of these tables are the following: psg of the repeated geometrical form and number and type (direct and/or reflected) of the geometrical form in the design. With these two data it is possible to obtain, in some cases, the PSG of a design pattern or tile design, like for example, when the psg is C1 with one direct aspect (1D), which is then catalogued with a PSG of type P1.

Table 2. Cataloguing of a pattern design formed by cyclic psg geometrical forms

psg	Aspects	PSG	Periodic Pattern	PF type	Associated isohedral tiling	Associated point patterns
C1	1D	P1	PP1	P-RE-RO-C-ROE	IH1,41	DPP8,16,20,41,51
	2D	P2	PP7	P-RE-RO-C-ROE	IH4,23,46,47,84	DPP7,8,13,15,16,19, 20,41,50
	3D	P3	PP21	ROE	IH7,33	DPP21,25,28,49
	4D	P4	PP30	C	IH28,55,79	DPP30,35,38,39,41
	6D	P6	PP42	ROE	IH21,31,39,88	DPP42,47,48a,48b,50
	1D1R	PG	PP2	RE-C	IH2,3,43,44	DPP13,20,51
		PM	PP3	RE-C	IH42	DPP15,16,41
		CM	PP5	RO-C-ROE	IH22,45,83	DPP16,19,41,50
	2D2R	PGG	PP9	RE-C	IH5,6,25,27,51,52,5 3,86	DPP9,13,16,19,20,41, 50,51
		PMG	PP10	RE-C	IH9,59	DPP11,15,19,41,50
		PMM	PP14	RE-C	IH48*	DPP14,15,16,39,41
		CMM	PP17	C-RO	IH54,78	DPP15,16,17,37,38,41
	3D3R	P31M	PP23	ROE	IH30,38	DPP23,48a,48b
		P3M1	PP27	ROE	IH87*	DPP27,47,50
	4D4R	P4G	PP33	C	IH56,81	DPP33,38,39,41
		P4M	PP37	RE-C	IH80*	DPP37,38
	6D6R	P6M	PP46	ROE	IH77	DPP46
C2	1D	P2	PP8	P-RE-RO-C-ROE	IH8,57	DPP8,16,20,41,51
	2D	P4	PP31	C	IH61	DPP31
	3D	P6	PP43	ROE	IH34	DPP49
	1D1R	PGG	PP10	RE-C	IH9,59	DPP20,41,51
		PMG	PP12	RE-C	IH58	DPP16,41
		CMM	PP18	RO-C-ROE	IH60*	DPP16,41
C3	1D	P3	PP22	ROE	IH10	DPP51
	2D	P6	PP44	ROE	IH90	DPP50
	1D1R	P31M	PP24	ROE	IH89*	DPP50
C4	1D	P4	PP32	C	IH62	DPP41
	1D1R	P4G	PP34	C	IH63*	DPP41
C6	1D	P6	PP45	ROE	IH11	DPP45

Table 3. Classification for C1/3D3R

FP type	forms inside	forms on the side	PP	DPP	parameters	relative orientations	PSG	IH
ROE	5	0	23 	23 	3	NO	P31M 	30 
ROE	4	2	23 	48a 	2	NO	P31M 	30 
ROE	4	2	23 	48b 	2	NO	P31M 	38 
ROE	3	4	27 	27 	3	NO	P3M1 	87* 
ROE	3	4	27 	47 	2	NO	P3M1 	87* 
ROE	3	4	27 	50 	1	NO	P3M1 	87* 

However, there are other cases in which these data are not enough to determine the PSG, as it happens for example when the psg is C1 with one direct aspect and one reflected aspect (1D 1R), which generates a PSG of type PG, PM and CM as possible solutions. To solve this problem, it is necessary to check other data such as: FP type, number of forms located both inside and on the sides of the FP, number of different parameters or distances between geometrical forms, and the relative orientations between the isometries of the geometrical form and the FP. Each one of these data, in the order indicated, eliminates a certain number of groups from all possible groups. To check these data, a number of tables have been developed, ordered in function of the psg type of the repeated geometrical form and the number and type of aspects with which the design is represented [13]. Table 3 illustrates this, showing the classification of design patterns and tile designs integrated by geometrical forms with psg C1, three direct aspects (3D) and three reflected aspects (3R).

In the example shown in Table 3, the type of fundamental parallelogram is not determinant, since it is always ROE. With respect to the number of geometrical forms inside and on the sides of the FP, there are 3 possible combinations: (5,0), (4,2) and (3,4). In the first (5,0) and second case (4,2) the designs are catalogued as P31M while for the third case (3,4) the cataloguing is P3M1. In this example, it is necessary to know neither the relative orientations between the isometries of the geometrical form and the FP nor the number of parameters. These tables also show the type of minimal region, corresponding to the tile of the related isohedral tile design.

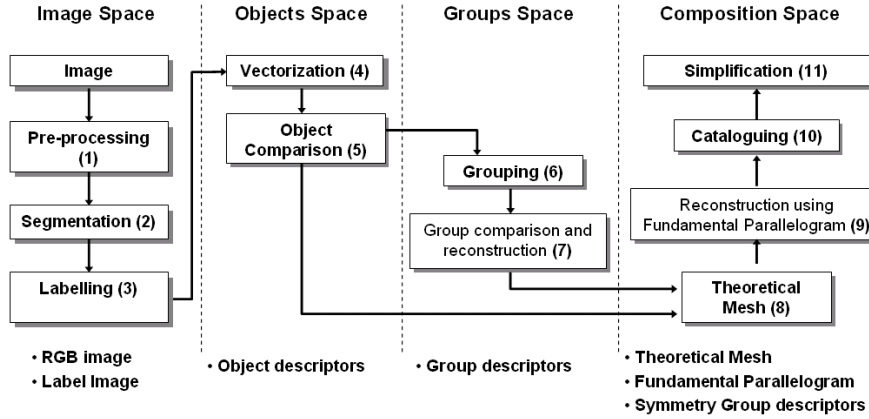


Fig. 1. Working spaces and analysis operators

3 Implementation Process

The Analysis tool executes a number of Operators which act on the input data (bit maps or vector image) and generates new output data. The input data are retrieved from the Acquisition database and the output data are stored in the Design database. Figure 1 shows the organization of the operators and their classification into different stages depending on the spatial representation used. Figure 2 (left) shows the image of a ceramic design element. This design element will be used as an example in the following paragraphs to show the different working spaces and operators.

3.1 Image Space

In the Image Space, we work with bitmap images (figure 2 - centre). As it is the first stage in the process, the input data are the images obtained from the acquisition database. The output is again an image but each region (object) has been properly labelled with an index (the value of each pixel) that differentiates it from the other regions and from the background.

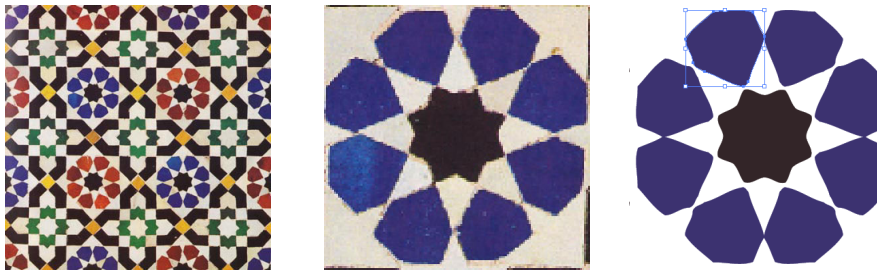


Fig. 2. Illustrating image of a historical tile design (left), detail (centre) and vectorization (right)

There are several operators, although they follow the same scheme:

- Smoothing: back noise removal with low-pass filters like mean or median.
- Segmentation [14]: decomposition of the image into regions identified by colour using clustering algorithms like Mean-Shift or K-Means.
- Labelling: allocation of an index to each differentiated region

3.2 Object Space

Taking the labelled image as input, in this working space a vector data structure is generated formed by a list of objects –which will constitute the output data of the object space–, each one of which contains a number of properties (colour, area, etc) and a list of contours (an external one and any number –zero included– of internal contours) that delimit the object's region. The contours are formed by a sequence of Bezier curves arranged cyclically (the last curve is followed by the first node).

Figure 2 (right) shows the vectorized objects found in the segmentation of figure 2 (centre). Within this working space there are two clearly differentiated phases, corresponding to two operators:

- *Vectorization* [15]: it defines the contours [16] that delimit the object through a set of Bezier curves by means of a two-step process: first, getting the border pixel sequence with a contour retrieval algorithm, and then, breaking the point sequence into subsequences that are approximated by Bezier curves using a least-square method. This representation is more manageable and compact
- *Comparison of objects* [17]: the comparison is done at two levels: first, simple parameters (area, perimeter, etc) and when they are similar, a more detailed comparison of their shape is performed through the object's signature [16] (using module and angle of the radiovector that goes from the mass centre of the object to the external contour). This comparison gives us the transformation that relates two similar objects, expressed with a reflection with respect to the horizontal axis (if necessary), one rotation and one displacement. By comparing one object with itself, we obtain its circular or reflected symmetry axes.

Both the comparison data between objects and the isometries are stored in each object, thus enriching the data structure.

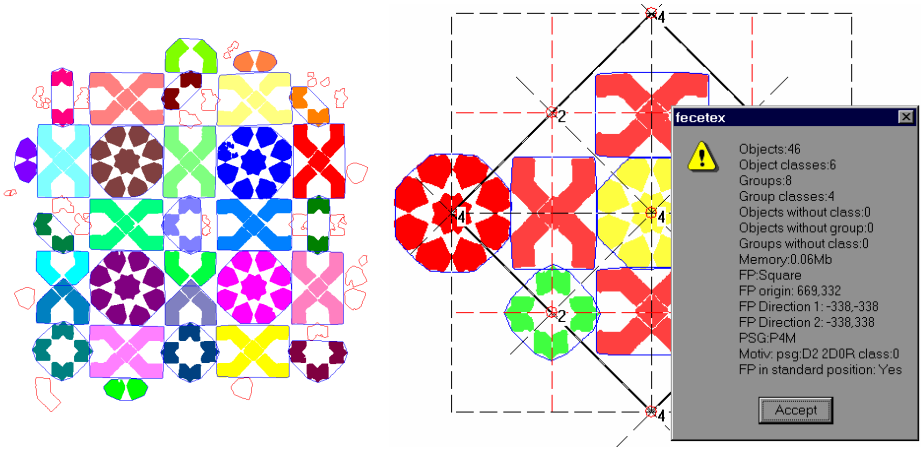


Fig. 3. Group detection (left) and cataloguing example with FP content (right)

3.3 Group Space

Taking as input data the vector data structure with the list of objects, we can generate a list of groups, in which each group is formed by a number of objects related by perceptual criteria and a contour (minimal convex polygon that includes such objects). Now the output data will consist of the two lists, objects and groups.

Figure 3 (left) presents with the same colour those objects belonging to the same group and surrounded by a contour. As grouping criterion, we have taken only the contact segments. This working space is similar to the object space, since we obtain first the working units (objects or groups) and then we compare them, although the procedures used are different:

- *Grouping* [18]: The grouping of the related objects is performed using perceptual criteria (inclusion, contact, cocircularity, colinearity, symmetry, overlapping).
- *Comparison and uniformization of groups*: the comparison is done at two levels, first it is checked that the groups contain a certain number of similar objects, and then the transformations that relate the similar objects between the two groups are compared (displacements, rotations or symmetries with displacement). The presence of a predominant transformation indicates that the groups are formed by the same objects and that these are equally distributed in the group they belong to. Such groups are considered equal and are uniformized. The uniformization process is done by replacing objects in one group by other objects which occupy the same position in another group, so that all the groups are formed exactly by the same objects with the same distribution

As it happened with the objects, the comparison data between groups and their isometries are stored in each group of the list.

3.4 Composition Space

In this space, analyzing the vector data structure with the lists of objects and groups, we obtain the necessary information on the design's structure, thus obtaining: the two vectors that define the sides of the FP and the PSG (symmetry axes and rotation centres). This structural information together with the lists of objects and groups constitutes the Analysis output data. Figure 3 (right) presents with the same colour the similar groups; the design's structure has also been drawn: fundamental parallelogram (in black), symmetry axes (broken line, axes without displacement in black and axes with displacement in red) and the rotation centres (red circumference and order number). Only the groups contained in the FP have been left, but without dividing any of them. The fundamental parallelogram is a Square, and the resulting Cataloguing is P4M.

The operators of the Composition space are the following:

- *Fundamental parallelogram*: from similar objects or groups where the transformation is only one displacement, we have to find two unique vectors which allow us to displace them with linear combinations (these two vectors act as the basis of the vector space defined by the positions of similar segments). When there are several bases, we will choose the one with an n-times area, since the repeatability of the design is that of the less frequently repeated elements. Such vectors form two sides of the FP, i.e., the smallest part of the design that by replicating and displacing it is able to generate the whole design.
- *Reconstruction*: the FP represents the repeatability, which allows us to check if all overlapping groups when displaced in the directions given by the two vectors of the parallelogram, are equal. If they are not, we will remove some and will replicate others to uniformize the design.
- *Cataloguing*: by considering the geometry of the FP, the symmetry axes and the rotation centres of the design, we obtain its cataloguing in accordance with the symmetry group theory.
- *Reconstruction by the PSG*: first, the design is simplified, since the content of the FP and its cataloguing will be sufficient to define the whole design; therefore we will suppress all redundant information without decomposing any objects or groups. Then, we will check that all the objects or groups related by symmetry axes or rotation centres are equal and following the correct orientation; otherwise, we will choose the best alternatives (those which fulfil the symmetry criterion) and the incorrect objects and groups will be replaced.

4 Results and Conclusions

To validate the methodology and tool developed by the authors we have successfully used 93 synthetic designs from Grünbaum and Shephard [8] and 45 from Kaplan and Salesin [20] covering all the cataloguing cases. As real test cases we have processed digital images from the Institute of Ceramic Promotion of the Council of Castellón (Spain), historic catalogues of digitized tile designs, digital images from books of design patterns and tile design (see Fig. 4) [19] and digitized textile images.

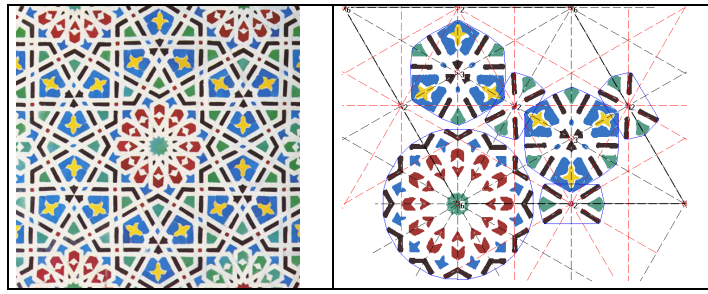


Fig. 4. Cataloguing of a P6M design element from a tile design in [19]

The performed tests show that this tool satisfactorily reaches its objectives with most of the processed images. In some cases the user must tune the system's parameters to improve the cataloguing operation. From 95 textile and ceramic images we have obtained the following results:

– 64% correct cataloguing with minor tuning of the system's parameters
– 33% correct cataloguing with major tuning of the system's parameters
– 3% not possible cataloguing

Typical processing time for analysis is presented in Table 4, where it is shown that processing time is related with design complexity (number of objects in the design). Main detected problems come from old design photographs (which presented irregularities in the objects and colours) and damaged tiles (dirt or cracks). However the operators that introduce reconstruction (comparison and uniformization of groups using their FP), can solve some of the problems above mentioned.

Table 4. Processing time for several examples on a Pentium III 450 MHz computer

Image type	Size	Objects	Processing time
Ceramic	1000x989	16	27"
Ceramic	1188x1188	206	1' 08"
Ceramic	2772x1608	556	6' 31"
Textile	1000x1728	48	52"
Textile	2032x1290	76	1' 50"
Textile	2035x1532	717	7' 10"

The tests show that the images to process should have the correct resolution to allow an object to be represented with a sufficient number of pixels, so that pixel size can be neglected, and the presence or not of some pixels in different representations of the same object does not become decisive for image processing. It is also necessary that the image contain enough repetitions to obtain the FP. These two conditions – greater resolution and greater extension– require more memory and processing speed, so that we can get image sizes difficult to process; therefore it may be necessary for example not to take into account the smallest objects and work only with those objects which are big enough.

Acknowledgments

This work has been supported by the Spanish Science and Technology Ministry and the European Union (Project DPI2001-2713).

References

1. Atallah, J.M.: On Symmetry Detection. IEEE Trans. On Comp. **34** (7) (1985) 663-666
2. Marola, G.: On the Detection of the Axes of Symmetry of Symmetric and Almost Symmetric Planar Images. IEEE Transactions on P.A.M.I. **11** (1) (1989) 104-108
3. Lladós, J., Horst, B., Martí, E.: Finding Rotational Symmetries by Cyclic String Matching. Pattern Recognition Letters **18** (14) (1997) 1435-1442
4. Shen, D., Ip, H. Cheung, K., Teoh, E.: Symmetry Detection by Generalized Complex (GC) Moments: a Close-Form Solution. IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (5) (1999), 466- 476
5. Lin, J.C., Sheng-Lin, C., Wen-Hsiang T.: Detection of Rotationally Symmetric Shape Orientations by Fold Invariant Shape Specific Points. Pattern Rec. **25** (5) (1992) 473-482
6. Zabrodsky, H., Peleg, S., Avnir, D.: Symmetry as a Continuous Feature. IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (12) (1995) 1154-1166
7. Stein, F.: Perceptual Grouping for Generic Recognition. Int. Journal of Computer Vision **20** (1996) 59-80
8. Grünbaum, B., Shephard, G.C.: Tilings and Patterns. W. H. Freeman, New York (1987)
9. Martin, G.E.: Transformation Geometry. An Introduction to Symmetry. Springer-Verlag, New York (1982)
10. Schattschneider, D.: The Plane Symmetry Groups: Their Recognition and Notation. The American Mathematical Monthly **85** (1978) 439-450
11. Shubnikov, A.V., Koptsik, V.A.: Symmetry in Science and Art. Plenum Press, NY, (1974)
12. Valor, M., Albert, F., Gomis, J.M., Contero, M.: Identificación y Aspectos Específicos de Patrones y Embaldosados a Partir de la Teoría de Grupos de Simetría del Plano. Proceedings of the XIV Int. Congress of Graphics Engineering, Santander, Spain, (2002) 32-43
13. Valor, M.: Diseño de Herramientas Gráficas para la Catalogación de Revestimientos Cerámicos. PhD Thesis (2002) Polytechnic University of Valencia (Spain)
14. Comaniciu, D., Meer, P.: Robust Analysis of Feature Spaces: Color Image Segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA, (1997) 750-755
15. Albert, F., Gomis, J.M., Valor, M., Valiente, J.M., Carretero, M.: Análisis Estructural de Motivos Decorativos en Diseño Textil. Proceedings of the XIII International Congress of Graphics Engineering, Badajoz, Spain, (2001) 35-36
16. González, R.C., Wintz, P.: Digital Image Processing. Addison-Wesley (1987)
17. Albert, F., Gomis, J.M., Carretero, M., Valiente, J.M.: Object Comparison in the Structural Analysis of Textile Design Patterns. Proc. of the XII ADM International Conference on Design Tools and Methods in Industrial Engineering, Rimini, Italy, (2001) 77-78
18. Köhler, W., Koffka, K., Sander, F.: Psicología de la Forma, Ed. Paidós, Barcelona (1969)
19. Paccard A.: Le Maroc et L´Artisanat Traditionel Islamique dans L´Architecture. Éditions Atelier 74, Bonlieu, (1983)
20. Kaplan, C., Salesin, D.: Escherization. SIGGRAPH 2000 Conference Proc. 499-510
21. Washburn, D.K., Crowe, D.W.: Symmetries of Culture: Theory and Practice of Plane Pattern Analysis. University of Washington Press, Seattle (1988)

Urban Site Modeling from LiDAR

Suya You, Jinhui Hu, Ulrich Neumann, and Pamela Fox

Integrated Media Systems Center
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781

Abstract. This paper presents a complete modeling system that extracts complex building structures with irregular shapes and surfaces. Our modeling approach is based on the use of airborne LiDAR which offers a fast and effective way to acquire models for a large urban environment. To verify and refine the reconstructed ragged model, we present a primitive-based model refinement approach that requires minor user assistance. Given the limited user input, the system automatically segments the building boundary, does the model refinement, and assembles the complete building model. By adapting a set of appropriate geometric primitives and fitting strategies, the system can model a range of complex buildings with irregular shapes. We demonstrate this system's ability to model a variety of complex buildings rapidly and accurately from LiDAR data of the entire USC campus.

1 Introduction

While current sensing and modeling technologies offer many methods suitable for modeling a single or a small number of objects, an accurate large-scale urban model still remains costly and difficult to produce, requiring enormous effort, skill, and time, which results in painfully slow evolution of such visual databases [1]. This problem is the main impetus for our work. One of our objectives is the rapid and reliable creation of 3D models of large-scale urban environments such as city models. Our approach is based on the use of airborne LiDAR (Light Detection and Ranging), which offers a fast and effective way to acquire models for a large environment. In urban areas, LiDAR also provides useful approximations for urban features and buildings. However, sample-rate limitations and measurement noise obscures small details and occlusions from vegetation and overhangs lead to the data voids in many areas. So, another objective of our work is to refine the acquired models to be geometrically accurate within all local details, rather than in a global average sense.

Over years, a wealth of research has appeared to address the urban site modeling problem from photogrammetry or from laser sensing data. For example, Elaksher et al. [6] proposed a system for the reconstruction of planar rooftop building wire-frames from LIDAR data. Coorg and Teller [5] constructed a large set of 3D building models by using spherical mosaics produced from accurately calibrated ground view cameras. The method can be applied to model a relatively large site area, but it is limited to simple shape buildings and does not capture the roof structure. Lee and Nevatia [2] presented a method of integrating aerial and ground view images for urban site modeling. Zhao [3] and Seresht [4] developed methods for extracting buildings by

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 579-588, 2003.
© Springer-Verlag Berlin Heidelberg 2003

combining color aerial images with DEM (Digital Elevation Model). CYBERCITY [13] is a commercial software package for structuring 3D objects. There are also other similar approaches or systems that use single sensor data or integrated multiple sensors, but these implementations are limited to a set of simple building elements or combinations.

In this paper we present a complete modeling system (Fig.1) that can extract a variety of complex building structures with irregular shapes and surfaces. Our approach is based on the use of airborne LiDAR data. To verify and refine the reconstructed geometry model, we present a primitive-based modeling approach that requires only minor user assistance. We have used the system to model a variety of complex buildings from LiDAR data of the entire USC campus. The results indicate that our system is suitable for producing large-scale urban models at modest cost.

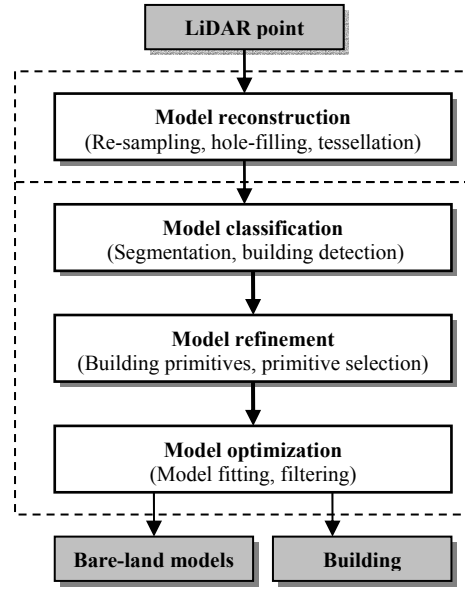


Fig. 1. Algorithmic structure and work flow of our modeling system

2 Model Reconstruction from LiDAR Data

A LiDAR sensor system permits an aircraft flyover to quickly collect a height field for a large environment with an accuracy of centimeters in height and sub-meter in ground position (typical) [14]. Multiple passes of the aircraft are merged to ensure good coverage. Due to its advantages as an active technique for reliable 3D determination, LiDAR has become a rather important information source for generating high quality 3D digital surface models.

With the cooperation with Airborne1 Inc. [14], we acquired the LiDAR model of the entire USC campus and surrounding University Park area. The end result is a cloud of 3D point samples registered to a world coordinate system (ATM - Airborne Topographics Mapper). We project and re-sample the points onto a regular grid (user defined resolution) to produce a height field or range image suitable for tessellation.

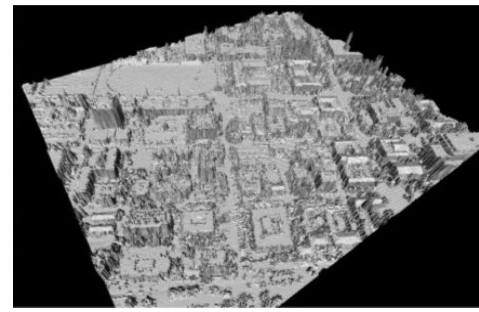


Fig.2. Reconstructed 3D mesh model of entire USC campus

Due to laser occlusions and the nature of the material being scanned, there are lots of holes in the range image without height measurements. We perform the hole-filling operation by directly interpolating the depth values in the range image in order to preserve the geometric topology of model. To preserve the edge information, we utilize an adaptive-weighting neighborhood interpolation. The interpolation weights are determined by an inverse function of the distance between the neighbor points and the point to be interpolated. The window size of interpolation is adaptive to the surface-hole size. When the surface-hole size is only a few points, a small window is

used that contains the close neighbors for weighing interpolation. For the large holes, the window size is increased to ensure sufficient points for interpolation.

Triangle meshes are used as the 3D geometric representation as they are easily converted to other geometric representations; many level-of-detail techniques use triangle meshes; photometric information is easily added with texture projections; and graphics hardware supports fast rendering of triangle meshes [11]. We have tested several tessellation methods including the closest-neighbor triangulation and Delaunay triangulation, and found that the Delaunay triangulation appears to be superior to preserve the topology and connectivity information of the original data.

The whole processing of model reconstruction is fully automatic. The system allows a user to select any portion of the input data to reconstruct a 3D mesh model under defined re-sample resolution. Once the parameters of data size and re-sample resolution are set, the system automatically performs the steps to process the 3D point cloud and outputs the reconstructed 3D mesh model in VRML format. Fig. 2 shows the reconstructed model of the entire USC campus and surrounding University Park area at original sample resolution.

3 Urban Model Classification

To extract the buildings from the reconstructed 3D mesh model, the points of the mesh model have to be classified according to if they belong to terrain, building, or something else. In our system, we classify the original LiDAR model into two categories: buildings, and bare-land. The building subset is a collection of the building models represented as the parametric forms, while the bare-land subset is the reconstructed 3D mesh model with the buildings removed.

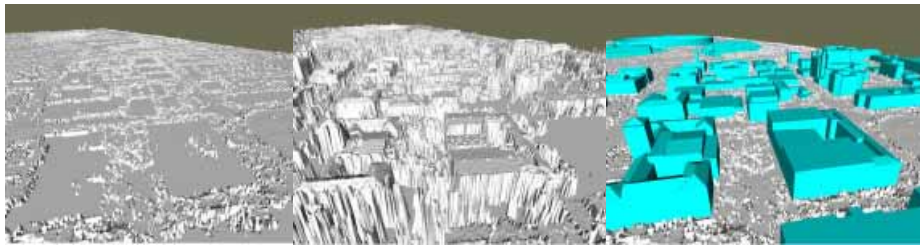


Fig. 3. Classifying the LiDAR model as two categories: (left) bare-land, and (middle) buildings. The extracted buildings are very rough that there are many artifacts remained around buildings. The initial classification has to be refined in order to remove the undesired areas to improve its utility and visualization value (right)

The classification approach is conducted based on an obvious fact: the objects, which have the height above a certain value, must be either vegetation or buildings. So, by applying a height threshold to the reconstructed 3D mesh data, we can create an approximate building mask. The mask is applied to filter all the mesh points, and only those masked points are extracted as building points. Fig. 3 illustrates the results of applying the approach to classify the USC campus mesh mode as the bare-land (Fig. 3 left) and the building areas (Fig.3 middle). As we can see, the extracted building subset is very rough that there are many artifacts remained around buildings. The initial classification has to be further refined in order to remove the undesired areas. Our strategy is to use an accurate geometry model to fit the building mesh data to produce a constrained CG building model. Once we obtain the refined building models with accurate geometry, we can easily remove those artifacts from the initial classification by combining the geometry shape cues. Fig. 3 (right) illustrates the accurate classification of the bare-land and the buildings embedded in the land.

4 Model Optimization and Refinement

Our model refinement is a primitive-based approach. We divide a complex building into several basic building primitives and model them using a parametric representation. As the models from constructive solid geometry allow the composition of complex models from basic primitives that are represented as parametric models, our approach is quite general. Also, as the type of primitive is not limited, may contain objects with curved surfaces, so the flexibility of model combinations is very high, hence we can model a range of complex buildings with irregular shapes and surfaces by combining appropriate geometry primitives and fitting strategies.

4.1 Building Primitives

Based on the shape of a building roof (flat-roof, slope-roof, dome-roof, gable-roof, etc.), we classify a building section into one of several groups, and for each group we define a set of appropriate geometry primitives, including the standard CG primitives such as a plane, slope, cube, polyhedron, wedge, cylinder, and sphere, etc., and high-order surface primitives such as ellipsoids, and superquadrics. These geometry primitives are the basic units used for building construction. They also can be combined with each other to form more complex new primitives. Fig. 4 illustrates a set of building primitives and their relationships defined for modeling a complex building.

A high-order surface primitive is useful to model irregular shapes and surfaces, such as classical dome-roof buildings and a coliseum or arena. Superquadrics are a family of parametric shapes that are mathematically defined as an extension of non-linear general quadric surfaces, and have the capability of describing a wide

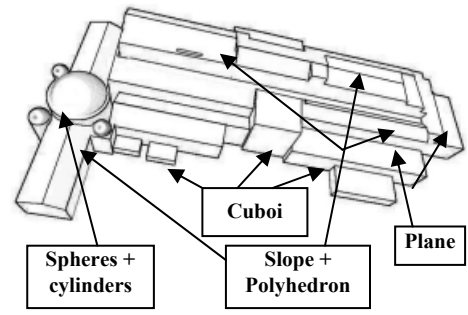


Fig. 4. Geometry primitives used for representing a building model

variety of irregular shapes with a small number of parameters [9]. In our work, we use them as a general form to describe all the nonlinear high-order primitives, as defined in (1).

$$r(\eta,\omega)=\begin{bmatrix} a_1\cos^{\varepsilon_1}\eta\cos^{\varepsilon_2}\omega \\ a_2\cos^{\varepsilon_1}\eta\sin^{\varepsilon_2}\omega \\ a_3\sin^{\varepsilon_1}\eta \end{bmatrix} \quad \begin{matrix} -\pi/2\le\eta\le\pi/2 \\ -\pi\le\omega<\pi \end{matrix} \tag{1}$$

where ε_1 and ε_2 are the deformation parameters that control the shape of primitive, and parameters a_1, a_2 and a_3 defines the primitive size in *x, y* and *z* direction, respectively. By selecting different combination of these parameters, superquadric can model a wide variety of irregular shapes, and also many standard CG primitives as well.

Once defined, each building primitive is represented as a parametric model. The parametric model describes the primitive by a small but fixed set of variable parameters. The number of parameters needing to be specified depends on the properties of each primitives and the knowledge assumed for the model fitting. For example, a generic plane in 3D space can be represented as

$$z = ax + by + c \tag{2}$$

which has 3 parameters need to be estimated. However, in the case of slope-roof fitting, we may reduce the parameters from 3 to 2 by setting either parameter *a* or *b* to zero. This is based on the observation that if a building's orientation is nearly parallel to the *x* or *y* axis of a defined world-coordinates, then either parameter *a* or *b* will be close to zero for most buildings, i.e. the roof of a building usually has only one slope along the *x* or *y* direction. We use the term "zero x/y slope" to indicate the constraint for slope-roof fitting. Similar constraints can also be established for other primitives, which are based on our observation and applications. We will details those in the following sections.

4.2 Primitive Selection

As we classify the building sections into several groups, in which appropriate building primitives are defined, we have to provide the system the information to indicate the selected building section and associated type. We term the building section being processed as the "Element of Interest" (EOI) that is a square area roughly bounding the building section. Currently, the EOI information needs to be provided by a user in that only few mouse-clicks are required. Once the user input is provided, the system automatically segments the building borders and surface points, and uses the indicated building primitive to fit the building mesh model. The amount of user interaction depends on the type of building primitives associate with the group. For example, the cube primitive is determined by two points and an orientation. So, this primitive fitting needs only two user mouse-clicks to indicate two diagonal-points on the roof surface. In most cases, 2 or 3 user mouse-clicks are sufficient. The constraints established for primitives also reduce the amount of user interaction.

4.3 Primitive Fitting

The following describes the most commonly used building primitives, including their mathematical parametric representations, and examples of building fitting. The detailed algorithm procedures for each primitive are also described.

Plane Primitives. The flat-roof is a typical roof type of man-made buildings, which can be modeled using the plane-primitive group, including 3D plane, cuboids, polyhedron, and the combined primitives such as hollow-cuboids. They all share the same property that the depth surface can be described by equation (2). A 3D plane primitive is usually determined by two reference points and an orientation. If we align the building's orientation to the global direction that is defined in our working coordinates, we can reduce the specified parameters to 2, i.e. each plane is specified by two diagonal-points.

After the user indicates (mouse clicks) the two reference points, the system automatically estimates all corners of the building roof based on the global direction. The estimated corner points are then used for detecting the roof edges using a depth discontinuity constraint. We proposed an improved 8-neighbors connectivity algorithm to detect building edges. First, we used the geometry connectivity information of Delaunay reconstruction to track the connected edge points. Only those edges that lie along the Delaunay triangulation are accepted as the possible edge points. Second, we utilized a depth filter to constrain the detected edges. The depth filter is applied to all the possible edge points, and only those points having similar depth values as that of the defined reference points are passed as correct edge points. Once the roof borders have been extracted, we parameterize them using least-square fitting, and then the roof corners are refined again based on the fitted roof borders.

Plane depth fitting is performed on all the surface points inside the roof border. The depth discontinuity constraint is used for surface segmentation. We opted not to use surface-normal information due to its sensitivity to noise. Our experiments show that the depth discontinuity constraint performs well for surface segmentation. After segmenting the surface points, the plane least-square fitting is applied to the depth values of those points, and the best fitting is the height of the refined surface.

Slope Primitive. Slope is a special case of the plane with non-zero horizontal or vertical normal direction. Similar to the plane primitive, a sloped roof with rectangular edges is also extracted with two reference points using the plane fitting method. The depth fitting for sloped surfaces, however, is more complex. A 3D plane defined in (2) has 3 parameters to be estimated, where the two parameters *a, b*, represent two slopes in the *x* and *y* directions, and the parameter *c* is an offset. But we can reduce the parameters from 3 to 2 based on the "zero x/y slope" constraint. In the case of a building does not meet the condition, we perform an orientation alignment to orient the building to the reference direction. The least-square method is also used for parameter estimation, using all the surface points inside the detected roof borders.

We observe the fact that most roofs of real buildings usually have two symmetric slopes. To facilitate this structure, we combine the two connected slope primitives to form a new primitive: *roof*. In this case, three reference points (rather than four if we model the two slopes separately) are need for parameter estimations: two on the slope edges, and one on the roof ridge. The surface points of the two symmetric slopes are segmented using the above method. The least-square fitting is performed on the

depth values of the segmented surface points for each of the two slope primitives. The accurate roof ridge is computed based on the intersection of the two modeled slope planes.

Cylinder Primitive. Surface fitting of a generic cylinder is a nonlinear optimization problem. However, we observe the fact that most cylinder primitives in buildings have an axis perpendicular to the ground. Based on this constraint, we can eliminate the rotation parameter from the estimate to simplify the primitive as a vertical cylinder for circle-roofs.

The roof extraction and surface segmentation are similar as the plane case using the depth discontinuity constraint. Two concentric circles are defined for segmentation: the inner circle for roof border detection, and the outer circle for surface point segmentation. Three parameters are required for specifying the concentric circles: one for the circle center and two for the radius. To guarantee there are enough surface points for accurate segmentation and model fitting, the defined circles should cover all the possible surface points on the rooftop.

To achieve an accurate boundary reconstruction from the ragged mesh data, we defined two filters to refine the detected edges: a depth filter constraining the edge points having similar depth values as that of the defined center, and a distance filter constraining the edge points to being inside of the estimated circle. The depth filter is similar as the one we applied for plane primitives, but using the circle center's depth value as a filtering threshold. The distance filtering is a recursive procedure. Using the detected edge points, we first fit them to the circle model to obtain initial estimates of the circle center and radius. We then use these initial estimates to filter the detected edges. Any edge points whose distance to the circle center is less than a threshold will pass the filtering. The distance deviation is used as a filtering threshold. After the distance filtering, the refined edge points are used recursively to estimate a new border parameter.

Sphere Primitive. The dome-roof is a very popular roof type in classical buildings. A simple dome-roof can be modeled as a sphere shape, but more complicated tones may need high-order surfaces to represent them. Similar to the cylinder primitive, the surface of a sphere is also a quadric surface.

To detect the roof border and surface points, two reference values are needed: one for dome-roof center and another one for roof size. To guarantee enough surface points for accurate segmentation and fitting, the defined area should cover all the possible points on the roof surface. Since the section-projection of a sphere is a circle, the methods for sphere-roof detection and surface segmentation are almost the same as those we used for the cylinder primitive, except for not using the depth filtering as we don't have the sphere center in 3D space.

The model fitting is performed on all the segmented spherical surface points. As in the cylinder case, the distance constraint has also been recursively used to achieve an accurate model reconstruction. The sphere primitive can also be combined with other type primitives. The most popular usage is the sphere-cylinder combination. Another use of sphere primitives is their use as a preprocessing step for high-order surface fitting. High-order surface fitting normally is a non-linear problem. Appropriate selection of initial estimates is vital to guarantee a convergence to an optimal solution.

High-order Primitives. The standard CG primitives have limited capability to model complex objects. One of the innovative features of our system is it supports high-order modeling primitives to facilitate irregular building structures. Superquadrics are a family of parametric shapes that are mathematically defined as an extension of non-linear generic quadric surfaces. They have the capability of describing a wide variety of irregular shapes with a small number of parameters. We use the superquadric as a general form to describe all the nonlinear high-order primitives.

As an example of applying the high-order primitives to model irregular building shapes, we describe the steps to model the Los Angeles Arena with an ellipsoid primitive. The ellipsoid is a special case of superquadrics with the deformable parameters $\varepsilon_1 = 1, \quad \varepsilon_2 = 1$.

1. Object segmentation

The region-growing approach [12] is employed to segment the irregular object from its background. Given a seed-point, the algorithm automatically segments the seeded region based on a defined growing rule. In our implementation, the surface normal and depth information are used to supervise the growing procedure.

2. Initial surface fitting

To guarantee a converged optimal solution, an appropriate initial value is required for the Levenberg-Marquardt (LM) algorithm [10]. A sphere primitive fitting is used for system initialization.

3. *High-order surface filling*

Once initialized, the system fits the ellipsoid primitive to the segmented surface points using LM algorithm. In this Arena modeling example, the algorithm needs 606 iterations to converge to the correct solution. Fig. 5 shows the final fitting result. The refined model appears to represent the LiDAR data very well although we have yet to survey the actual building and compare its measures to our refined model dimensions.

5 System Evaluation

We have implemented a prototyped modeling system. Based on our on-hand LiDAR dataset, we have modeled the entire USC campus and surrounding University Park area including the Coliseum, LA Arena, museums, and gardens (Fig. 5). The system has been tested using a range of different building structures. For example, Fig. 6 shows the model of the LA Natural History Museum across the street from USC campus. Note the inclusion of slanted roof segments and domes makes this facility very complicated to model. Our system allows users to create the building models in a few minutes by selecting a few points in the LiDAR data.

Due to lack of actual building measurements, quantitative evaluation of the modeling accuracy is difficult. In our work, we use two strategies to evaluate our modeling system. The first strategy is to verify the reconstructed model dimensions by embedding the model in the original LiDAR data. As the LiDAR data is physically acquired from the real world, it appears to represent the real structures very

well. We have used the strategy to evaluate the reconstructed accuracy with every primitive we proposed in the system.

The second strategy is to use imagery geo-referencing to verify the accuracy of the geometry model. We evaluated our system using several imagery sources including aerial photographs, ground image/video captured with high-resolution digital cameras, and terrain maps. By projecting those images onto the geometry models, we can immediately observe the errors resulting from model reconstruction. This strategy allows us to evaluate the accuracy of very fine building structures.

Acknowledgment

This work was supported by a Multidisciplinary University Research Initiative (MRUI) on "Next Generation 4-D Distributed Modeling and Visualization". We thank the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, for their support and facilities. We thank Airborne1 Inc. for providing us with the USC campus LiDAR data. Our thanks also go to HP, NVidia, Intel, and Microsoft for their supports.

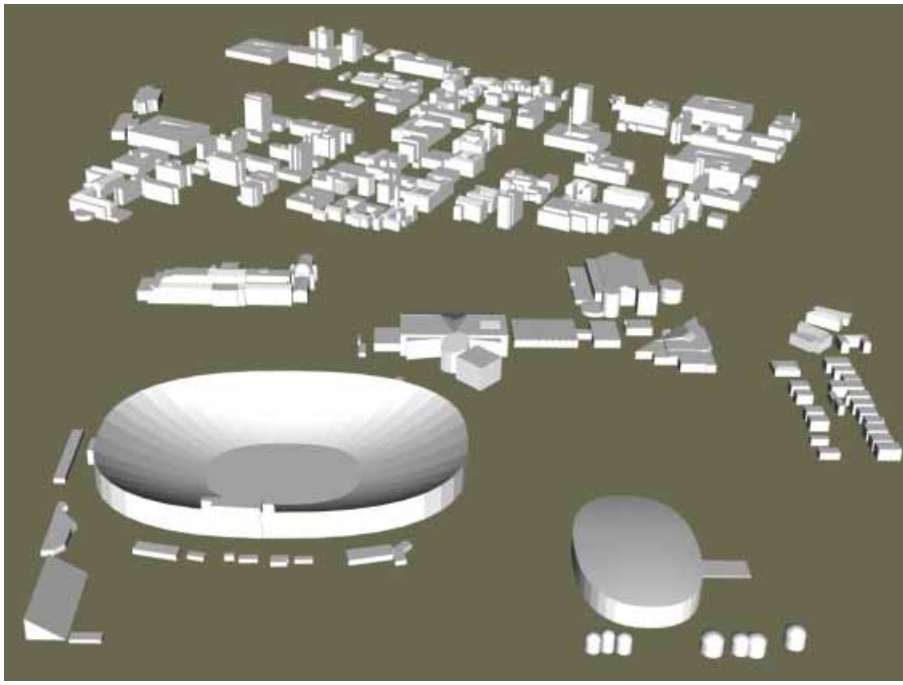


Fig.5. Complete refined models of USC campus and surrounding University Park area. The model was created in two days by author using the proposed system.

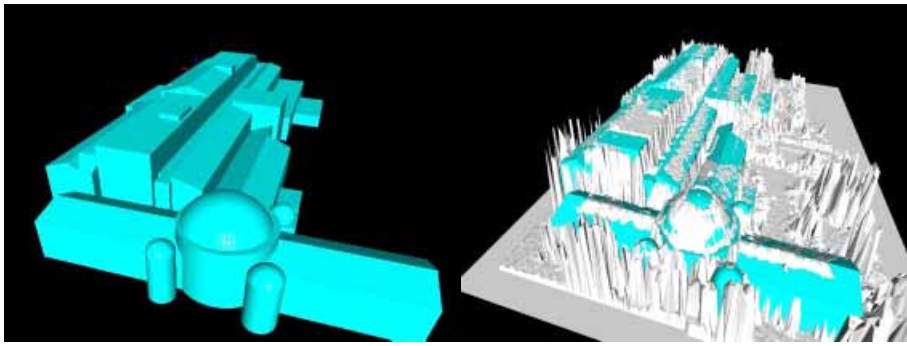


Fig.6. System performance evaluation by embedding the refined model in the original LiDAR data: (left) refined model of LA Natural History Museum, and (right) the refined model is embedded in the original mesh data. The refined model appears to represent the LiDAR data very well.

References

1. W. Ribarsky, T. Wasilewski, and N. Faust, "From Urban Terrain Models to Visible Cities", *IEEE Computer Graphics & Applications*, Vol. 22, No. 4, 2002.
2. S. C. Lee, S. K. Jung and R. Nevatia, "Automatic Integration of Façade Textures into 3D Building Modelings with Projective Geometry Based Line Clustering", *EUROGRAPHIC'02*, 2002.
3. Zhao, B. and Trinder, J., "Integrated Approach Based Automatic Building Extraction", *19th ISPRS Congress*, Book 3B, pp. 1026-1032, Amsterdam.
4. Seresht, M. and Azizi, A., "Automatic Building Recognition from Digital Aerial Images", *19th ISPRS Congress*, Book 3B, pp. 792-798, Amsterdam.
5. Coorg, S. and S. Teller, "Extracting Textured Vertical Facades from Controlled Close-Range Imagery", *CVPR*, 1999, pp. 625-632.
6. A. Elaksher, J. Bethel, "Building Extraction Using Lidar Data", *ASPRS-ACSM Annual Conference and FIG XXII Congress*, April 22-26, 2002.
7. Haala, N., C. Brenner, *"Generation of 3D city models from airborne laser scanning data",* the 3rd EARSEL Workshop on LIDAR Remote Sensing on Land and Sea, Tallinn, 105-112,1997
8. A. Fruh and A. Zakhor, *"3D Model Generation for Cites Using Aerial Photographs and Ground Level Laser Scans",* IEEE Conference on Computer Vision and Pattern Recognition, *2001.*
9. Suya Y., Ulrich N., *Automatic Object Modeling for 3D virtual Environment,* Proceedings of NMBIA, pp.21-26, 1998.
10. William, H., Saul, A., William, T., and Brian, P., *"Numerical Recipes in C", Cambridge University Press, 1992.*
11. Foley, J. D., Van Dam, A., Feiner, S. K. and Hughes, J. F., "Computer Graphics: principles and practice", Addison-Wesley, Reading, Massachusetts, 1990.
12. Richard E.W. and Rafael C.G., "Digital Image Processing", Prentice Hall PTR, 2nd Edition.
13. *http://www.cybercity.tv*
14. *http://www.airborne1.com*

An Automatic Outline Fitting Algorithm for Arabic Characters

M. Sarfraz and M.A. Khan

Department of Information and Computer Science,
King Fahd University of Petroleum and Minerals,
KFUPM Box No. 1510, Dhahran 31261, Saudi Aabia.
sarfraz@ccse.kfupm.edu.sa

Abstract. In this paper, an efficient algorithm for automatic outline capture of digital character images is presented. This method is well suited for non-Roman languages like Arabic, Urdu, Persian, etc. Contemporary word processing systems store shapes of the characters in terms of their outlines, and outlines are expressed as cubic Bezier curves. The process of capturing outlines includes steps detection of boundary, discorvering *Corner Points* and *Break Points* and fitting the curve. The work done in this paper, fully automate the above process and produces optimal results.

1 Introduction

Two fundamental techniques for storing fonts in computer are bitmap and outline [9,10]. Outline representation has many advantages over bitmap such as scaling, shearing, translation, rotation, and clipping. Arabic characters are different from other characters such as English, Latin etc. They are written cursively from right to left and each character has two to four different forms, depending on its position in the word. This paper proposes an algorithm to eliminate the human interaction in obtaining the outline of original digital character. In traditional approaches [5], initially, a hand drawn character is scanned from paper to obtain a gray level image. From this gray-level image, boundary or contour of the character is obtained. Then *Corner Points* of the character are determined by some interactive method or by some automated corner detection algorithm [8,6]. Curve fitting is done by segmenting the contour outline at the *Corner Points*. Normally, the curve fitting methods are based on conics or Bezier cubics [4].

The proposed algorithm improves the results of work by the same authors [2]. The methodology in this paper has many advantages over traditional approaches as it completely automates the whole process and it provides the desired accuracy by reparemeterization and/or by adding some more points, we called these points *Break Points*. The set of *Significant Points* consists of *Corner Points* and *Break Points*. This paper uses Bezier cubics in the segments.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 589–598, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 The Algorithm

Our proposed algorithm of automatic approximation of boundary of digital character images consist of following steps.

1. Getting Digitized Image
2. Extracting Boundary (contour)
3. Detecting *Corner Points*
4. Filtering Noise
5. Fitting Parametric Cubic Bezier (iterative step)
6. Reparameterization

3 Getting Digitized Image

Digitized image of a character can be obtained directly from some electronic device or by scanning an image. We used both methods. The quality of digitized scaned image depends on various factors such as image on paper, scanner and attributes set during scanning. The quality of digitized image obtained directly from electronic device depends on the resolution of device, source of image, type of image etc. Fig. 1 shows a digitized image.

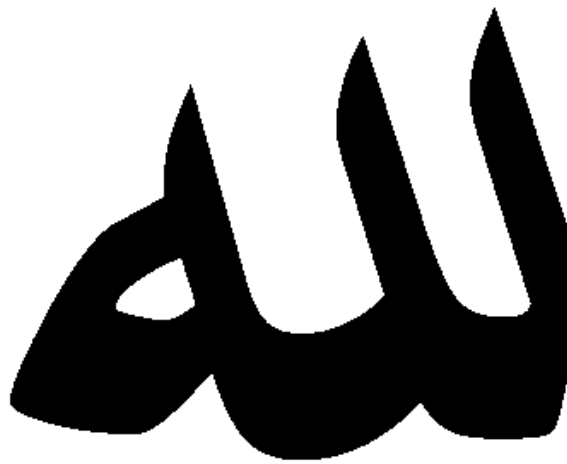


Fig. 1. Digitized Image

4 Extracting of Boundary

Boundary of digitized image is extracted by using some boundary detection algorithm. There are numerous algorithms for detecting boundary [3]. We have

used the algorithm in [11]. The input to this algorithm is a bitmap file. The algorithm returns number of pieces and for each piece number of boundary points and values of these boundary points $p_i = (x_i, y_i), i = 1, \ldots, N$. Fig. 2 shows detected boundary of the image of Fig. 1.

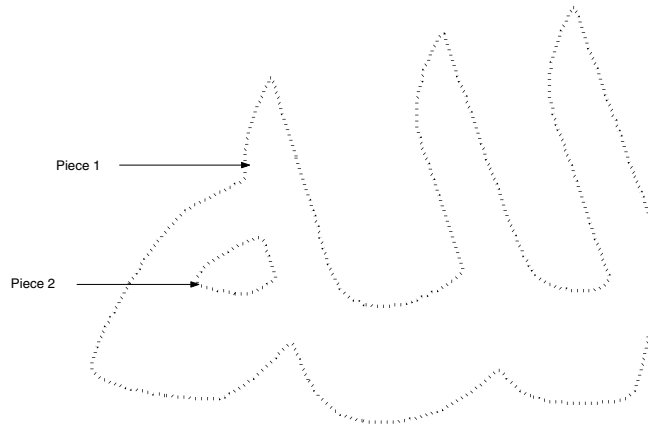


Fig. 2. Detected Boundary of the image in Fig. 1

5 Detecting Corner Points

Next important step, in the development of proposed algorithm, is detection of *Corner Points*. *Corner Points* are those points which partition the outline into various segments. A number of approaches have been proposed by researchers [6,8]. These include Curvature analysis with numerical techniques and some signal processing methods. The detection of corner actually depends on the actual resolution of the image and processing width to calculate the curvature. We adopted the technique used in [7], this algorithm sometimes gives duplicate corners. The algorithm has been modified so that the duplication is removed. Let

$$C^k(i) = a_{ik}.b_{ik}/|a_{ik}||b_{ik}| \tag{1}$$

where

$$\begin{aligned} a_{ik} &= (x_i - x_{i+k}, y_i - y_{i+k}) \\ b_{ik} &= (x_i - x_{i-k}, y_i - y_{i-k}) \end{aligned} \tag{2}$$

A threshold value T for $C^k(i)$ is set. A point P_i is a *Corner Point* if:

1. $C^k(i)$ takes local maxima
2. $C^k(i) > T$

The value of k depends on several factors, such as the resolution of the original digital image. Without threshold value, the algorithm is too sensitive to small variations of $C^k(i)$.

6 Filtering Noise

The points obtained by extracting the contour of digitized image may have noise (jagged edges). Sample data of contour is passed through the filter to remove noise. We adopted a technique known as an approximation to a *Gaussian Filter*. Mathematically we can express *Gaussian Filter* as follows:

$$\begin{aligned} x_i &= 0.5x_i + 0.25x_{i+1} + 0.25x_{i-1} \\ y_i &= 0.5y_i + 0.25y_{i+1} + 0.25y_{i-1} \end{aligned} \tag{3}$$

The filter spread out the local variation by replacing each point with $\frac{1}{2}$ wieghted average of its own value and $\frac{1}{4}$ weighted average of its immediate neighbour values . Filtering process is repeated number of times. By experimentation we found that six to ten iterations are enough to filter noise from the contour. Fig. 3 shows effect of filtering on contour upto six iterations.

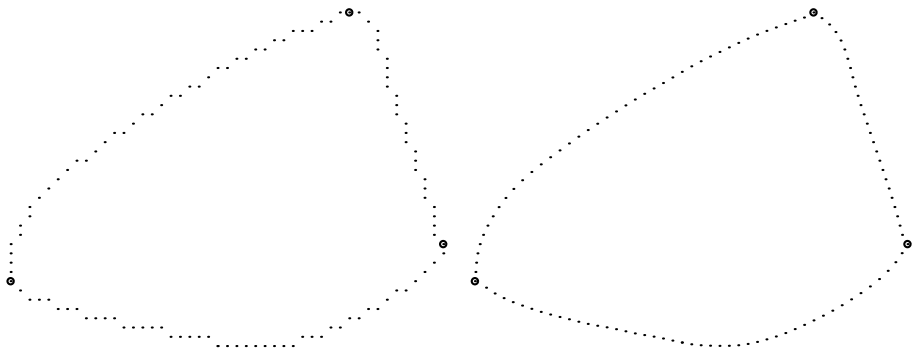


Fig. 3. Digitized Contour (left contour is with noise, right contour is without noise)

7 Fitting Parametric Cubic Bezier

We divide the contour points of each piece into groups called segments and fit cubic Bezier curve to each segment. The division is based on *Corner Points*. It means that if there are m *Corner Points* $cp_1, \ldots, cp_m$ then there will be m segments. The situation is illustrated in Fig. 4 for Piece1 of Fig. 2. If Contour Points of k^{th} segment are $p_u, \ldots, p_w$ then, we can opt the following notation.

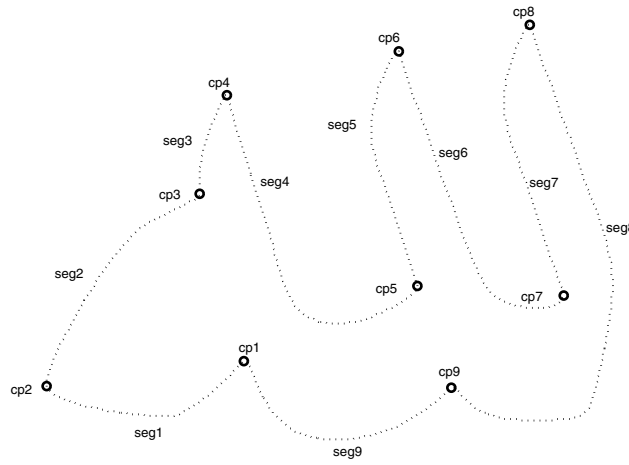


Fig. 4. Corner Points and Division of Contour into Segments

$$p_u = cp_k \text{ where } 1 \leq k \leq m\,.$$

$$p_w = \begin{cases} cp_{k+1} \text{ if } 1 \leq k < m; \\ cp_1 \quad \text{if } k = m. \end{cases}$$

7.1 Finding Intermediate Control Points

The Bezier form of cubic polynomial curve segment has four control points P_0, P_1, P_2 and P_3. The Bezier curve interpolates the two end control points P_0 and P_3 and approximates the two intermediate points P_1 and P_2. The two end control points are the two *Corner Points* of the curve segment. But we have to find the two intermediate control points of cubic Bezier to fit the best optimal curve. The description of cubic Bezier curve is as follows:

$$\begin{aligned} Q(t) &= \sum_{k=0}^{3} P_k B_k(t) \\ &= (1-t^3)P_0 + 3t(1-t^2)P_1 + 3t^2(1-t)P_2 + t^3 P_3\ , \end{aligned} \tag{4}$$

where $0 \leq t \leq 1$, $P_0 = p_1$, $P_3 = p_n$.

We have used *chord-length parameterization* to estimate the parametric value t associated with each point p_i as follows:

$$t_i = \begin{cases} 0 & \text{if } i = 0; \\ \frac{|p_1p_2|+|p_2p_3|+,\ldots,+|p_{i-1}p_i|}{|p_1p_2|+|p_2p_3|+,\ldots,+|p_{n-1}p_n|} & \text{if } i < n; \\ 1 & \text{if } i = n. \end{cases}$$

After having the t value associated with each point, we fit parametric cubic Bezier curve to set of data points of each segment.

7.2 Optimal Curve

Our goal is to approximate the digitized curve by a parametric curve in *best way*. For this purpose, we use the *least square method*. That is, we define the sum of squared distances from the digitized curve to the parametric curve. Mathematically we can write:

$$S = \sum_{i=1}^{n} [Q_i(t) - p_i]^2. \tag{5}$$

Our goal is to minimize S. We find partial derivatives of (5) with respect to P_1 and P_2 and equate them to zero. That is

$$\frac{\partial S}{\partial P_1} = 0\ , \tag{6}$$

$$\frac{\partial S}{\partial P_2} = 0\ . \tag{7}$$

The solution gives values of P_1 and P_2 that approximate the digitized curve by a parametric curve in *best optimal way* for given values of t. Now we have all the four control points P_0, P_1, P_2, P_3 in hand. Using these control points, we fit the cubic Bezier to contour segments. The demonstration of fitted cubic Bezier curves (solid line) over digitized curves (dotted line) is shown in Fig. 5.

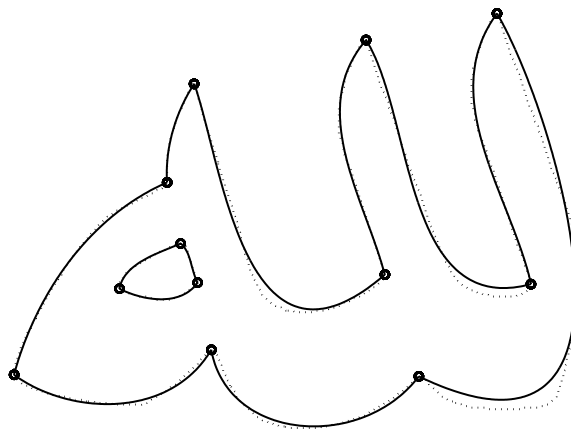


Fig. 5. Fitted Cubic Bezier over Boundary

7.3 Breaking Segments

A fitted Bezeir curve to a segment may not satisfy the threshold tolerance limit. We compute the *squared distance* between each point p_k of digitized curve and its corresponding point $Q(t_k)$ of parametric curve as follows:

$$d_k^2 = |p_k - Q(t_k)| \ . \tag{8}$$

Among all the computed distances by (8), we find maximum squared distance d_{max}^2 as follows:

$$d_{max}^2 = Max(d_1^2, d_2^2, \ldots, d_n^2) \ .$$

If d_{max}^2 exceeds predefined error tolerance limit $d_{tolerance}^2$ then the segment is broken into two segments at the point of maximum distance and the point corresponding to maximum distance is added to list of *Significant Points*. Number of segments and number of *Significant Points* are increased by one.

The process is repeated for each segment until all the segments of all the pieces meet the threshold tolerance limit. Fig. 6 shows the fitted Bezier curve using *Significant Points* (*Corner points* and *Break Points*). *Corner Points* are shown by $\circ$ and *Break Points* are shown by $\triangle$.

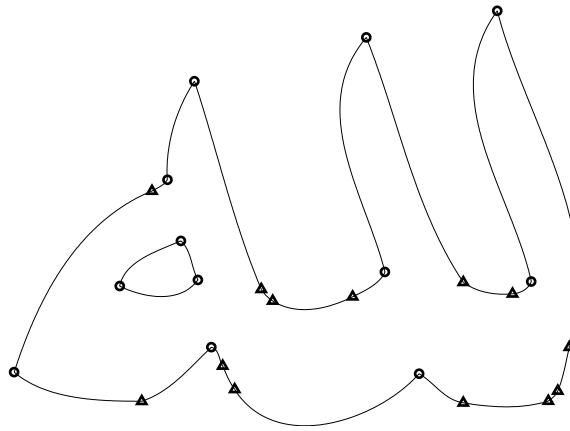


Fig. 6. Fitted Cubic Bezier Outline without Reparameterization

8 Reparameterization

Reparameterization means finding new and better values of t parameter so that we might not have to break a segment into two or more segments and hence we would need lesser number of Bezier curves.

We compare the original digitized curve and fitted parametric curve using squared distance criteria. If parametric curve does not meet the *threshold tolerance limit*, we will apply the reparameterization step rather than breaking it into two segments. After reparameterization, if the segment matches up to the desired tolerance limit then fine, otherwise we break them into two segments as we were doing previously.

Reparameterization is explained as follows: Given a parametric curve $Q(t)$ and a point p, we have to find a point on the parametric curve closest to p. In

other words, we have to find the parameter value t such that the distance from p to $Q(t)$ is minimum. This situation is illustrated in Fig. 7. Note that distance from p to $Q(t)$ is perpendicular to the tangent i.e. $Q'(t)$ of the curve at $Q(t)$. We can write the reparameterization equation as follows

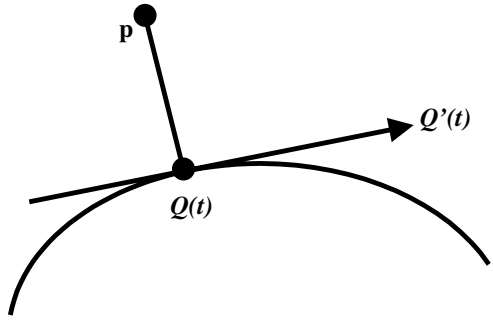


Fig. 7. Distance between p and $Q(t)$

$$[Q(t) - p].Q'(t) = 0 \; . \tag{9}$$

This is a quintic equation[1] in t and can be solved by Newton-Raphson's method.

$[Q(t)-p]$ is polynomial of degree three and $Q'(t)$ is of degree two. So equation (9) is polynomial of degree five. Let

$$Q_1(t) = Q(t) - p \; , \tag{10}$$

$$Q_2(t) = Q'(t) \; . \tag{11}$$

Now, we can write equation (9) as follows:

$$Q_1(t).Q_2(t) = 0 \; . \tag{12}$$

We already have initial approximation of roots (i.e. t parameter). The new and better value of parameter t can be determined by Newton-Raphson method as folows:

$$t_{new} \leftarrow t_{old} - \frac{f(t)}{f'(t)} \; . \tag{13}$$

Using new values of parameter t we find new control points and apply the fitting process as usual. Since reparameterization is an expensive process, therefore we fix a maximum limit on number of times a segment can go for reparameterizaton. If either the maximum limit of reparameterization exceeds or reparameterization increases the square distance between digitized curve and parametric curve rather than decreasing it (in some cases it is possible) then we break the segment into two segments at point of maximum distance. Fig. 8 shows fitted cubic Bezeir with reparameterization.

[1] This is dot the product of $[Q(t) - p]$ and $Q'(t)$.

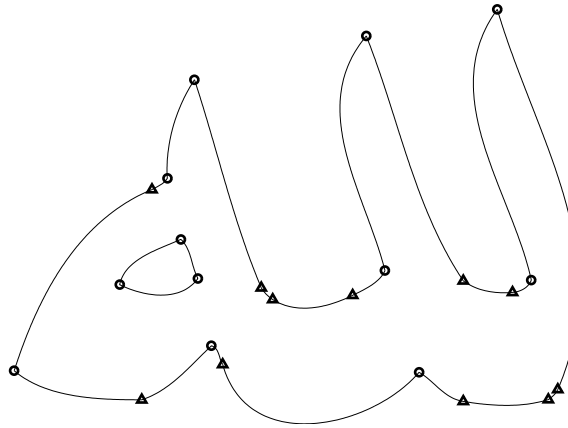


Fig. 8. Fitted Cubic Bezier Outline with Reparameterization

When all the segments meet the square distance threshold limit then there is no need to keep the specific t values. We used specific t values to find the best possible intermediate control points and providing initial estimate for finding new t values in Newton-Raphsons' method. Now we can write a general expression for finding t values. By this general expression we can find t values on the fly (i.e. during fitting Bezeir to a segment). If we fit a cubic Bezier to a segment using n points then the expression for t can be written as follows:

$$t_i = \begin{cases} 0 & \text{if } i = 1; \\ t_{i-1} + 1/(n-1) & \text{if } 2 \leq i \leq n-1; \\ 1 & \text{if } i = n. \end{cases}$$

Fig. 9 shows final outline of the image in Fig. 1. Reparameterization step improves the peformance of algorithm by reducing number of *Significant Points* required in fitting cubic Bezier. Fig. 8 requires 23 *Significant Points* (with reparameterization) as compared to Fig. 6 that requires 25 *Significant Points*(without reparameterization).

9 Conclusion

An improved algorithm for approximation of boundary of digital character images has been presented which is particularly suitable for non-Roman languages like Arabic. However, it can be used for Roman languages too. In addition to the detection of *Corner Points*, a strategy to detect a set of *Break Points* is also explained to optimize the outline. Parametric cubic Bezier curves are used for fitting. Filtering Noise and Reparameterization steps are added to improve the performance of algorithm. The proposed approach eliminates the human interaction in obtaining the outline of original character [1].

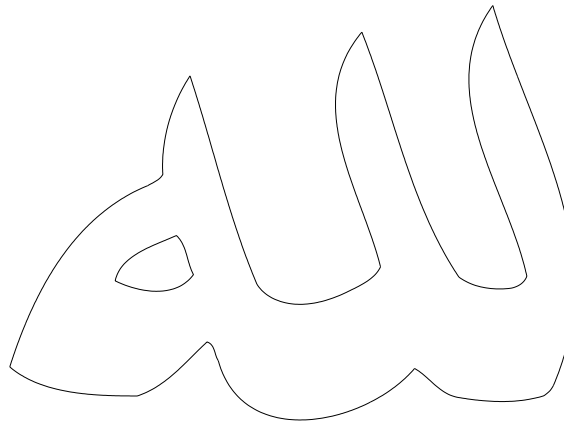


Fig. 9. Final Outline

Acknowledgement

The authors acknowledge the support of King Fahd University of Petroleum and Minerals in the development of this work.

References

1. Murtaza Ali Khan: An Efficient Font Design Method. MS Thesis, KFUPM. (Jan. 2001)
2. M. Sarfraz and M. A. Khan: Automatic outline capture of Arabic fonts. Information Sciences, (2002) 269–281
3. Avrahami, G. and Pratt, V.: Sub-pixel edge detection in character digitization. Raster Imaging and Digital Typography II, (1991) 54–64
4. Hussain Fiaz: On the Capture and Representation of Fonts. PhD Thesis, Brunel University, England (Dec. 1991)
5. Koichi Itoh and Yoshio Ohno: A Curve fitting algorithm for character fonts. Electronic Publishing, Vol. 6, No. 3. (1993) 195–198
6. Beus H. L.: An improved Corner Detection Algorithm based on chain coded plane curves. Pattern Recognition, Vol. 20, No. 3. (1987) 291–296
7. Davis L: Shape Matching using relaxation techniques. IEEE Trans. PAMI. (1979) 60–72
8. H. C. Liu and M.D. Srinath: Corner detection from chain-code. Pattern Recognition. (1990) 51–68
9. Karow P.: Font Technology Methods and Tools. Springer-Verlag, Berlin (1994)
10. Karow P.: Digital Typefaces: Description and Formats. Springer-Verlag, Berlin (1994)
11. Azhar Quddus: Curvature Analysis Using Multiresolution Techniques. MS Thesis, KFUPM. (1998)

Hardware-Accelerated Real-Time Rendering for 3D Sumi-e Painting

Shin-Jin Kang, Sun-Jeong Kim, and Chang-Hun Kim

Department of Computer Science and Engineering, Korea University, 1, 5ka, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
{directx, sunjeongkim, chkim}@korea.ac.kr

Abstract. This paper presents a method for real-time 3D Sumi-e rendering using normal graphics hardware. Sumi-e is one of the traditional oriental painting styles. Most research on Sumi-e paintings has focused on 2D or 2.5D Sumi-e brushwork simulation. On these systems, complicated user's hand drawing is required to generate the image of Sumi-e effects, and it can render the 2D or 2.5D Sumi-e images only. We present an automated rendering system for 3D image of Sumi-e painting. It uses 3D common object as an input data and does not need any additional input of user brushwork. Especially for the real-time rendering, hardware-accelerated algorithm for Sumi-e rendering is newly suggested in our system. It is designed with efficiency for customer level graphics hardware. The results of this paper show that the features of traditional Sumi-e painting are successfully modeled and that 3D Sumi-e painting is rendered in real-time effectively.

1 Introduction

The Sumi-e is the traditional oriental art form of Black ink painting. It is a simple elegant Black ink art on white paper. It is non-photorealistic rendering (NPR) which stands in contrast with to conventional graphics rendering methods of photo-realistic. The recent tendency of NPR system is simulating painting style and natural media, e.g. pen and ink, watercolor, charcoal, pastel, hatching, etc. About the Sumi-e style in NPR, many researches of 2D drawing systems have been shown. In these areas, the delicate simulations of brush, black ink and paper are presented, and a 2D Sumi-e image is generated accepting the hand drawing of the users. In this paper, we present a real-time rendering method for 3D Sumi-e painting. There are three major challenges for Sumi-e painting. First, there is no model for 3D rendering Sumi-e painting. When generating the scene of Sumi-e painting style in 3D games or virtual environment, we can not apply previous 2D rendering methods directly. We need a new concept for 3D rendering technique of Sumi-e painting. Second, 2D drawing work is always required to generate Sumi-e image with previous methods. That leads common users to find difficulties in making it, because they are short of experience. An automated rendering system will enable users to construct Sumi-e scene using 3D arbitrary objects without complicated painting skills. At last, most of previous

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 599–608, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2D Sumi-e drawing systems are interested in its quality not processing time. However new 3D system for 3D game and virtual environments must guarantee real-time performance.

The following are the three main contributions of this paper. First, this paper describes the new modeling approach for turning to real-time 3D NPR Sumi-e rendering. We newly modeled 5 features of Sumi-e painting for 3D Sumi-e rendering: *Nong-Dam, Bal-Muk, Brush Stroke, Atmospheric Depth & Inside Omission* and *Paper Effect*. Users will be able to express their emotions efficiently with these representative features. Second, an immediate Sumi-e painting system is suggested using an arbitrary 3D model as an input data. It does not require users' complicated painting input. And it has the feature of supporting various styles of Sumi-e with selectable rendering modules. Third, the real-time rendering is achieved on customer level graphics hardware by using hardware-accelerated rendering algorithms. These are the programmable shading, sphere mapping, shell texturing, and the composition of paper buffer with fiber structure. Our rendering algorithms do not require much processing time for 3D operations. And they are optimized for hardware acceleration for the best real-time performance.

2 Related Work

We implemented real-time rendering system for 3D Sumi-e. There have been number of systems for 2D Sumi-e brushwork and real-time NPR rendering. Early efforts in 2D Sumi-e painting focused on a brush work simulation. Strassmann [9] swept a one dimension texture to show shading tone. Pham [6] modeled brushstrokes based on variable offset approximation of uniform cubic B-splines. Using the theory of elasticity, Lee [3] modeled a brush as a collection of rods with homogenous elasticity along the entire brush. Way [10] presented a method of synthesizing rock texture in Chinese landscape painting. In real-time rendering of NPR, a few recent systems have addressed real-time NPR rendering. Majumder [5] has implemented real-time charcoal rendering applied with CEO (Contrast Enhanced Operators) by using hardware-accelerated bump mapping and Phong shading. Lake [2] presented a method for cartoon rendering suits for programmable pipeline. Praun [7] introduced TAM (Tonal Art Map) representation and showed that it permited real-time rendering of stroke-based texture for hatching rendering. And he also suggested hardware hatching system with Webb [11] using volume rendering and pixel shading.

3 Feature of Sumi-e

The Sumi-e is one of the traditional arts of Asia (It is called *Su-Muk* in Korean). The essential tools are called the four treasures. These are the brush, Black ink, ink stone and paper. The painter uses Black ink freely with controlled brush strokes. They create beautiful lines and forms by means of brush strokes using a number of techniques and methods to bring life to the subject. The flow and spread of ink on the paper in many shades is an idealization of the form itself.

Sumi-e painting usually speaks simply from the power of its basic inspiration. Artistic beauty most often lies in that which is natural and has personality. Sumi-e is classified into many styles by the drawing technique. Among them, Gu-Ruk is one of the representative Sumi-e techniques together with "Mol-Gol". It draws an object outline silhouette with various brush styles and paints inside in black tone. We present a method for rendering 3D Sumi-e with the technique of Gu-Ruk.



Fig. 1. Two representative Sumi-e painting styles (*left: Gu-Ruk style right: Mol-Gol style*)

4 Feature of Hardware-Accelerated Based Algorithm

A programmable pipeline using a GPU has shown remarkable hardware acceleration in T&L and pixel shading. But there are still many limitations as follows: the limited number of instructions and registers, the limited code size, and absence of if & branch instructions (in Direct 8.1 SDK). Because of these critical limitations, while designing of new algorithm which is to use programmable pipeline or importing of existing rendering algorithm, an optimization process for graphics hardware is essentially required. We try to achieve real-time rendering by designing effective rendering algorithms suitable for hardware acceleration considering limitations above.

5 Modeling of 3D Sumi-e Effect

The rendering of 3D Sumi-e scene needs a new modeling process. We modeled Sumi-e effects into 5 categories as follows - *Nong-dam, Bal-muk, Brush stroke, Atmospheric depth & inside omission* and *Paper effect.*

Nong-Dam Effect In the Sumi-e painting, shading model is different from an ordinary one in lighting source. It should be able to express imaginary volumetric feeling of object. It is called *Nong-dam* effect. In details, in Sumi-e painting, this kind of property can be expressed by varying luminosity from dark tone to bright



Fig. 2. A Sumi-e painting by Dong-Suk Jang

tone forwarding outside. In 3D Sumi-e rendering, a new shading model which is able to express this kind of property is required.

Brush Stroke Effect Brush stroke is an important method in Sumi-e painting for explaining the object shape and the painter's emotion. In the 3D space, it needs the detection of the silhouettes and features of the input 3D model, and rendering of 3D line. Because it usually takes the long computation time, fast and stable silhouette detection and line drawing algorithms for real-time rendering should be considered. And during this process, system should be able to support the function of changing the property of brush stroke for the users by giving the parameter control.

Bal-Muk Effect One of Sumi-e features is irregular black ink spreading, which is settled by the ratio of black inks to absorbed water amount in brush. It is called *Bal-muk* effect. Therefore, an additional shading method to express this effect is required for the 3D Sumi-e rendering system.

Atmospheric Depth and Inside Omission Effect Atmospheric depth refers to the fact that objects get blurry and hazy in the distance due to particles in the air. In the Sumi-e painting, objects in the background are usually painted with the diluted ink to give the distance feeling and add depth to the painting. And also, in many Sumi-e painting, because of simplicity and deformation concept, object inside is usually omitted.

Paper Effect The paper used in Sumi-e painting differs from the one typically used in other paintings. It is much thinner, textured, and quite absorbent, so painted fluid flows easily along its fibers. This occurs because the paper is a mesh of fibers, and the spaces between them act as thin capillary tubes to transport water away from the initial position. Three paper types are commonly used in Sumi-e painting, i.e., rice paper, egg paper, and silk cloth. Because of their characteristic fiber structure, qualities and dilution styles are very different from each other. In 3D Sumi-e rendering system, options for selecting paper type are necessary for the user.

6 Sumi-e Rendering System

We implemented the NPR rendering system based on five Sumi-e effects modeling. Our system consists of the six modules of the Sumi-e effects as follows. Figure 3 shows the overview of the system.

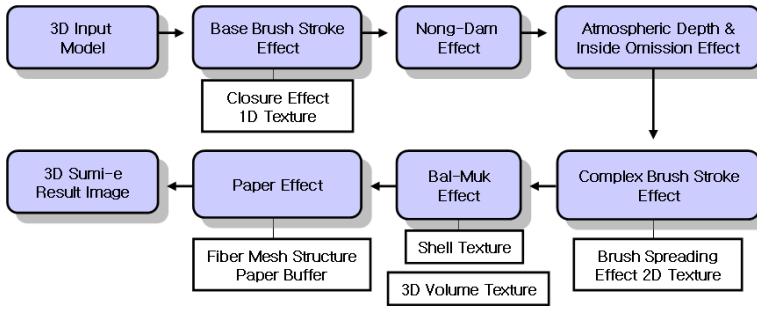


Fig. 3. System flow

6.1 Nong-Dam Effect

In the Sumi-e painting, Nong-dam effect should be able to express an imaginary volumetric feeling of object. So in our system, we used equation (1) as to have an alpha value at each vertex to express Nong-Dam effect. And then do alpha test with some threshold value in order not to render near edge pixels for outside silhouette drawing.

$$Va = (V - C) \cdot N \tag{1}$$

(V: view space vertex position, C: view space nong-dam effect center position
N: view space vertex normal, Va: alpha value of Nong-dam)

6.2 Brush Stroke Effect

The explicit computation of silhouettes requires long processing time. And it needs if & branch operations, which are not inappropriate for the current programmable pipeline. Therefore, it is not suitable for hardware acceleration based rendering system. Instead, rendering the texture mapping achieves the brush effect without an explicit computation of silhouettes. In our system, we used two types of silhouette line drawing based on texture mapping. Artists may sometimes like to leave some parts of the silhouette unrendered or very soft. The observer is allowed the freedom to imagine the interpretation of the effect. This effect is termed by the artists as the *closure*. To express brush drawing with

closure effect, we define closure effect center point and make diffusion variation from that center (base brush stroke) (2). Each Sumi-e painting is composed of many unique brush strokes. To achieve these brush styles, we additionally used sphere mapping with various sets of brush spreading texture (complex brush stroke) (3). Sphere mapping can be accelerated by current hardware and has an advantage of not requiring additional calculation for silhouette detection. And it can also show various silhouette drawing effects easily by changing brush texture source. Final brush strokes are rendered by compositing two brush drawings above.

$$Ti = max\{(C \cdot N), 0\} \tag{2}$$

(C: view space vector from vertex to closure effect position,
N: view space vertex normal, Ti: 1D texture index value of base brush)

$$\begin{aligned} r &= 2(N \cdot E)N - E \\ m &= \sqrt{r_x^2 + r_y^2 + (r_z + 1)^2} \\ Tu &= r_x/2m + 0.5 \\ Tv &= r_y/2m + 0.5 \end{aligned} \tag{3}$$

(r: view space reflection vactor E: view space eye position vector,
N: view space vertex normal, Tu: u texture index value of complex brush,
Tv: v texture index value of complex brush)

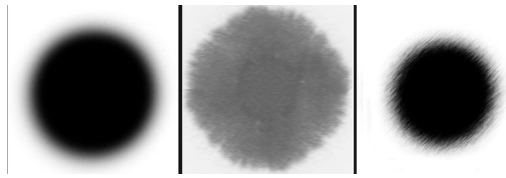


Fig. 4. Various 2D textures of brush spreading

6.3 Bal-Muk Effect

To simulate this Bal-muk phenomenon, we added supplementary shading effect by using shell texture [4]. The shell method creates a four channel (RGBA) image for each layer of the shell model by overlaying a grid on the bounding box containing the black ink spreading. Because shell texture is a real-time method for expressing volumetric feeling with the customized outline silhouette, interpolated ink spreading effect can be taken effectively on the fly. We created

the shell texture, based on the spreading of black ink. We use equation (4) to calculate shell position. And equation (5) for texture index value.

$$Pos = V_o + (h/m) \cdot N \tag{4}$$

(Vo: original view space vertex position, h: shell level,
m: maximum number of Shells, N: view space vertex normal,
Pos: vertex positon of shell)

$$z_index = 1/\{k \times (h/m)\}^q \tag{5}$$

(k: scaling value, h: shell level, m: maximum number of Shells,
q: attenuation factor, z_index: z texture index value of shell texture)

6.4 Atmospheric Depth and Inside Omission Effect

Fogging is an effective hardware acceleration method for hiding object in real-time. And Programmable shading supports customized fogging. To achieve this atmospheric depth & inside omission effect, in our system, exponential fogging is used for atmospheric depth and spherical fogging with noise value for inside omission effect.

$$Fv = \{(V_o - C)/(V_{max} - C)\}^q \tag{6}$$

(V0: view space vertex position, C: view space object center position,
q: attenuation factor, Vmax: maximum vertex position from center C,
Fv: fogging value of inside closure effect)

6.5 Paper Effect

To simulate this effect, we constructed paper buffer having a fiber mesh structure [3]. The image which has passed the 4 stages is sent to a paper buffer first. And then image processing is achieved on that paper following the line of mesh fibers. The paper buffer consists of 2^n blocks and the intensity of image processing is determined in proportion to the number of fibers in a block. By blending this paper buffer with the previous 3D rendering result using the screen aligned billboarding, final 3D Sumi-e image is generated. Each pulp is generated with using equation (7) in the paper buffer.

$$Pulp_Pos = c \sin \frac{2\pi x}{l} \qquad (\frac{-l}{2} \leq x \leq \frac{l}{2}) \text{ (l:pulp length)} \tag{7}$$

7 Result

Results with various shader combinations are seen at Fig. 6. Style(A) uses Nong-dam, Base brush stroke and Bal-Muk effect to empasize simplicity. Style(B) uses Nong-dam, Base brush stroke and Complex brush stroke to accentuate brush

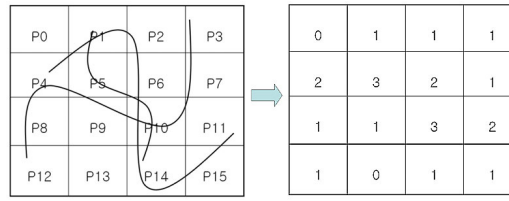


Fig. 5. Pulp distribution in paper buffer

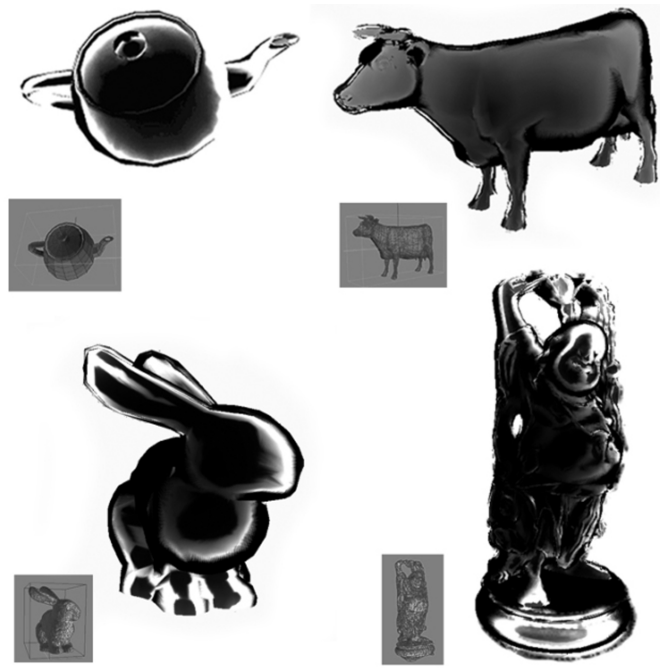


Fig. 6. Rendering results with various shader combinations (*top left: style(A) top right: style(B) bottom left: style(C) bottom right: style(D)*)

strokes. Style(C) shows the irregualr spreading of Black ink by adding Bal-muk effect on style(B). Style(D) uses all the effects and shows paper effect. Computer-generated Sumi-e paintings drawn by our system are seen at fig. 7. Our system consists of Pentium IV 1.7GHz, 640M RAM, nVIDIA GeForce4 Ti 4400 video card. Table. 1 shows the real-time performance. All these models render from 12 to 140 frames per second on our system. The Brush style & width, black ink spreading, Bal-muk effect, paper type and shader composition type can be adjusted by the user to satisfy user's various desire of painting.

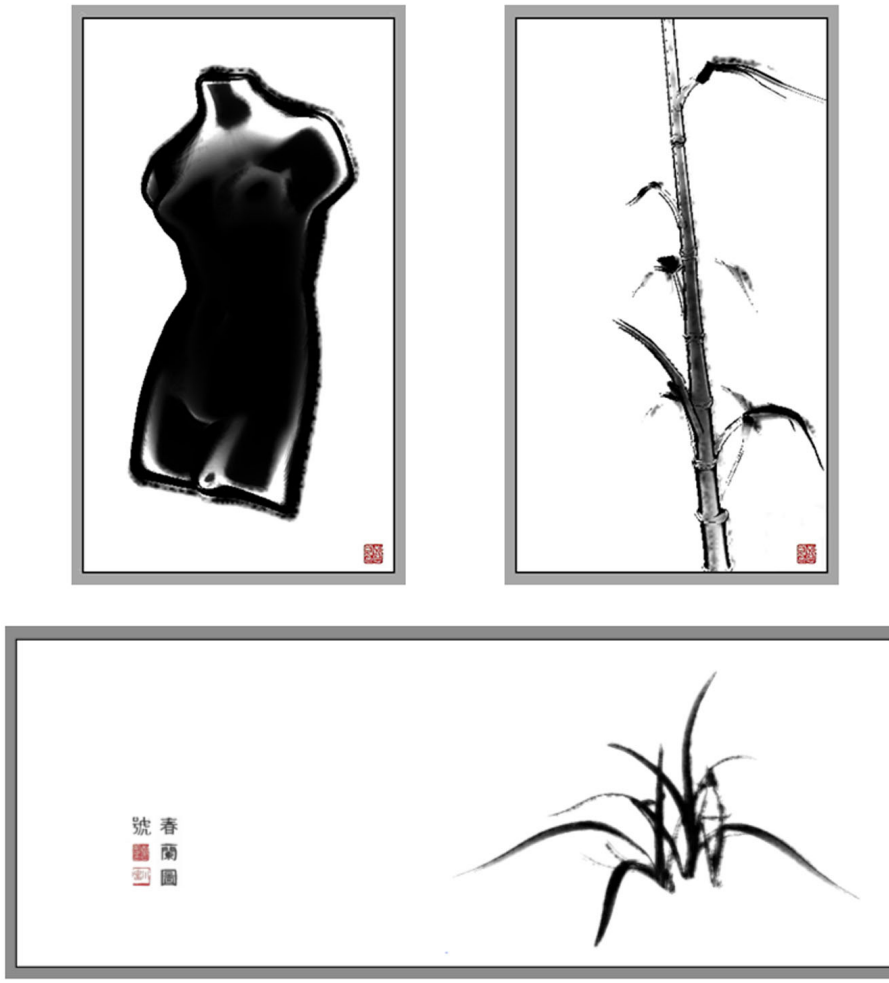


Fig. 7. Computer-generated Sumi-e paintings

Table 1. Real-time performance (unit:fps)

Model	*#Face*	*Style(A)*	*Style(B)*	*Style(C)*	*Style(D)*
Bunny	1,858	142	146	140	36
Cow	2,888	133	135	129	32
Bamboo	7,500	65	124	53	18
Happy Buddha	15,000	35	95	31	12

Acknowledgments This work was supported by grant No. R01-2002-000-00512-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

1. W. Heidrich, H.-P. Seidel, Realistic, Hardware-accelerated Shading and Lighting, Proceeding of SIGGRAPH 1999, pp. 171-178.
2. A. Lake, C. Marshall, M. Harris, M. Blackstein, Stylized Rendering Techniques For Scalable Real-Time 3D Animation, Proceedings of NPAR 2000, pp.13-20.
3. J. T. Lee, Diffusion Rendering of Black Ink Painting Using New Paper and Ink Models, Computers & Graphics 25, pp. 295-308.
4. J. Lengyel, E. Praun, A. Finkelstein, H. Hoppe, Real-Time Fur over Arbitrary Surfaces, Symposium on Interactive 3D Graphics 2001, pp. 227-232.
5. A. Majumder, M. Gopi, Hardware Accelerated Real Time Charcoal Rendering, Proceeding of NAPR 2002, pp. 59-66.
6. B. Pham, Expressive Brush Strokes, Graphical Model and Image Processing 53(1), pp. 1-6.
7. E. Praun, H. Hoppe, M. Webb, A. Finkelstein, Real-Time Hatching, Proceeding of SIGGRAPH 2001, pp. 581-586.
8. T. Pudet, Real-Time Fitting of Hand-Sketched Pressure Brush Strokes, Proceeding of Eurographics 1994, pp. 205-220.
9. S. Strassmann, Hairy Brush, Proceeding of SIGGRAPH 1986, pp. 225-232.
10. D. L. Way, Z. C. Shih, The Synthesis of Rock Textures in Chinese Landscape Painting, Computer Graphics Forum Vol. 20 Number 3, pp. C123-C131.
11. M. Webb, E. Praun, A. Finkelstein, H. Hoppe, Fine Tone Control in Hardware Hatching, Proceeding of NPAR 2002, pp. 53-58.

Generation of Symmetrical Patterns Based on Text and Colour

Amer Salman, Rula Salman, Edwin Love

1 Thames Valley University, Course Leader for Research Degrees, London College of Music Media and Creative Technology, Grove House, 1 The Grove, Ealing
amer.salman@tvu.ac.uk
http://www.tvu.ac.uk
2 ABC Europe Ltd, I.T. Assessor, No 72 New Bond Street, London, W1S 1PR, UK,
Tel:++44(0)207586 8522
rula.salman@yahoo.co.uk
London W5 5DX, UK, Tel: ++44 (0) 208 231 2235
3 Thames Valley University, Lecturer Assistant, London College of Music Media and Creative Technology, Grove House, 1 The Grove, Ealing
edwin.love@tvu.ac.uk

Abstract. When we look at an item of artwork, we are invited to consider on individual description or observation at a time. Through our research we have found a way to combine two or more descriptions and observations within a single artefact. Poems or just plain text can be seen as artworks. In this paper we explain how we can 'read' a set of 'instruction' (e.g. poems or prose) which transform themselves into images. Once created, these unique constructions, which richly illustrate and reflect the influence their language whether random, symmetrical or recurring. In producing such a work of art we might hypothesise that once an item of literature is written it contains within it it's own pattern of imaging. It would be interesting to know what pattern and images are produced when analysing word structures that are 100's of years old. Analysing and transforming these words into graphical representation in this way may reveal elements in the work of art that has not hitherto to been seen by man.

Keywords: symmetry, poem, artefact, art, and computer graphics.

1. Introduction

Over the past few years the authors have studied several hundred symmetric geometrical patterns using computer graphics [7,8,9,10]. In this paper, we have used different concepts eg. colour, text, symmetry and patterns.

Brilliant colours and abstract geometric designs fascinated many nineteenth century European artists, architects, and designers. Such was the English architect and art decorator Owen Jones who made the first systematic study of architecture in Spain. Most readers would be surprised to learn that such an analysis was used to record and display the colours of architecture in which colour lithography was first employed in book printing in Britain. He published his monumental work The Grammar of Ornament [2], which surveyed world decorative art.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 609-619, 2003.
© Springer-Verlag Berlin Heidelberg 2003

"Symmetry deserves to be the subject of periodic festivals, so that in its lofty atmosphere one can search for the relation between different manifestations of the human spirit", said K. S. Mamedov [4] about symmetry. It is the glue that binds the universe. Symmetry is seen in the rings of Saturn, flowers, fish, and birds and in the skins of soap bubbles. It is often used to recognise beauty, security, universal order stability and understanding and gives the brain the power to predict.

"Life makes patterns out of pattern-less disorder, but I suggest that life itself was made by a pattern and that this design is inherent in cosmic forces to which life was, and still is, exposed", said Lyall Watson about patterns [14]. There are patterns in the sky, the sea, in music and in other words everywhere. Modern research into neural networks has pattern recognition as its key quest.

Dorothy Washburn, an archaeologist at the university of Rochester, and Donald Crowe, a mathematician at the university of Wisconsin in the USA [13], found that the symmetries used and patterns created in various cultures are not random. Therefore people in dissimilar cultures tended to like varied symmetry and patterns. Studies of colour, text and symmetrical patterns, with which this paper is concerned, can therefore have significance that goes well beyond their immediate appeal.

2. Basic Concept & Methods

In this section, we explain and illustrate the common methods used in the structuring of beautiful symmetrical patterns. Our intention is to help the reader to understand the terms and concepts used in this field.

2.1 Grids and Circles

Some patterns could be constructed more easily by starting with a grid. The grids used most commonly include the isometric grid (made from Quattro triangles), the square grid and the rectangular grid. Fig.1 shows patterns based on a grid, which combines squares and equatorial triangles. The construction method is packed with circles (consider the first complete circle in the bottom part) density and employs division into seven. The middle of the construction involves joining the points as illustrated in the figure. We can have a different version of the pattern based on the original using an overlapping and filling technique as shown in the rest of the pattern.

2.2 Periodic Pattern Type

Various portions of a given pattern can be selected to replicate a pattern by simply copying and pasting repeatedly. Fig.2c shows example of such a pattern. Fig.2b is a unit cell that is used to repeat and generate the whole pattern. The unit cell can be rectangular, square rhombus or a special rhombus whose sides include an angle of 60 degrees, furthermore Fig.2b can be generated by isometric transformation (rotation, reflection and glide reflection) in Fig.2a. In fact this figure is internationally named as a p4g pattern type.

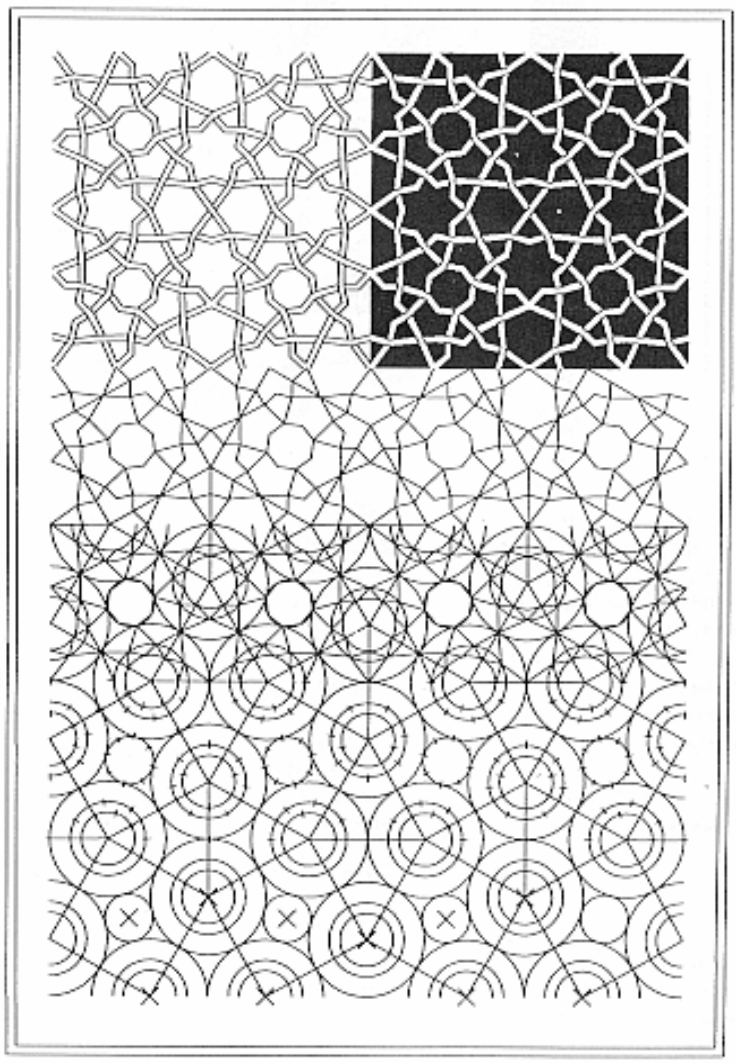


Figure 1: Three complex and beautiful variations of pattern made of grids and circles

Fig. 1 and Fig. 2 are both the same patterns, where the suggested construction is of course quite different. Different author's [6,12,1] have suggested different methods to construct such a pattern. For an explanation of how patterns are classified and how they can be recognised for their symmetrical property the reader should refer to chapter 3 of [8].

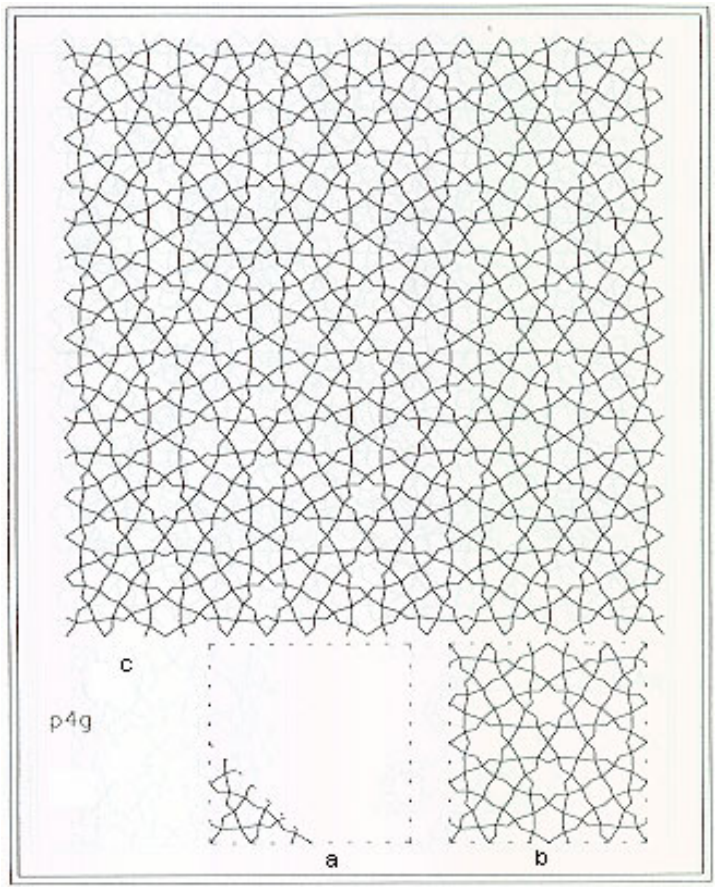


Figure 2: P4g is the international name of the pattern drawn by lines

3. Analysis of Relevant Study

We found three relevant studies that are worth mentioning. We have given our own name for each one.

3.1 An Engineer Approach

John Maeda [3] argues that at present computer art often involves collaboration between an artist and an engineer. He believes that there is a need for a new breed of creator, the artist that is also an engineer. Maeda also argues that we need to create tools whereby the artist has all the control. To establish his arguments he created a system called “Design by Numbers”.

3.2 Grammatical Approach

Martin Wattenberg [15] has carried out some work into what he calls syntax highlighting. This is where different grammatical element of the text is given different colours. It allows for much more information to be gleaned when scanning rather than reading (and as such is perhaps ideal for hypertext writing) and can help to resolve some of the ambiguities inherent in English at least, whereby a single word can be a noun, a verb or an adjective.

3.3 Synethesia Approach

Tsutumo Miyasato [5] took Synethesia as the basis of some of the work that he did on producing images from Japanese Tanka poems by representing sounds or phonemes with colours. His team's goal was to "generate images so that passions are reflected". His team produced a series of beautiful images.

4. The Investigation

Our previous work has dealt with the production of symmetrical patterns through the use of geometric and group-theoretic methods [9], also, the use of methods for the seventeen distinct types of two colour frieze patterns [10]. Work by other researchers and scientists had been carried out to produce images through the use of other methods that are not based on colour, text and symmetry.

The main bulk of this work was to develop methods to design and build an application in Java programming language that will allow for the automatic transformation of text into colours, which then produce a symmetrical pattern. We then went beyond the steps of producing a traditional image and analysed and categorised the produced image against the seventeen types of symmetrical patterns.

We wanted to produce a tool that can be used, by an artist, to produce images. Anyone can then use this programme. One does not have to have full knowledge or background of mathematical methods, or to have used computer graphics beforehand. We included a number of easily adjustable parameters such as the ability to change the colour mapping used and also the size, shape and arrangement of the tiles that makes up the big image. The aim was to make the artist able to simply tweak things and get an instant response. This tool could be used to investigate the influences of style, language and content on the images produced.

5. Generation of Symmetrical Patterns by Text & Colour

The process of producing symmetrical patterns went through several stages in order to reach the final presentation. And for this we followed specific steps and instructions.

- We created a mapping system that would relate individual letters to different colours i.e., if we type the letter (a) this would generate a different colour from

the letter (b). This lead to once a text is typed the mapping system would automatically transform the text.

- When we repeated the process of the colour mapping system we then ended up with a unit cell representing the typed text. This individual unit cell would then give us a tile of the image or the picture we were aiming to produce.

- Having implemented the first two steps, the programme is now able to repeat the unit cell a number of times to produce a final pattern of colours from the typed text.

- At this point the user can then take over and be able to alter the appearance of the final image through the use of tools that are available such as re-size, zoom and other options available.

- We found that some of the most interesting patterns came from quite simple phrases that were repeated a number of times. In our paper, we have used poems written by famous poets instead of general or simple phrases. We have analysed the generated patterns using the group-theoretical analysis, for example see [11].

The pattern in Fig.3 is a P4mm type. It is generated from a poem by John Wilbye (1574-1638).

"Love not me for comely grace,
For my pleasing eye or face,
Nor for any outward part;
No, nor for a constant heart!
For these may fail or turn to ill:
Should thou and I sever,

Keep, therefore, a true woman's eye,
And love me still, but know not why!
So hast thou the same reason still
To dote upon me ever."

Fig. 4 shows Cmm pattern type obtained from a poem by Percy Bysshe Shelley

"The fountains mingle with the river,
And the rivers with the ocean;
The winds of heaven mix for ever
With a sweet emotion;
Nothing in the world is single;
All things, by a law divine,
In one another's being mingle
Why not I with thine?"

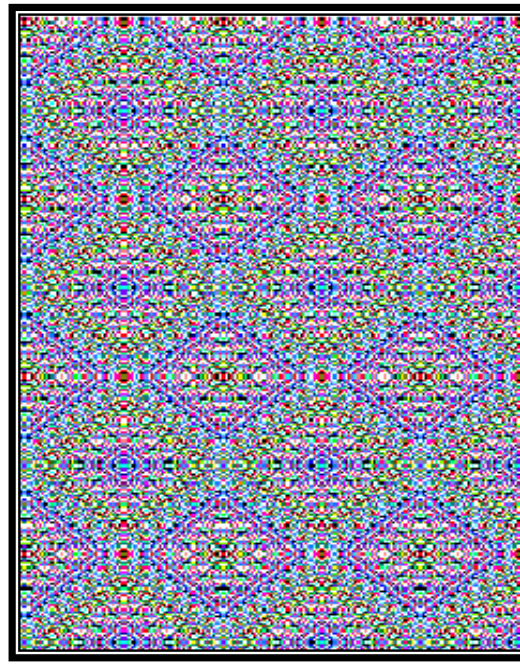


Figure 3: P4m pattern produce poem by John Wilbye

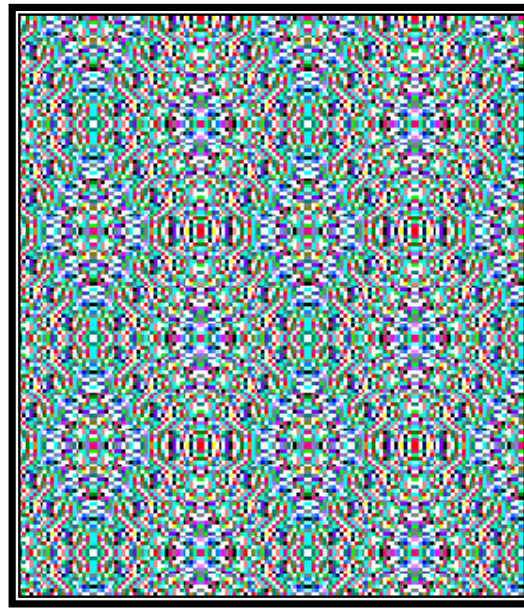


Figure 4: Cmm pattern type produced from Percy Bysshe Shelley poem

Fig.5 and Fig.6 are two different examples.

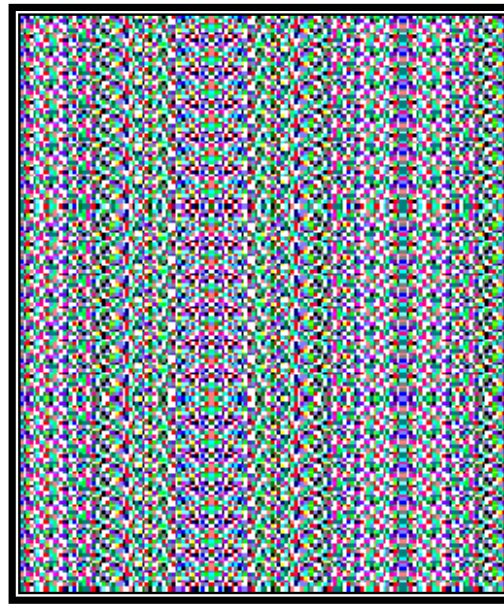


Figure 5: P2m pattern produced by William Shakespeare

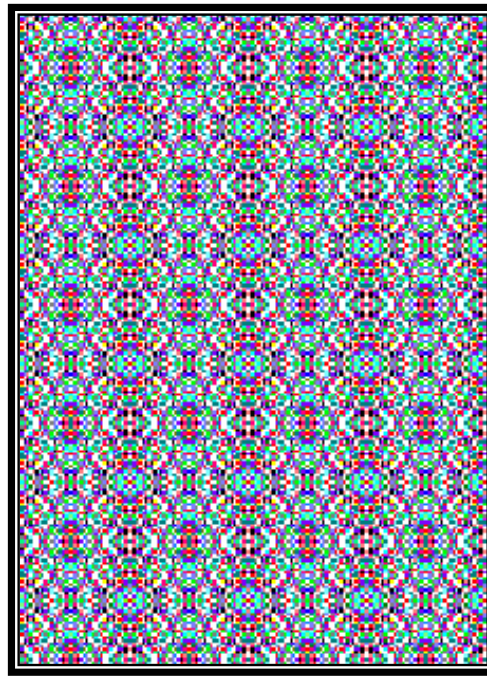


Figure6: Pm pattern produced unknown poet

It should be remembered that our figure analysis refers to group symmetry without colour consideration. If we allow colour of course the conclusion would be quite different.

6. The Software Application

The user of this programme has buttons available on screen to assist in producing a symmetrical pattern. There are different buttons available to transform the text into its final presentation; these include, Load text, Save text, Save tile, Clear text, Repeats, Scale, Palette. As in any software all results can be saved whether this was an image, tile or a piece of text. Fig.7 below is a UML class diagram for the application, where green (rectangles) classes are standard JDK Classes, blue (triangles) classes are 3rd party classes and red (circles) classes are classes that we wrote.

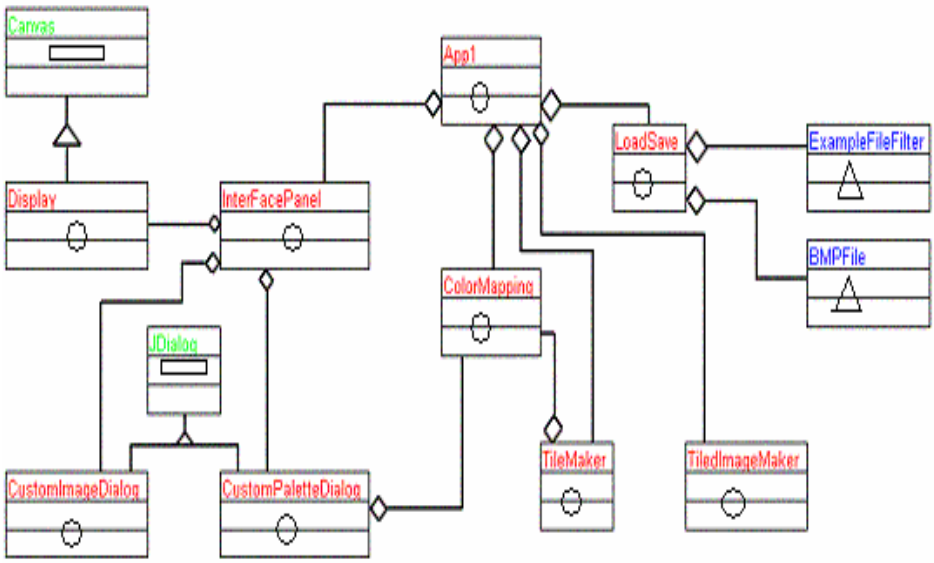


Figure 7: UML Class Diagram

7. Conclusion

This work has established a unity between the written word, colour and the generation of symmetrical patterns. We have found that words and colours have a unique relationship that once established and appreciated, becomes a celebration of unity and symmetry across language forms. The greatest satisfaction in this work come from the ability to transform and translate existing art forms into symmetrical pattern using language colour, symmetry and the media of computer graphics.

A recent discovery by Dorothy Washburn and Donald Crowe [13], noted that some cultures generate or prefer and recognise certain symmetry types. After generating and analysing hundreds of patterns we have found that our method had generated different types of symmetry such as Cmm, P4mm, P2mm etc, and that types such as the Pg, Pgg were very rare.

8. Further Studies

Some of the future developments for our research will be:

- Security systems could also incorporate colour mapping. Just like the fingerprint is a unique code, we could generate one in the form of colour mapping to use this as an added security feature.

- We will experiment on a combination of poems, from different cultures and generations to investigate whether they could reflect their emotional meaning.

- For future studies we aim to produce images generated from musical notes. It would be interesting then to compare the colour meaning of a sad symphony with a happy one. For example comparing a piece of music from Miss Saigon and a piece from the Nutcracker Suite.

9. Acknowledgement

We wish to express our appreciation to Mr Donald Phillip for his support during the research, and to Mr Malcolm Hignett of LCMM for his valuable comments.

10. References

[1] B. Grunbaum, Z. Grunbaum, G.C.Shephard, Symmetry in Moorish and other Ornaments, Comp. & Math. With appls. Vol 12B, 641-653, 1986.

[2] O. Jones, , The Grammer of Ornament, Day and Son, London 1856, recent reprint Studio Editions, London 1988.

[3] J. Maeda, Design by numbers. MIT Press, 1999.

[4] K.S. Mamedov, Crystallographic Patterns, Comp. Math with Applic., vol. 12B, nos 3-4, pp.551-529, 1986.

[5] T. Miyasato, Generation of Passikon Images from 'Tanka' Poems based on the Synesthesia Phenomenon. Information Visulazation, IV99, p516-518, 1999.

[6] J. Rosen, Symmetry at the Foundation of Science, Computers and Mathematics, vol 17, number 1-3, 1989.

[7] A. Salman, E. Nüesch, R. Salman, WORM a Visualisation Engine for the Inner Structure of Racing. Accepted for publication by the ISCA 18th International conference held March 26-28 2003.

[8] A. Salman & J. Abas, Symmetries of Islamic Geometrical Patterns.World Scientific 1995, reprinted on 1998.

[9] A. Salman & S. J. Abas, Geometric and Group-theoretic Methods for Computer Graphic Studies of Islamic Symmetric Patterns, Computer Graphics forum volume 11, number 1, 43-53,1992.

[10] A. Salman, Methods for the Seventeen Distinct Types of Two Colour Frieze Patterns, Leonardo, P324, April, 1992.

[11] D. Schattschneizsder, The Plane Symmetry Groups: Their Recognition and Notation, The American Mathematical Monthly, 85, 439-450, 1978.

[12] A.V.Shubnikov, V.A.Koptsik, Symmetry in Science and Art, Nauka, Moscow, 1972, Plenum Press, New York, 1974.

[13] D. Washburn & D. Crowe, Symmetries of Culture, University of Washington Press, 1982.

[14] L. Watson, A Matter of Life and Death, London, Coronet, 1976

[15] M. Wattenberg of SmartMoney, http://mappa.mundi.net/maps/maps_023/

Reuse of Motion Capture Data in Animation: A Review

Weidong Geng, Gino Yu

Multimedia Innovation Centre
The Hong Kong Polytechnic University, Hong Kong
{weidong.geng, mcgino}@polyu.edu.hk

Abstract. The reuse of motion capture data is receiving increasing attention in computer animation. This interest is motivated by a wide spectrum of time-critical applications, such as video game development and animation production. This paper gives an overview of various tasks and techniques involved in the reuse of motion capture data in terms of motion authoring pipeline. We assume that whenever the user is asked to build motions, he/she will at first go to retrieve/browse the pre-recorded motions in the motion database, look for the best-fit candidate motion segments/pieces/clips, and manage to adapt them to fit the specified requirements. The two core issues in motion reuse, motion adaptation techniques and motion library construction, are the focus of this review.

1 Introduction

Generating realistic motion for character animation remains one of the great challenges in computer graphics, as people have proved to be adept at discerning the subtleties of human movement and identifying inaccuracies. Motion capture is the process of recording motion data in real time from live actors and mapping it into computer characters. It is one of the most promising technologies bringing realistic and natural motions into character animation. The use of motion capture is currently most widespread and well-accepted in video game and animation [1].

However, motion capture also has its share of weakness [2, 3, 4]. Motion capture systems are still very expensive to use. The motion capture process is labor intensive and time-consuming for actors and technicians. In order to make motion capture widely available, the motion capture data needs to be made reusable [5]. This means that we may create the needed motions by reusing pre-recorded motion capture data. Furthermore, with the increased availability of motion capture data and motion editing techniques, it currently yields a trend to create the qualified motion by piecing together example motions from a database [3]. This alternative approach potentially provides a relative cheap and time-saving approach to quickly obtain high quality motion data to animate their creatures/characters.

This survey will concentrate on two core parts in motion reuse: how to build the motion library and how to adapt motions to new needs. Section 2 will present a general overview about how to reuse motion capture data. Section 3 covers the research on motion adaptation techniques. Section 4 extends the discussion to motion capture data representation and motion library construction. Finally, section 5

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 620-629, 2003.
© Springer-Verlag Berlin Heidelberg 2003

concludes the paper by giving general comments on existing work in the area of motion reuse, and discuss future directions of research in this area.

2 Overview of Motion Reuse

The motion reuse is typically carried out by applying appropriate motion transformation techniques on the motion clips in the motion capture database. Lamorout and van de Panne proposed to use a large set of representative motions together with techniques for tailoring these motions to make them fit new situations [6]. Molina-Tanco and Hilton built the statistic models from the database of motion capture examples [7], and then re-arrange the segments of original data in the motion library and generate a realistic synthetic motion by them. Jeong et al developed the motion editing system, Marionette version 0.1[8]. It has a motion database, and a rich set of motion operations such as motion cut/paste, rotation, transition, blending, interpolation and retargeting. From the point of view of implementation, one conceived representative system architecture for motion reuse is shown in Fig1.

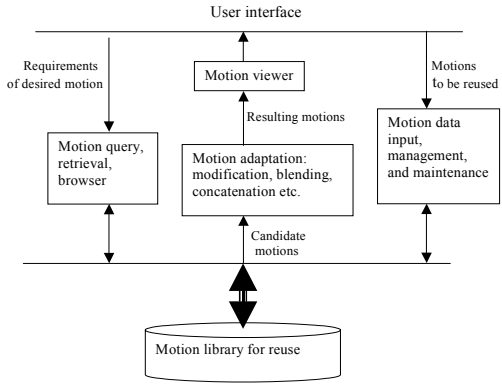


Fig. 1. A potential architecture for motion reuse system

Whenever the user is asked to build motions, he/she will at first go to retrieve/browse the pre-captured motions in the motion database, look for the best-fit candidate motion segments/pieces/clips, and manage to adapt them to the specified requirements. The resulting motions will be presented to the user by motion viewer. If it is accepted, this means the user successfully gets the desirable motions.

From the technical point view of motion reuse, firstly, we should have sufficient available motion data to be reused. Secondly, we need to make candidate motions fit into new situations. Therefore, we can summarize that this motion reusing approach is mainly limited by

1. The library of motions available to reuse, including the range of types of motions, the quantity of data, secondary information needed for reusing purpose (e.g. actor's dimension and skeleton), and convenient tools for database construction and motion data retrieval.

2. The quality of tools available for adapting motions, i.e. how and in what acceptable level we can maintain the realism, aesthetics, and naturalness of the source motions despite all the possible changes in building new motions from them.

3 Motion Adaptation Principles and Operations

3.1 Adjustment Principles

The fundamental issue in motion adaptation is how to generate convincing human motion transformation while retaining the desired characteristics of motions and making the changes to undesired aspects. Recently, much of the research has been devoted to developing various kinds of editing tools to modify and vary the existing motions, and/or produce a convincing motion from prerecorded motion clips. They are roughly reviewed in Table 1. Other noteworthy work includes retargeting motions to new characters [9], motion transitions generation [10] and motion path editing [11].

Table 1. A summary of motion adjustment principles

Methodol1ogy/ Principles	*Assumption and Algorithmic ideas*	*Relevant work*
Signal processing	Motion parameters is treated as a sampled signal, and techniques from the image and signal processing domain are adapted to edit, modify, blend and align motion parameters of articulated figures.	[12], [13], [14], [15]
Constrained optimization	It makes some of the features that are to be preserved or changed in a motion explicit as spatial or geometric constraints such as space-time constraints, inverse kinematics and physical laws, momentum constraints. Then the constraint-based problem solving is employed to generate the desired motion.	[16], [17], [18], [19]
Synthesis By statistical models.	Statistical models (e.g. Hidden Markov Model) are used to "learn" and extract "meaningful" attributes (e. g motion patterns) of motion the statistical properties of example motions, and then perform the motion synthesis on them.	[7], [20], [21]
Interpolation	It assumes that the motions can be parameterized by an interpolation scheme, and then it empirically uses linear or non-linear methods to creates the individual motion clips.	[22], [23]

3.2 Operations Involved in Motion Reuse

Motion manipulation operations can be classified in many ways depending on the criterion that focuses on particular aspect of the techniques. A popular way to categorize them is according to the number of motions involved in an operation. Sudarsky and House defined the motion editing performed on single motion sequence as Unary operations, and the manipulation on two or more motion clips as Binary operations [24]. Jeong et al define them as inter-motion and intra-motion operations [8]. Inter-motion operations need two or more motions, and intra-motion operations need only one. Here we present an empirical classification by their roles and functions in motion building pipeline for motion captured animation.

3.2.1 Motion Modification

Motion modification is going to make changes on the non-essential attributes of source motion. Its basic assumption is that most of central characteristics are preserved in the resultant new motion. The instances of this kind of operations comprise

- Joint repositioning. It means changing the pose of body. For example, Bruderlin and Williams decouples the representation of original motion curve and the displacement curve [12]. Then the user can substitute displacement curve with another one, and accordingly change the positions of joints in each frame.
- Motion smoothing. The smoothness of a motion depends on the density of control points in a motion curve representation. The motion can be smoothed by changing the number of control points using a filtering system [15, 17], or increasing/decreasing motion signals in high or low frequency bands [13].
- Motion warping [14]. The animator interactively defines a set of keyframe–like positional (time) constraints which are used to derive a smooth deformation that preserves the fine structure of the original motion.
- Changing the orientation. Basically it changes the rotation parameters of root or pivot joint in the animated skeleton, for example, changing the walking motion on a straight line into walking on a circle.

3.2.2 Motion Synthesis

Motion synthesis can be described as generating new motions by piecing together a set of pre-captured example motion data. Two or more motions can be synthesized at the same time. The resulting motion is a relatively new motion, which is similar to source motions somewhat in content and/or in style. Existing motion synthesis methods consist of

- Synthesis by statistical models. It is a typical way to synthesize novel movements from examples. It usually at first learns a statistical model from the captured data which enables realistic synthesis of new movements by sampling the original captured sequences. Then the statistical model identifies the segments of the original motion capture data to generate novel motion sequence according to the specified synthesis requirements.
- Synthesis by blending. It means parts of clipped motions are mixed to create a new motion. In a motion curve representation, it generates a curve that lies somewhere

in between the two original curves, according to a blending function or parameters. Partial motion combination can be considered as its extreme cases with a kind of "substitution". This operation is useful when only part of a motion is meaningful.

- Synthesis by interpolation. It generates the intermediate motions from set of example motions and associated parameters. Most impressive results on this operation come from Rose et al [23]. They used combination of linear approximation and radial basis functions to calculate the fitting hyper-surface approximating the example points. It is particularly effective for style manipulation since we can generate various motions without establishing the computation model of styles or nuances.
- Synthesis as graph search [5]. The collection of motion sequences could be represented as a directed graph. The nodes of the graph are individual motion sequences. The edges connect frames of incident nodes (motion clips), where the outgoing edge originates from the last frame of the current motion, and the incoming edge points to the first frame of next motion. The motion generation in this case is converted as searching a suitable path from the graph by the soft and hard constraints specified by the user.

3.2.3 Motion Concatenation

Motion concatenation combines two clip motions seamlessly by the transitional motions, removing discontinuities at the boundary between two motions. The resultant motion is the "addition" of the source motions both in time interval and content. Cyclification can be considered as a special case of concatenating the same motion twice or more. Its key issue is how to smoothly generate transitional motions. Several existing transition generation approaches are

- Blending the overlapping part of motions. It is similar to the image morphing. The blending weights can be monotonically deceasing function in range of [0, 1]. It fades out one motion, and fades in the next one smoothly [24]. It is simple and intuitive, but it may cause unnatural motions such as footskate;
- Space-time constraints [10]. Its motion transition generation uses a combination of space-time constraints and inverse kinematic constraints to generate seamless and dynamically plausible transitions between motion segments.
- Basis controller [25]. They use a small set of parameterized basis controllers, such as balancing controller and landing controller, to create the continuous transitional actions between the two source motions. It can be considered as a simplified version of key-frame animation by controller.

3.2.4 Motion Retargeting

The direct-mapping of captured movements from live subjects to a virtual actor potentially yields observable unnaturalness if their skeleton or surrounding environment has a big difference. Motion retargeting is introduced to solve this problem. The resultant motion shares the same motion content or high-level motion behavior with the source motion, but they are different in the lowest motion data level

The motion retargeting has two typical scenarios:

- Retargeting motion to new characters. Gleicher convert the motion retargeting operations into a set of kinematic constraints and spacetime constraints

formulation, based on the assumption that the skeleton has the identical structure with different size [9]. Bindiganevale & Badler proposed a method to automatically extract the important event using spatial constraints [26]. The visual attention can be kept after the motion is mapped to the virtual character.

– Retargeting motion to new environment. It aims to alter the captured motion into new situation such as new path or new terrain. The path is depicted as an abstraction of positional aspects of the movements. Gleicher presented a path transformation approach to permit a single motion to be applied in a wide variety of settings [11].

4 Motion Database Construction

Motion database is the basis for motion reuse. The major weakness in motion capture data is that it lacks of structure and adaptability. While a captured motion can record the specific nuances and details of a performance, it also specifically records the performance. It encodes a specific performer, performing a specific action, in a specific way [11]. The specificity makes it difficult to alter, especially since the key "essence" of the motion is not distinguished from the large amount of potentially irrelevant details. Therefore, we should ease the difficulty of motion adaptation by storing sufficient information in the motion database, including actor's skeleton, marker information, marker placement, visual attention, footplant, and so on.

4.1 Motion Data Organization

A typical strategy of motion data organization is based on the directed graph. Rose et al employ "verb graphs", in which the nodes represent the verbs and the arcs represent transitions between verbs [23]. The verb graph, acts as the glue to assemble verbs (defined as a set of example motions) and their adverbs into a runtime data structure for seamless transition from verb to verb for the simulated figures within an interactive runtime system.

Arikan & Forsyth also present a similar framework that generates human motions by cutting and pasting motion capture data [5]. The collection of motion sequence could be represented as a directed graph. Each frame would be a node. There would be an edge from every frame to every frame that could follow it in an acceptable splice. They further collapse all the nodes (frames) belonging to the same motion sequence together. Kovar et al construct a directed graph called a motion graph that encapsulates connections among the database [3]. The motion graph is a directed graph wherein edges contain either pieces of original motion data or automatically generated transitions. All edges correspond to clips of motion. Nodes serve as choice points connecting these clips. i.e., each outgoing edge is potentially the successor to any incoming edge. New motion can be generated simply by building walks on the graph.

4.2 Retrieval of Best-Fit Motion

The retrieval of best-fit motion is mainly determined by the motion editing requirements. Arikan & Forsyth classify the motion editing requirement as hard constraints (can be satisfied exactly) and soft constraints (can not be generally satisfied exactly), including [5]

- The total number of frames should be a particular number
- The body should be at a particular position and orientation at a particular time
- A particular joint should be at a particular position (and maybe having specific velocity) at a specific time
- The motion should have a specified style (such as happy or energetic) at a particular time.

In the algorithm level, one of the key issues of such motion retrieval is what similarity should be used to define the fit of a motion. A best-fit motion primitive should satisfy several possible-conflicting preconditions such a good match of character, environment and style. The similarity metric must weight these factors appropriately in order to efficiently extract the best-fit motion from the set of motion examples.

For instance, Lamouret & van de Panne defined the similarity as the following distances over the candidate samples **m**: [6]

$$d_{min}=\min_{\{m\}}(d_{state}+k_1 d_{env}+k_2 d_{user})$$

Where $\mathbf{d_{state}}$ measures the compatibility of the initial motion state. $\mathbf{d_{env}}$ measures the compatibility of environment (e.g. terrain). $\mathbf{d_{user}}$ measures the compatibility of particular user specifications. $\mathbf{k_1}$ and $\mathbf{k_2}$ adjust the relative importance of the terms. The best-fit motion sequence will have the minimal distance according to the above equation.

5 Discussion and Summary

As the availability of motion capture data is increasing, there has been more and more interest in using it as a basis for creating computer animations when life-like motion is desired. The idea of reusing motions is not new. The existing work has already addressed many of the difficulties, especially the need for novel movement generation. Aiming at exploring the novel motion reuse approaches in entertainment industry, we also have developed a Kungfu motion library and a suite of motion editing tools for game development [27, 28]. The implemented system shows promising results in providing a cost-effective approach to quickly build motions for character animation.

In light of reusable motion libraries, its access should be flexible and convenient, and its content should be sufficient and well-structured. For example, the animator can browse a human movement library, instantiating a specific movement by demanding realism and specifying the character dimensions, foot placements, even the emotional state of the character.

Once the candidate motions are chosen, how can they be adapted to precisely fit the current situation? There are no perfect solutions yet. Besides the adaptation operation, we should also perform the cleaning-up of artifacts such as foot–skate from

the newly generated motions [29]. An analysis of advantages and drawbacks of existing motion adaptation approaches are given in table 2.

Table 2. comparison of motion adaptation approaches

Methodology /principles	*Strengths*	*Limitations/difficulties*
Signal processing	It can provide analytic solutions at interactive speed, and lots of existing algorithms in signal processing can be "borrowed" directly.	The signal processing does not explicitly describe the operation in terms of features in motions, and make it difficult for high-level motion editing
Constrained optimization	It can potentially provide high-level motion editing operations, and the user can make the adjustments to an animated character with direct manipulation.	The resulting motions are restricted by the range of constraints and the performance of solvers. The user can not specify "extra" adaptation
Synthesis by statistical models	It structures and models the "essential" part motion capture data, and help make them data to be reused more plausibly.	It synthesizes motion based on abstractions of data rather than actual data, they risk losing important details
Interpolation	It is simple, well-known and easily implemented in key-frame animation	The resultant motions are heavily dependent on the parameters specified by the user and employed interpolation methods.

The tendencies and future work in motion reuse are

- Motion retrieval and synthesis in the designer's way. It is preferred that the animator just roughly draft a motion sequence in an intuitive way such as by video, and then the system help compose the "goal motion" for them. This allows interactive manipulation of the constraints. Higher level stylistic constraints can also be incorporated into the motion synthesis framework. They can be labeled with the intrinsic style by hand, or by learning. For example, Brand & Hertzmann analyzed patterns in motion dataset, and try to infer these styles or obtain higher level descriptions of motions [20].
- A hybrid approach of motion adaptation by controller and example. The animators are expecting the methods to generate both realistic and controllable motion through a database of motion capture. Keyframe animation has the advantage of allowing the animator precise control over the actions of the character. Motion capture provides a complete data set with all the detail of live motion, but the animator does not have the full control over the result. Pullen and Bregler have already made efforts towards combining the strengths of keyframe animation with those of using motion capture data [4].
- Embedding AI technology into the motion reuse. The motion domain knowledge and the human skeleton model can be used to guide and facilitate the reuse of motion capture data. For example, Sun & Metaxas automate the gait generation using human lower-part model [30]. Furthermore, some AI algorithms such as state-space search can be directly applied in the motion reuse. Arikan & Forsyth consider selecting a collection of clips that yields an acceptable motion as a

combinatorial problem, and then use a randomized search of hierarchy of graphs to build new motions [5]. We can also try to build the visual plausible motions by knowledge-based simulation [31].

- High-level motion planning. While a motion database helps define the immediate capabilities of a character, it does not provide a means of ordering the motion primitives to achieve a given artistic or expressive goal. In fact, many actions require planning and anticipation. Planning techniques thus need to be developed to achieve higher-level semantic goals.
- Extension to multi-person scenarios with dynamic external interactions. Up to now, most existing motion adaptation techniques merely deal with the pure motions without any external interaction from other character or environment. However, plenty of application scenarios, such as multi-person combat, are demanding tools to simultaneously edit and direct the motions for multiple characters. Some pioneer work has been done by Zordan and Hodgins now [32].

Acknowledgement

This work is supported by the internal research grant of The Hong Kong Polytechnic University.

References

1. Menache, A.: Understanding motion capture for computer animation and video games", Morgan Kaufman (Academic Press), San Diego, USA (2000)
2. Gleicher, M. and Ferrier, N.: Evaluating Video-Based Motion Capture, Proceedings of Computer Animation 2002, 75-80
3. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs, SIGGRAPH 2002, 473 - 482
4. Pullen, K. and Bregler, C.: Motion Capture Assisted Animation: Texturing and Synthesis, SIGGRAPH 2002, 501-508
5. Arikan, O. and Forsyth, D.A.: Interactive Motion Generation from examples, SIGGRAPH 2002, 483-490
6. Lamouret, A. and van de Panne, M.: Motion Synthesis by Example, 7th Eurograph Workshop on Animation and Simulation, (1996) 199-212
7. Tanso, L. M. and Hilton, A. Realistic Synthesis of Novel Human Movements from a Database of Motion Capture Examples, Proceedings of the workshop on Human Motion 2000, 137-142
8. Jeong, Il-K. , Park, K-J, Baek, S-M, Lee, I.: Implementation of a Motion Editing System, Proceeding of Virtual Systems and Multimedia (VSMM'01), IEEE Computer Society, (2001) 761 - 769
9. Gleicher, M.: Retargeting Motion to New Character, SIGGRAPH'98, 33-42
10. Rose, C., Guenter, B., Bodenheimer, B. Cohen, M. F. Efficient Generation of Motion Transition using Spacetime COnstraints, SIGGRAPH'96, (1996) 147-154
11. Gleicher, M. Motion Path Editing, ACM Symposium on Interactive 3D Graphics'2001, 195-203
12. Bruderlin, A. and Williams, L.: Motion Signal Processing, SIGGRAPH '95, 97 - 104
13. Unuma, M. Anjyo, K., Takeuchi, R.: Fourier Principles for Emotion-based Human Figure Animation, SIGGRAPH'95, 91-96

14. Witkin, A. and Popovic, Z.: Motion Warping, SIGGRAPH'95, 105-108
15. Sudarsky, S and House, D.: Motion Capture Data Manipulation and Reuse via B-spines, in CAPTECH'98, LNAI 1537, Spring-Verlag Berlin Heidelberg, (1998) 55-69
16. Gleicher, M. Motion Editing with Spacetime Constraints, ACM Symposium on Interactive 3D Graphics, (1997) 139-148
17. Lee, J. and Shin, S.-Y. : A Hierarchical Approach to Interactive Motion Editing for Human-like Figures", SIGGRAPH'99, 39-48
18. Gleicher, M.: Comparing Constraint-based Motion Editing Tools, Graphical Models, 63, (2001)107-134
19. Liu, C. K. and Popovic, Z.: Synthesis of complex dynamic character motion from simple animations, SIGGRAPH 2002, 408-416
20. Brand, M. and Hertsmann, A.: Style Machines, SIGGRAPH' 2000, 183-192
21. Li, Y., Wang, T.-S., Shum, H.-Y.: Motion Texture: A two level statistic model for character motion synthesis, SIGGRAPH 2002, 465-472
22 Wiley, D. J., and Hahn, J. K.: Interpolation Synthesis of Articulated Figure Motion, IEEE CG&A, 17(6), (1997)39-45
23. Rose, C., Cohen, M. F., Brodenheimer, B.: Verbs and Adverbs: Multidimensional Motion Interpolation, IEEE CG&A, 18 (5), (1998)32-38
24. Sudarsky, S. and House, D.: An Integrated Approach towards the Representation, Manipulation and Reuse of Pre-recorded Motion, Proceeding of Computer Animation 2000, IEEE Computer Society, (2000) 56-61
25. Wooten, W. L. and Hodgins, J. K.: Transitions between dynamically simulated motions: Leaping, Tumbling, Landing, and Balancing", the visual proceedings of ACM SIGGRAPH'97, August 3-8, Los Angeles, California, USA
26. Bindiganavale, R, and Badler, N. I.: Motion Abstraction and Mapping with Spatial Constraints", CAPTECH'98, LNAI 1537, Spring-Verlag Berlin Heidelberg, (1998) 70-82,
27. Geng, W.-D. Lai, C.-S. Yu, G.: Design of Kungfu library for 3D game development," The 2nd International Conference on Application and Development of Computer Games, Hongkong, (2003) 138-141
28. Geng, W.-D. Chan, M., Lai, C.-S., Yu G. Implementation of runtime motion adjustment in game development, The 2nd International Conference on Application and Development of Computer Games, Hongkong, (2003)142-147
29. Kovar, L., Gleicher, L. Schreiner, J.: Footskate Cleanup for motion capture editing", ACM Symposium on Computer Animation 2002, 97-104
30. Sun, H. C. and Metaxas, D. N.: Automating gait generation, SIGGRAPH 2001, 261-269
31 Barzel,.R. Hughes, J. F., Wood, D. N.: Plausible Motion simulation for Computer Graphics Animation, EUROGRAPHICS workshop on Computer Animation and Simulation, (1996) 183-197
32. Zordan, V. B. and Hodgins, J. K.: Motion capture-driven simulations that hit and react, In Proceedings of ACM SIGGRAPH symposium on Computer Animation, San Antonio, Texas, (2002) 89-96

A Real-Time Natural Motion Edit by the Uniform Posture Map Algorithm

Jin Ok Kim[1], Bum Ro Lee[2], and Chin Hyun Chung[2]

[1] School of Information and Communication Engineering, Sungkyunkwan University, 300, Chunchun-dong, Jangan-gu, Suwon, Kyunggi-do, 440-746, KOREA
jinny@ece.skku.ac.kr
[2] Department of Information and Control Engineering, Kwangwoon University, 447-1, Wolgye-dong, Nowon-gu, Seoul, 139-701, KOREA
chung@daisy.kwangwoon.ac.kr

Abstract. Many researchers have taken the effort to describe the dynamics of the articulated body by the analytic method. They have obtained excellent results in various fields. However, for the articulated body moving with its voluntary will, it is difficult to generalize the motion pattern by analytical modeling, because the motion pattern is extremely subjective and unpredictable. The learning networks overcome the restriction of analytic modeling through the deductive learning method. The Uniform Posture Map (UPM) is proposed to synthesize a new motion between existing clip motions. It is organized through the quantization of various postures with an unsupervised learning algorithm; it places the output neurons with similar postures in adjacent positions. Using this property, an intermediate posture of applied two postures is generated; the generating posture is used as a key-frame to make an interpolating motion. The UPM needs fewer computational costs, in comparison with other motion transition algorithms. It provides a control parameter; an animator can not only control the motion simply by adjusting this parameter, but also produce animation interactively. The UPM prevents the generating of the invalid output neurons to present unreal postures in the learning phase; thus, it makes more realistic motion curves; finally it contributes to the making of more natural motions.

1 Introduction

Through the automation of tedious repetitions, computer systems maximize the efficiency of operations and encourage workers to focus on their jobs. This is also true in the process of digital animation production. While animators had to draw each frame manually in the past, the introduction of the key-frame animation technology has allowed animators to devote more time to their creative work [1] [2] [3]. Since all motion dynamics occurring in animation are decided according experience and intuition, the resulting animation could be extremely subjective. Such a method could be suitable for the exaggerated expressions of character animations, but not for realistic ones [4] [5] [6]. Motion capture technology, however, could solve these problems. The motion-capture system records

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 630–639, 2003.
© Springer-Verlag Berlin Heidelberg 2003

the signal of sensors attached on the articulated body over time. Because the motion data obtained from the motion capture system is a reflection of the actor's dynamics, animation that is more realistic than bey frame animation can be produced. However, although such a motion capture system can record realistic articulated body motions, captured motion data lacks control points to modify motion curves; it is also impossible to control the motion in a consecutively captured frame. Since one has to capture motion separately whenever one needs similar motions with inconsiderable difference, the motion data editing has been studied from various points of view. If one could reuse existing motion data through editing, it is possible to reduce the cost of producing animation while making the production process more efficient. There are various methods of editing the captured motion [7] [8] [9] [10]. One builds a basic motion database composed of short unit motion clips. One selects two clip motions from this database, and then one makes a seamless new motion by connecting two unit motion clips by generating proper intermediate motions. Rose [11] adopted nonlinear programming to make such intermediate motion data. A model with a minimum DOF (Degree of Freedom) is defined by the posture of the articulated body. In the learning phase, one can classify the bones of an entire articulated body into four bone classes according to their properties. The DOF vector is defined as a set of DOF values included in each bone class, and four partial posture maps are generated for each class. In the synthesis phase, we generate four partial postures with four partial posture maps and assemble the partial postures into a complete posture. This posture is used to generate intermediate motion frames by interpolating the B-spline [12]. In comparison with many other heavy computational algorithms, the learning algorithm does not need heavy computational costs; additionally, an animator could control the resulting motion by adjusting only one parameter. Thus, the algorithm provides a real time method for an animator to produce animation interactively. Above all, the algorithm contributes to the making of a more natural motion.

2 The Uniform Posture Map Algorithm

The most popular algorithm, applied to the motion signal processing such as a motion transition and a motion blending, is an optimization method called Non Linear Programming (NLP), Sequential Quadratic Programming, or BFGS. Many researchers have proposed user-interactive motion editing methods with spacetime constraints [13]. However, there are several problems with this approach; one is the computation load problem [1] [14]. In this paper, we use a model with 21 bones; it is a DOF vector with 63 elements. It is a significantly high-order degree. Trying to search the error surface in a 63-dimension vector space and calculating the exact object function in each numerical step during the optimization process, it could be a horrible experience. In general, most practical NLP requires a computation load of $O(n^2)$ and $O(n^3)$; this is a critical problem in real-time-user-interactive systems [14] [15]. Another problem is that NLP always decides its solution to minimize its energy object functions [16].

However, the model, which moves its body by the power of its own muscles, does not always move to minimize energy. In particular, when we want to edit a choreographed motion such as a ballet, the problem would be prominent. The third problem is the convenience of motion control. To control motion easily, the motion transition algorithm has to provide an intuitive control parameter. But NLP does not have such control parameters; thus, it is difficult to control result motions through intuitive interactions. Kohonen's self-organizing map [17] [18] is an unsupervised learning network for solving many problems with the NLP algorithm. It consists of an input layer and an output layer, and organizes a feature map reflecting the feature of input samples in a learning phase. Fig. 1 shows the basic structure of self-organizing maps.

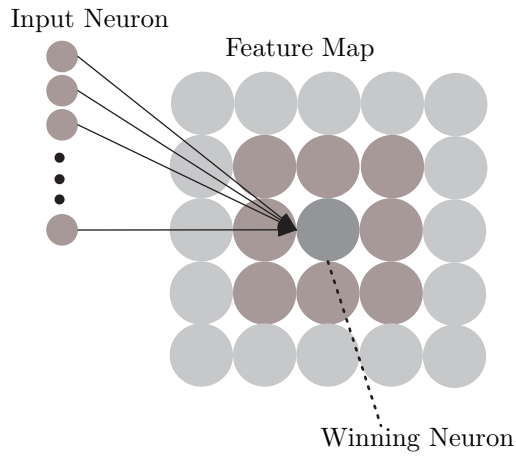


Fig. 1. Self-organizing map.

In general, the self-organizing map reflects a feature of the sample space correctly, and has a relatively faster learning time [19]. But when the sample space has a complex shape, the general learning process cannot generate a proper output neuron. In particular, when the sample space has a non-convex, the part of the output neurons are sometimes formed out of the valid space, because the self-organizing learning algorithm has a tendency of preserving its neighborhood relationships [17] [18]. It is necessary to decide on a lot of parameters to train the self-organizing map. To get a proper parameter set, frequent learning and checking for a generated self-organizing map is required. In actual cases, the method for determining such parameters is not defined as a rule. Therefore, such parameters have to be determined by a tedious trial-and-error process. To solve these problems, we propose a Uniform Posture Map (UPM). We set its initial weights randomly and make a self-organizing map on the basis of these initial weights. For the proposed algorithm, we organize its network topology

on the basis of the information of the sample vector. The only required two parameters to be determined previously are the maximum acceptable radius of an output neuron, and the size of its neighborhood. The following paragraph describes the UPM algorithm;

1. Weight Initialization
   - The maximum radius of output neuron, R, is set properly
   - The initial radius of the neighborhood, N, is set properly.
   - Input sample is set as the weight of the first output neuron.
2. Calculation of the distances between all output neurons and input samples.
   - If all distances exceed the R, a new output sample is inserted into the map; the input sample is set as the weight of a new output.
   - Otherwise, go to step 3.
3. Calculation of Activation
   - The activation X_i of input unit i is determined by the instance presented to the network.
   - The activation O_j of output unit j is calculated by

$$O_j = F_{\min}(d_j) = F_{\min}\left(\sum_i (X_i - W_{ji})^2\right)$$

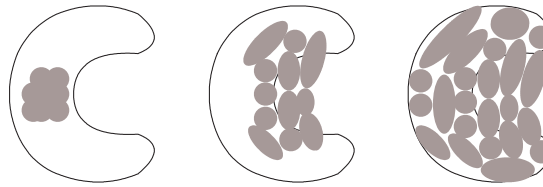
     where $F_{\min}$ is the unity function if unit j is the output node with minimum d_j of its neighborhood, and the zero function otherwise.
4. Weight Training
   - Weight is updated by

$$\Delta W_{ji} = O_j \eta (X_i - W_{ji})$$

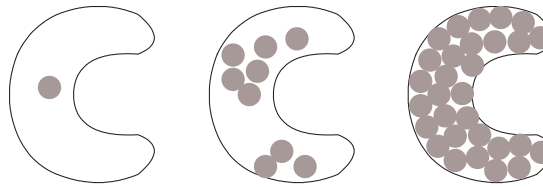
     where η is a gain term $(0 < \eta < 1)$ that decreases over time. The radius of the neighborhood also deceases over time.
   - Repeat by presenting a new instance, calculating activations, and updating weights until $\eta = 0$.

Fig. 2 shows the self-organizing map and uniform posture map having different learning schemes.

As shown in Fig. 2, the UPM does not have the fixed number of output neurons. If there is an input sample with a distance from all output neurons beyond a critical value, it does not vary the existing output neurons, add a new output neuron to its map, and set the weight of new output neurons with the input sample. Therefore it could reserve its validation in a non-convex region, and overcome the drawback of self-organizing maps. It has computational merit in intermediate motion generating because it requires only the computation load, $O(n)$, to calculate a distance of two points and distances between all output neuron and a certain point in the map.



(a) The learning of self-organizing map



(b) The learning of uniform posture map

Fig. 2. Different learning scheme of two methods.

3 Motion Synthesis

When the composition of the intermediate posture has been completed, we accomplish a motion synthesis. First of all, we have to consider the root component in the entire motion transition. Because the posture map lacks information about the movement of the root, we have to determine the trajectory and rotation of the root during the motion transition. The root has 6 DOFs which consist of three translational DOFs and three rotational DOFs. The rotational DOF is interpolated by linear interpolation and B-spline. We define the translational variation of the y coordinate of the root as such, because the y component of the root has a trivial variance in most general motions. We adopt an algorithm proposed by Rose [11] [15] and amend it for the goals to determine the x and z coordinates. We define a function for the x and z coordinates.

$$f(t) = \begin{cases} t < t_1, & p(t) = l(t) \\ t_1 \leq t \leq t_2, & p(t) = p(t_1) + m(t) \\ t_2 < t, & p(t) = p(t_2) + n(t) \end{cases}$$

Where $l(t)$ is the root trajectory of the first motion,

$$m(t) = \sum_{\alpha=t_1}^{t} \left\{ v_1 \left(1 - \frac{\alpha - t_1}{t_2 - t_1} \right) + v_2 \left(\frac{\alpha - t_1}{t_2 - t_1} \right) \right\}$$

$$n(t) = \sum_{\alpha=t_2}^{t} v_2(\alpha - t_2),$$

and t_1 is a time, concerning the end frame of first motion, t_2 concerns the first frame of second motion, and $p(t_1), p(t_2)$ are the coordinates at time t_1, t_2. The foot location of the model must be satisfactory for some physically restricted conditions.

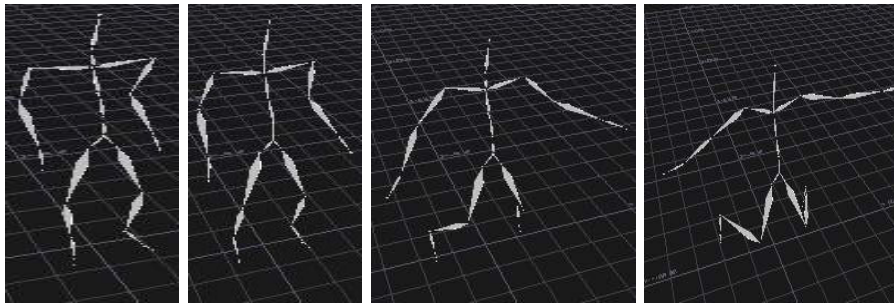
4 Experiment and Result

4.1 Experiment

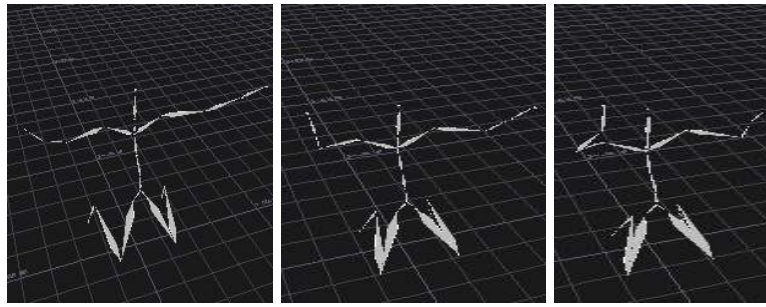
The motion transition system is composed of three separate phases: the data preprocess phase, the map learning phase and the motion synthesis phase. In the data pre-processing phase, we first gather human motion-capture data with the optical motion-capture system. The data consists of the 3D positions of optical markers attached on human actors, acquired over short intervals. We map the 3D positions into the DOF of the bone set. Next, we modulate the motion data to set bounds for the angular range from 0 to 2π, and reduce some redundant DOFs by a manual data modification. In the learning phase, we divide the motion data by their property into four classes: the body, the global, the medium and the detail. We make four uniform posture maps by the UPM algorithm. Once we make the posture maps, we need not build another posture map in the next processing phase and can reuse the posture map. In the motion synthesis phase, we synthesize an intermediate posture containing the feature of two postures simultaneously: the first posture of the second motion, and the last posture of the first motion. With a generated intermediate posture, we make a new motion by approximation of the B-spline. The optical motion capture system of VICON® is used to get the experimental motion data. The FilmBox® is used to edit model data to reduce DOFs manually, and to render the resulting character animation. We make a nine processed sample motion clips with 1800 frames.

4.2 Result

We have applied the UPM to various motion clips and demonstrated it with two motion clip pairs in this paper: two dance motion clips, and two cocktail blending motion clips. We set 20 frames as transition intervals and insert the intermediate posture frame, made by the UPM, to the tenth frame.



(a) 195th frame (b) 200th frame (c) 205th frame (d) 210th frame

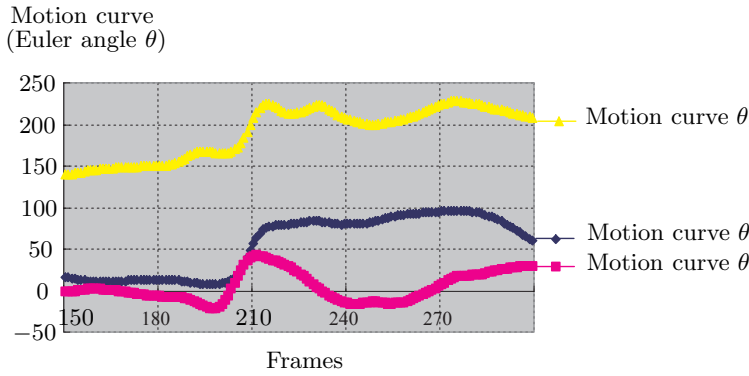


(e) 215th frame (f) 220th frame (g) 225th frame

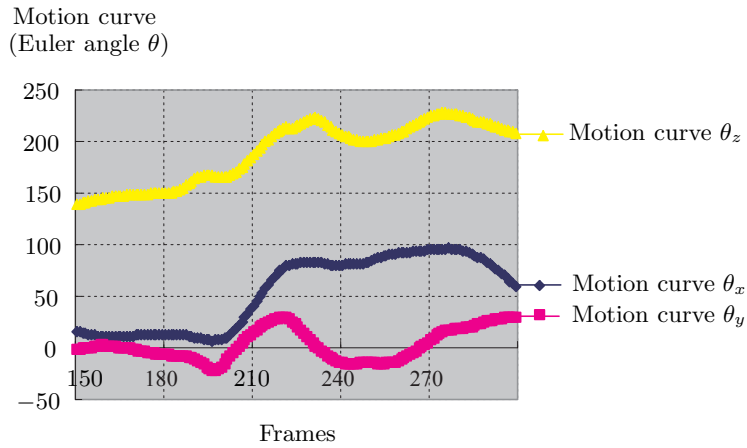
Fig. 3. Motion transition – Dance.

In Fig. 3, we display the motion transition result with dancing motion clips. Each frame is extracted from 5 frames. The frames, used in a motion transition algorithm as an input, are shown in Fig. 3(b) and Fig. 3(f), and the synthesized intermediate posture frame is shown in Fig. 3(d). In an interpolated motion, because each posture is generated on the basis of real posture, most of the motion has a realistic posture. We can notice that the generated intermediate posture is not based on simple linear interpolation.

Fig. 4 shows the comparison of two motions curving over the entire frame range. Fig. 4(b) shows that the curves in the transition intervals form a direct line when a linear interpolation method is used in the motion transition. This could generate rough motion curves over the entire range, resulting in an unnatural motion. However, because the UPM generates smooth motion curves and the motion curves do not secede from the valid posture space, it could guarantee smooth and natural motion transition.



(a) Interpolation using UPM



(b) Linear Interpolation

Fig. 4. Comparison motion curves of over entire range.

In general, computation times for the motion transitions are strongly dependent on the number of DOFs in the involved model. Rose [15] showed that the motion transition of the 44-DOF model generating the spacetime transition motion took 72 seconds in Table 1. However, we have generated 20 intermediate frames in a second. Because we have used a much higher dimension model with 63 DOFs, this is enough to prove the merits of the motion transition algorithm

Table 1. Comparison of the computing time

Method	CPU	DOF	Processing Time
UPM	Pentium 133MHz	63	1.5 sec
Spacetime	Pentium 100MHz	44	72 sec

in terms of motion generation speed. We have applied the algorithms to many other motion clips, and we have had satisfactory results. Most results have shown a seamless natural motion; most of the motions have observed physical laws.

5 Conclusion

It is important to reuse existing motion capture data for reduction of the animation producing costs as well as for the efficiency of the producing process. Because its motion curve has no control point, however, captured data is difficult to modify interactively. Motion transition is a useful method for reusing existing motion data. It generates a seamless intermediate motion with two short motion sequences. In this paper, the UPM is proposed to perform motion transitions. It requires only the computational load, $O(n)$, to generate intermediate motion curves, which, in turn, make animators generate intermediate motions in real-time. Additionally, it provides an intuitive control parameter: an animator can control motion simply by adjusting this parameter. This property of the UPM makes an animator produce animation interactively. Above all, the algorithm contributes to making more natural motions. The UPM prevents unreal posture from generating in learning time; therefore, the new generated motion never exhibits unnatural behavior. This is a very important superior feature of the algorithm. In order to make motions natural, other algorithms have to adopt more physical constraints that require more computational costs. In addition, the UPM could be applied to various fields such as real time 3D games, virtual reality applications, and web 3D applications.

References

1. Brand, M., Hertzmann, A.: Style machine. In: Proc. of SIGGRAPH 2000, ACM Press (2000) 183–192
2. Sun, H.C., Metaxas, D.N.: Automating gait generation. In: Proceedings of SIGGRAPH 2001, ACM Press (2001) 261–270
3. Shin, H.J., Lee, J., Shin, S.Y., Gleicher, M.: Computer puppetry. ACM Transactions on Graphics **20** (2001) 67–94
4. Popović, Z.: Controlling physics in realistic character animation. Communications of the ACM **43** (2000) 50–58

5. Noser, H., Thalmann, D.: A rule-based interactive behavioral animation system for humanoids. IEEE Transactions on Visualization and Computer Graphics **5** (1999) 281–307
6. Sannier, G., Balcisoy, S., Thalmann, N.M., Thalmann, D.: A system for directing real-time virtual actors. The Visual Computer **15** (1999) 320–329
7. Jung, S.K.: Motion Analysis of Articulated Objects for Optical Motion Capture. Ph. D. dissertation, KAIST (1997) Dept. of Computer Science.
8. Wiley, D.J., Hahn, J.K.: Interpolation synthesis of articulated figure motion. IEEE Computer Graphics and Applications **17** (1997) 39–45
9. Brudelin, A., Williams, L.: Motion signal processing. In: Proc. of SIGGRAPH 95, ACM Press (1995) 97–104
10. Witkin, A., Popović, Z.: Motion warping. In: Proc. of SIGGRAPH 95, ACM Press (1995) 105–108
11. Rose, C., Bodenheimer, B., Cohen, M.F.: Verbs and adverbs: Multidimensional motion interpolation. IEEE Computer Graphics and Applications **18** (1998) 32–40
12. Zhao, J., Badler, N.I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. ACM Transactions on Graphics **13** (1994) 313–336
13. Gleicher, M.: Motion editing with spacetime constraints. In: Proc. of Symposium on Interactive 3D Graphics. (1997) 139–148
14. Grzeszczuk, R., Terzopoulos, D., Hinton, G.: Neuro animator: Fast neural emulation and control of physics-based models. In: Proc. of SIGGRAPH 98, ACM Press (1998) 9–20
15. Rose, C., Guenter, B., Bodenheimer, B., Cohen, M.F.: Efficient generation of motion transitions using spacetime constraints. In: Proc. of SIGGRAPH 96, ACM Press (1996) 147–154
16. Cohen, M.F.: Interactive spacetime control for animation. In: Proc. of SIGGRAPH 92, ACM Press (1992) 293–302
17. Ritter, H., Martinetz, T., Schulten, K.: Neural Computation and Self-Organizing Maps. Addition Wesley, New York (1992)
18. Gallant, S.I.: Neural Network Learning and Expert System. MIT Press (1993)
19. Baraldi, A., Alpaydin, E.: Constructive feedforward art clustering networks. IEEE Transactions on Neural Networks **13** (2002) 662–677

A Feature-Preserved Simplification for Autonomous Facial Animation from 3D Scan Data

Soo-Kyun Kim, Sun-Jeong Kim, and Chang-Hun Kim

Dept. of Computer Science and Engineering, Korea University,
1, 5-ka, Anam-dong, Sungbuk-ku, Seoul 136-701, Korea
{nicesk, chkim, sunjeongkim}@korea.ac.kr

Abstract. We propose a new simplification algorithm of facial models for animation. For the facial animation, the models are often simplified from complex scan data based on geometric features, but it leads to decrease the quality and such features are easily noticed by human perception. For example, a lip line and eyebrows easily lose their details by geometry-based simplification. In this paper, facial features are extracted using an image processing of a 2D texture image and the curvature analysis of the 3D geometry, which improves the details around the feature areas of the facial model. Especially if lip contact line is simplified to one or two edges, it may not be proper for lip animation. Finally, we will show that our simplified model can produce as good as a facial animation as the one from the original model.

1 Introduction

An individual head model is usually reconstructed from the scan data acquired from an optical scanning system. Since a scan data is usually large in size, it needs to be simplified, and there are various mesh simplification algorithms to do such a task. However, most of these algorithms are not suitable for facial modeling because they do not take into account the features of a human face. For example, figure 2 shows the facial animation of a model, which is simplified by a geometry-based simplification, that exhibits a poor quality near features like a lip and eyes, although it is simplified to the same number of vertices as that of Figure 1.

In real time or interactive applications, models with millions of polygons are still burdensome even with fast graphics hardware. For this reason, simplification of surfaces has been the subject of a great deal of research. Simplification algorithms [5,9,12,16] based on iterative edge contraction have gained a lot of attention. Since the new vertex position can be controlled to retain the original appearance, retriangulation is not needed.

Ronfard and Rossignac [15] measured the error at a vertex by maximum distance between the vertex and the planes. Instead of the maximum distance, Garland and Heckbert [4,5] used the sum of the squared distances, along with

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 640–649, 2003.
© Springer-Verlag Berlin Heidelberg 2003

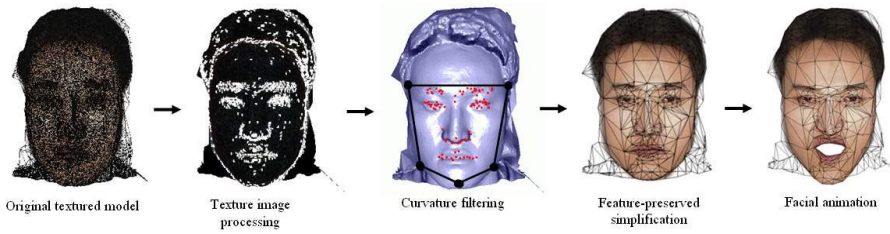


Fig. 1. A brief overview of our simplification process: The input model is texture-mapped scan data; at first, facial features are extracted from texture image; then final features are remained through curvature filtering within bounding area; after our feature-preserved simplification, the animation of the simplified facial model shows higher quality than Figure 2.

a memory-computation efficient algorithm called "Quadric Error Metrics" to accumulate the cost of contraction at an object vertex as the simplification progresses. Garland [5] and Hoppe [9] uses extended error metric that combines geometric and surface attributes (normal, colors, texture coordinate). Cohen [1] did not use a quadric form but proposed a texture error measurement algorithm. Discrete curvature [2,3] approximated from a geometric reasoning is useful to enhance the shape description of triangular surfaces. Therefore, discrete curvature can be used as a good criterion of simplification that can preserve the shape of an original model well as in [11].

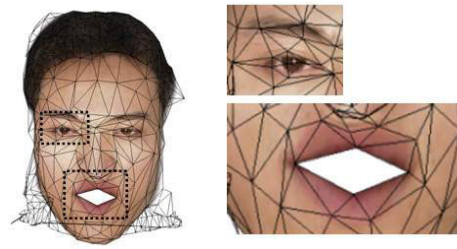


Fig. 2. Low quality of facial animation of a simplified facial model by the previous method [5]. (left)Animation, (right)Close-up of dashed boxes in (left).

Several methods were also proposed to find feature points in meshes by applying image processing techniques to 3D mesh. Guskov [7] applied geometric filter for 3D meshes to enhance the representation of high frequency region. Hubeli [10] proposed a method to find feature edges in a 3D mesh that gives edges weights based on difference value between two normals and then fits polynomials to intersection line between parameter plane and mesh. A drawback of

this method is that users must select a threshold value that cannot be found intuitively. Besides, both of them are not satisfactory in terms of the time spent to extract feature points from 3D mesh.

Facial animation [13,14,17,18] can be embodied by various methods, among which we apply one of performance-driven animations, expression cloning to our simplified results. Expression cloning [14] is a method to retarget facial expression animations to unlike models. So to speak, it adapts approximate animation of a model to another. It enables users to apply the same animation to various models. The anatomical knowledge of faces are not needed, because this technique uses motion vectors.

In this paper, we propose a different approach to model a coarse but feature-preserved head model for facial animation. As we told before, a simplified facial model must be proper to be animated. At first, conspicuous pixels are extracted from a texture image using image processing technique, because those features such as lip line and eyebrows are easy to lose their details by geometry-based simplification. Those pixels find the corresponding vertices by simple parameterization, and then among those vertices the feature points of a facial model are selected through discrete curvature filtering within the bounding area, which is automatically assigned based on pre-knowledge of facial animation. At last, giving weight to one-neighbor vertices of feature points, our algorithm simplifies a facial model preserving features. The proposed method is comparatively simple and easy, and helps to control the quality of a coarse facial mesh generated by feature-preserved simplification.

2 Feature Extraction

The goal of our framework is to extract feature parts in a head scan data, simplify the model with preserving feature points, and to animate the result mesh. Here, we define feature parts and feature points in the face, which include eyes, eyebrows, a nose and a mouth. It is important to simplify them, still keeping their features. Feature points are defined as points that stand out in a facial model such as sharp or ridged points.

2.1 Extraction of Feature Points from a Texture Image

Range scanning system provides a texture image as well as a geometry data. Previous methods have focused on geometric information in order to find feature points and edges, but they failed to obtain superior results and satisfying implementation time. Our method enables us to detect contour edges from texture information and to use them easily.

Edge detection is a commonly used operation which reinforces and extracts edges of images. An edge is a boundary line where images are bordered, or overlapped with the background. We call them edge parts including changes of chroma or gray scale in image. While the Sobel, Prewitt, and Canny algorithms

[6] are commonly used for edge detection, but each algorithm has both advantages and disadvantages. Most frequently used the Sobel edge detection is more sensitive to diagonal lines in edges than to horizontal or vertical lines. The Prewitt appears to have the same result value as the Sobel, but has faster response time. It also exhibits a similar form of a mask to that of the Sobel edge detection but gives a different weight to boundaries of brightness. Given different weight value of mask, the contour edges are embossed. Due to its fast response time and better detection of contours, we employ the Prewitt algorithm in our implementation.

We assign threshold value on R, G, and B to extract prominent pixels by Prewitt edge detection. R, G and B have advantages and disadvantages respectively. One of the most efficient ways is to find all feature points by union of things found from each channel. This approach requires a user-defined value. We assign different threshold values on R, G, and B respectively, and extract the suitable contour edges. Following equation is to assign whether a pixel (u_j, v_j) is a feature or not:

$$\text{isFeature}(u_j,v_j) := ((r_{(u_j,v_j)} \geq R_{threshold}) \| (g_{(u_j,v_j)} \geq G_{threshold}) \| (b_{(u_j,v_j)} \geq B_{threshold}))$$

In the right figure of Figure 3, white marks indicate that feature edges are detected and black ones the opposite.

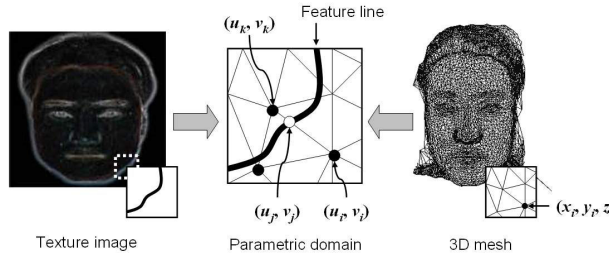


Fig. 3. Simple parameterization of 3D mesh (right) on the texture map (left) using texture coordinates.

After the edge detection process, we must find the 3D coordinates corresponding to the feature edges. An intuitive method is to parameterize 3D geometric coordinates on a texture map with respect to their texture coordinates. In Figure 3, the vertex coordinate (x_i, y_i, z_i) is parameterized on the texture image using the texture coordinate (u_i, v_i). Sometimes pixels of the part marked white are not always on the vertex. The feature pixel (u_j, v_j) is on the edge in Figure 3. In this case, a new vertex has to be inserted precisely to preserve the exact feature line. However, simplification algorithm does not allow the increse of the number of vertices or faces. Therefore, we choose another way. The feature pixels find the closest texture coordinates of a vertex, and then its corresponding vertex coordinates is assigned as a feature vertex. To preserve feature lines, our simplification

algorithm puts more weight to both feature vertices and their one-neighbors. Equation (1) is for a marked pixel (u_j, v_j) to find the closest vertex's texture coordinates (u_k, v_k).

$$\text{theClosestVertex}(u_j,v_j)=\min_k \sqrt{(u_j-v_k)^2+(v_j-v_k)^2} \quad (1)$$

Therefore, because of the marked pixel (u_j, v_j), the vertex (x_k, y_k, z_k) is assigned as a feature point, and then both this vertex and its one-neighbors get more weight for simplification to preserve features in Figure 3.

3 Discrete Curvature Operator

The points extracted from a texture image can be classified as two groups. One contains points located on the important part and the other points that is not important or noise. Here, we use feature points in a texture image and find the best facial feature such as eye, nose, and mouth using discrete curvature operator.

3.1 Bounding Area on a Facial Model

Feature points should consist of only feature parts in the facial model. Hence, we make a bounding area in the facial model, and remove points located on the outer wall of the bounding area.

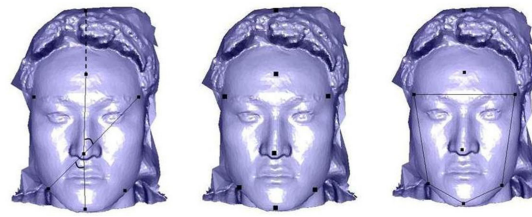


Fig. 4. Generation of a bounding area.

Firstly, the local axes of the model are fitted to the world axes to find two points in the facial model. The head top is positive y-direction, the right ear x-axis, and the nose tip z-axis. The head top is the vertex of the highest y-coordinate. The nose tip is the vertex of the highest z-coordinate. So we find two points (the head top and the nose tip vertex). Next, we use the two points found above in the facial proportional method (The facial proportional method is acquired by analyzing [8]) and find the necessary points for making a bounding area. Since six points found by the facial proportional method are not the points from real coordinate, we find the points with the shortest distance through equation (1). The following describes the facial proportional method:

Step 1 **Find a point on forehead:** a point on the forehead can be found using a center point of the line segment connecting the points of the head top and of the nose tip. (We find points that become 1.5 times the length when the facial model has hair.) See Figure 4 (left).

Step 2 **Find a point between nose tip and jaw top:** we find the distance from the point on the forehead to the nose tip, and the distance is acquired between the nose tip and the jaw top. The point of the jaw top can be located on 2/3 of the distance. See Figure 4 (left)

Step 3 **Find end points of forehead:** we connect a point on forehead to the nose tip, connect a line segment, and rotate it via the nose tip and find the end points of the forehead. The rotation is based on the following formula (Upper line: about 30 degrees, Lower line: about 35 degrees, these rotation degrees are experimental results). But there is no consideration for z-axis; it can find a point in the back of the head. We find a point of the nose tip and a point of the back of the head, get the average of the two points and find a side for positive direction. See Figure 4 (left) and (center).

Step 4 **Find end points of the jaw:** connect a point on the jaw top to the nose tip, connect a line segment, and rotate it via the nose tip and find the end points of the jaw.

Step 5 **Generate a bounding area:** we generate a bounding area that connects the five points. See Figure 4 (right).

3.2 Curvature Filtering

After removing the outer points of a bounding area, remaining points are either feature or noise points. We remove noise points, and thin down the feature points. To do this, we compute the mean curvature of the points extracted from a texture image. To speed up the later computation, we selectively choose some points among the feature points, called curvature filtering. First, we remove the small curvature of points in the mean curvature sense. The equation for computing the mean curvature of a point v extracted from a texture image is as following:

$$curvature_sum = \sum_{i \in N_v} \frac{\frac{1}{2} \sum e_i d_i}{\frac{1}{3} A} \tag{2}$$

where N_v is a set of one-neighbor vertices of the feature point v, e_i is an edge of a vertex, dihedral angle d_i is an angle between two adjacent surface normals, A is the sum of areas of one-neighbor faces $A = \sum_{i \in N_v} Area_i$.

We compute curvature about extracted points from a texture, divide the number of points extracted from texture and get *curvature average* after computing, and remove vertices below the *curvature average*, because they do not form feature marks in the facial model. We define that the feature part with bigger curvature in the facial model. We extract feature points by finding the contour of image and these points are computed by curvature operator. We find

feature points using curvature operator. The following is the pseudo-code of the filtering process described above:

```
void extract from a image feature points (list){
curvature average = curvature_sum/#of extractedpointsfromtexture
  for all extracted vertices from a texture image
    if(curvature(v)>curvature average){
       v1 = vertex.neighbor(v);
       v2 = vertex.neighbor(v1);

       if(v1 == v2)
         vertexlist.insert(v1);
       else
         remove from feature points(v);
     }
}
```

4 Feature-Preserved Simplification

This method is based on QEM (Quadric Error Metrics) [5], which uses the iterative edge contraction. The QEM defines a quadric $Q^f(\mathbf{v})$ on each face f of the original mesh, which equals to the squared distance of a point $\mathbf{v} = (p) \in \mathbf{R}^3$ to the plane containing it. For each vertex v of the original mesh, the sum of quadrics on its adjacent faces weighted by face area is assigned, After each edge contraction $(v_1, v_2) \rightarrow \overline{v}$, the position of new vertex $\overline{v}$ is assigned by minimizing the quadric, and then the next edge contraction is chosen as the one with the lowest such minimum. If a vertex is an auto-detected feature vertex, then we give larger weight onto its squared distance to prevent removing such a feature vertex, which is important for facial animation and may have a small value in quadric error. A feature point $(\mathbf{v}_i)$ and its neighbor vertex $(\mathbf{v}_j)$. When an edge (v_1, v_2) is contracted with associated quadrics error $Q^{\overline{v}} = w_{v_1} Q^{v_1} + w_{v_2} Q^{v_2}$, the position of $\overline{\mathbf{v}}$ is selected to minimize $w_{\overline{v}} Q^f(\overline{\mathbf{v}})$.



Fig. 5. Feature-preserving simplification. (left)Curvature filtering, (center)Geometric model, (right)Textured model.

We must consider texture mapping during mesh simplification, because most facial model has a texture image. Our method can achieve good simplified facial mesh considering a texture image and preserving feature points. Before the total error metric **e** is calculated, the weight term of the texture attribute is added to $Q^v(\mathbf{v}_j)$. The extended QEM defines a point $\mathbf{v} = (p) \in \mathbf{R}^5$.

$$\mathbf{e}_i(\mathbf{v} \in \mathbf{R}^5) = w_i Q^v(\mathbf{v}_i) + \sum_{j \in i} w_j Q^v(\mathbf{v}_j) \tag{3}$$

Figure 5 (left) is the result that uses curvature filtering and removes feature and noise points, curvature filtered feature and its neighbor points. (center) and (right) are the results that use 3 and simplify feature-preserved model.

5 Results and Applications

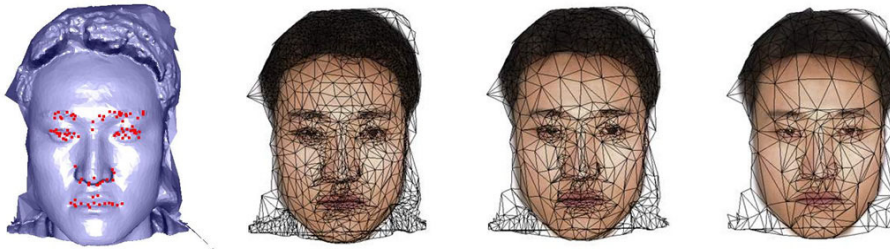
We tested our algorithm on a PC with a 1.7GHz Pentium 4 processor. The results are shown in Figure 6, which are images from Mr. Shim model. You can see that the feature-preserved simplification can improve the quality of the model better than the QEM-based simplification. Table 1 lists the data size and execution times. In addition, we simplify the model and use Expression Cloning for more effective animation. Expression Cloning [14] is a technique to retarget a model's expression to other models'. In our experiment, opening the mouth was challenging because there is no point on the lips in the scan data due to the occlusion. Therefore, we made additional points around the mouth to generate lip animation. As you can see in the Figure 7, feature-preserved simplification works even more effectively in the facial animation. Figure 7 shows images made by the proposed method and they present a superior animation quality of a simplified model.

6 Conclusion and Future Work

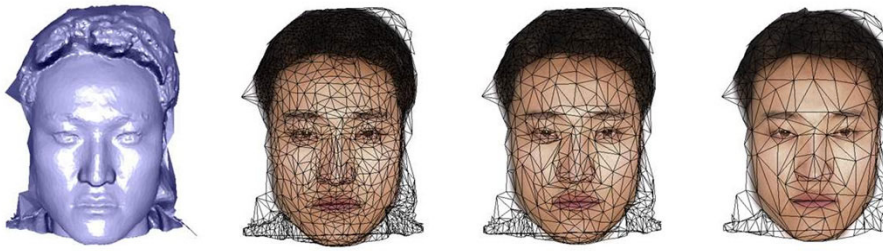
We have presented a new method to create a coarse head model from a dense scan data by an image-based feature detection technique and the feature-preserved simplification. It is a hard and time-consuming process to find feature points and edges automatically from geometric models. But this method is fast, easily extracts a geometric model's feature points from a texture image, and simplifies the facial model while preserving feature points. In addition, this method can be used in mesh editing, facial animation, and so on.

Despite the advantages mentioned, we could not find exact the boundaries of lips when filtering the feature points extracted from an image using discrete curvature, which need to be studied as a future work. Designing a progressive scheme for generating feature points based on a facial mesh is another challenge.

Acknowledgments. This study was supported by a grant of the Korea Heath 21 R & D Project, Minisry of Health & Welfare, Republic of Korea. (02-PJ1-PG1-21400-0001)

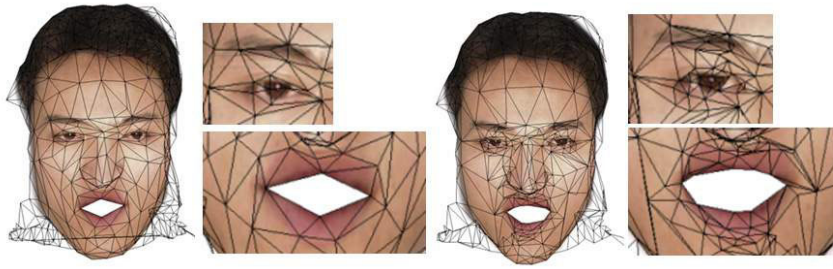


(a) Feature Points (b) Feature-preserved simplification

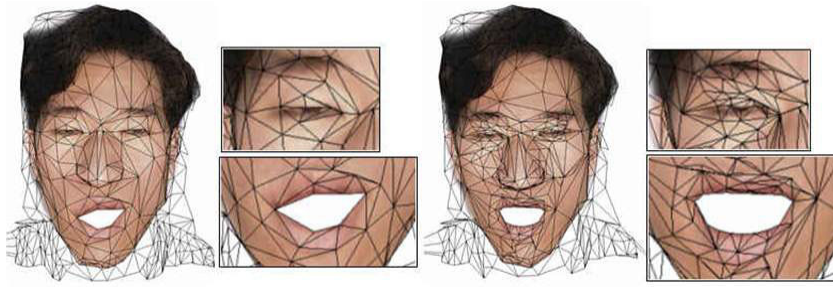


(c) Original Model (d) QEM-based simplification

Fig. 6. Comparison between QEM and Feature-preserved simplification(Left to right faces: 10,000, 3,000, 1,500, 600).



(a) Simplified model animation using the QEM (b) the proposed method



(c) Simplified model animation using the QEM (d) the proposed method

Fig. 7. Comparison of facial animation.The face model mesh consists of 600 triangles

Table 1. Data and Running Time (sec.)

Number	Mr. Shim.	Mr. Choi.
Input # of faces	10,000	10,000
Extracted points from texture	0.0018s	0.00191s
Curvature filtering	0.047s	0.053s
Feature-Preserved Simplification	2.11s/600faces	2.09s/600faces

References

1. J. Cohen, M. Olano, and D. Manocha. Appearance-Preserving Simplification. SIGGRAPH'98 Proceedings. pp. 115-122. 1998.
2. M. Desbrun, M. Meyer, P. Schroder, and A. Barr. Discrete Differential Geometry Operators for Triangulated 2-manifolds. VisMath'02, Berlin.
3. N. Dyn, K. Hormann, S.-J. Kim and D.Levin. Optimizing 3D Triangulations Using Discrete Curvature Analysis. Mathematical Methods for Curves and Surface, pages 135-146, 2001.
4. M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. SIGGRAPH'97 Proceedings, pp.209-216, 1997.
5. M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. IEEE Visualization 98, pp.263-269
6. R. Gonzalez and R.Woods. Digital Image Processing, Addison Wesley, 1992, pp. 414 - 428.
7. I. Guskov, W. Sweldens, and P. Schroder. Multiresolution signal processing for meshes. SIGGRAPH '99, pp325-334, 1999.
8. B. Hogarth. Drawing the Human Head. Watson-Guptill, 1989.
9. H. Hoppe. New Quadric Metric for simplifying Meshes with Appearance Attributes. IEEE Visualization 99, pp.59-66, October 1999.
10. A. Hubeli and M. Gross. Multiresolution Feature Extraction from Unstructured Meshes. IEEE Visualization'01 Conference Proceedings, 2001.
11. S.-J. Kim, S.-K. Kim, and C.-H. Kim. Discrete Differential Error Metric For Surface Simplification. In Proceedings of Pacific Graphics 2002, pp. 276-283, October 2002
12. P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. IEEE Visualization'98 Conference Proceedings, pp.279-286, 1998.
13. Y.C.Lee, D.Terzopoulos, and K.Waters. Realistic Face Modeling for Animation. SIGGRAPH' 95 Proceedings, pp.55-62, 1995.
14. J. Y. Noh and U Neumann. Expression Cloning. SIGGRAPH 2001 Proceedings, pp.277-288, 2001.
15. R. Ronfard and J. Rossignac. Full-range 3D approximations for rendering complex scenes. Modeling in Computer Graphics Forum, Eurographics '96 Proceedings, Vol.15, pp.67-76, 1996.
16. W. J. Schroeder, J.A. Zarge and W. E. Lorensen. Decimation of triangle meshes. SIGGRAPH '92, Proceedings, pp.65-70, 1992.
17. F. Ulgen. A Step Toward Universal Facial Animation via Volume Morphing. 6th IEEE International Workshop on Robot and Human communication, pp.358-363, 1997.
18. K. Waters and J. Frisbie. A Coordinated Muscle Model for Speech Animation. Graphics Interface, pp.163-170, 1995.

Human Motion Tracking by Combining View-Based and Model-Based Methods for Monocular Video Sequences*

Jihun Park, Sangho Park, and J.K. Aggarwal

[1] Department of Computer Engineering
Hongik University
Seoul, Korea
jhpark@hongik.ac.kr
[2] Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
sangho@ece.utexas.edu
[3] Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
aggarwaljk@mail.utexas.edu

Abstract. Reliable tracking of moving humans is essential to motion estimation, video surveillance and human-computer interface. This paper presents a new approach to human motion tracking that combines view-based and model-based techniques. Monocular color video is processed at both pixel level and object level. At the pixel level, a Gaussian mixture model is used to train and classify individual pixel colors. At the object level, a 3D human body model projected on a 2D image plane is used to fit the image data. Our method does not use inverse kinematics due to the singularity problem. While many others use stochastic sampling for model-based motion tracking, our method is purely dependent on parameter optimization. We convert the human motion tracking problem into a parameter optimization problem. A cost function for parameter optimization is used to estimate the degree of the overlapping between the foreground input image silhouette and a projected 3D model body silhouette. The overlapping is computed using computational geometry by converting a set of pixels from the image domain to a polygon in the real projection plane domain. Our method is used to recognize various human motions. Motion tracking results from video sequences are very encouraging.

* This work was partially supported by grant No. 2000-2-30400-011-1 from the Korea Science and Engineering Foundation. We thank Ms. Debi Prather for proofreading of this paper.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 650–659, 2003.
© Springer-Verlag Berlin Heidelberg 2003

1 Introduction and Related Work

Reliable tracking of moving humans is essential to motion estimation, video surveillance and human-computer interface. Tracking non-rigid objects such as moving humans presents several difficulties for computer analysis. Problems include segmentation of the human body into meaningful body parts, handling occlusion, and tracking the body parts along the image sequence. The approaches that have been proposed for tracking a human body can be classified into two groups: model-based approaches and view-based approaches. Refer [1,6] for reviews. Model-based approaches use *a priori* models explicitly defined in terms of kinematics and dynamics. View-based approaches use heuristic assumptions when no *a priori* model is available. These two approaches can be combined at various levels to increase efficiency [15]. This paper presents a new approach to human motion tracking that combines techniques from both view-based and model-based approaches. The proposed system processes the input image sequence at both pixel level and semantic object level. At the pixel level, a Gaussian mixture model is used to classify individual pixels into several color classes, which are merged into coherent blobs by relaxation labeling. At the object level, a 3D human body model projected to the 2D image plane is used to fit the image data. The view-based processing at the pixel level efficiently reduces the overhead in model-based processing at the object level by providing foreground silhouettes.

All kinematics-based motion tracking methods may be classified into two groups depending the use of inverse kinematics – computing joint angles given end-tip kinematic parameters. If no inverse kinematics is used, we call that approach a *forward kinematics-based approach*; otherwise we call it an *inverse kinematics-based approach*. Our work is forward kinematics-based, while the papers reviewed here [8,12] are inverse kinematics-based.

Morris and Rehg [12] presented one of the first model-based methods for deriving differential inverse kinematics equations for image overlapping, although differential kinematics originated from robotics [5]. Morris and Rehg [12] used a 2D scaled prismatic model for figure fitting, and reduced the singularity problem by working in the 2D projection plane. But singularity is inevitable because this method is based on differential inverse kinematics. Huang, et al., [8] extended the inverse kinematics work presented in [12] to solve motion parameters of the articulated body in a statistical framework using the expectation-maximization (EM) algorithm. Sidenbladh et al. [17] converted the human motion tracking problem into a probabilistic inference problem aimed at estimating the posterior probability of body model parameters given an input image.

The rest of the paper is organized as follows: Section 2 describes the procedure at the pixel level and describes the blob formation. Section 3 presents the human body modeling and forward kinematics-based cost function for fitting. Section 4 describes the issue of kinematics and singularity, explaining why we use a forward kinematics-based approach to avoid the singularity problem. Results and conclusions follow in Section 5.

2 Pixel Classification

2.1 Color Representation and Background Subtraction

Most color cameras provide an RGB (red, green, blue) signal. The RGB color space is, however, not effective for human visual perception of color and brightness. Here, the RGB color space is transformed to the HSV (hue, saturation, value) color space to make the intensity or brightness explicit and independent of the chromaticity.

Background subtraction is performed in each frame to segment the foreground image region. Refer to [13] for details. The color distribution of each pixel $v(x, y)$ at image coordinate (x, y) is modeled as a Gaussian. Using k_b training frames ($k_b = 20$), the mean $\mu(x, y)$ and standard deviation $\sigma(x, y)$ of each color channel is calculated at every pixel location (x, y). Foreground segregation is performed for every pixel $v(x, y)$, by using a simple background model, as follows: at each image pixel (x, y) of a given input frame, the change in pixel intensity is evaluated by computing the Mahalanobis distance from the Gaussian background model.

$$\delta = \frac{|v(x, y) - \mu(x, y)|}{\sigma(x, y)} \tag{1}$$

The foreground image $F(x, y)$ is defined by the maximum of the three distance measures, δ_H, δ_H, and δ_V for the H, S, V channels;

$$F(x, y) = max[\delta_H(x, y), \delta_S(x, y), \delta_V(x, y)] \tag{2}$$

F is then thresholded to make a binary mask image. At this stage, morphological operations are performed as a post-processing step to remove small regions of noise pixels.

2.2 Gaussian Mixture Modeling for Color Distribution

In HSV space, the color values of a pixel at location (x, y) are represented by a random variable $v = [H, S, V]^t$ with the vector dimension $d = 3$. According to the method in [13], color distribution of a foreground pixel v is modeled as a mixture of C_0 Gaussians weighted by prior probability $P(\omega_r)$, given by;

$$p(v) = \Sigma_{r=1}^{C_0} p(v|\omega_r) P(\omega_r) \tag{3}$$

where the r-th conditional probability is assumed as a Gaussian, as follows:

$$p(v|\omega_r) = (2\pi)^{-d/2} |\Sigma|^{-1/2} exp[-\frac{(v - \mu_r)^t \Sigma^{-1} (v - \mu_r)}{2}], r = 1, ..., C_0 \tag{4}$$

Each Gaussian component is represented by the prior probability $P(\omega_r)$ of the r-th color class ω_r, a mean vector μ_r of the pixel color component, and a covariance matrix Σ_r of the color components.

2.3 Training the Gaussian Parameters

In order to obtain the Gaussian parameters, an EM algorithm is used with the first η frames of the sequence as the training data ($\eta = 5$). Initialization (E-step) of the Gaussian parameters is done as follows: all prior probabilities are assumed as equal, i.e., the mean is randomly chosen from a uniform distribution within a possible pixel value range, and the covariance matrix is assumed to be an identity matrix. Training (M-step) is performed by iteratively updating the above-mentioned parameters. Refer [4] for details. The iteration stops if either the change in the value of the means is less than 1 percent with respect to the previous iteration or when a user-specified maximum iteration number is exceeded. We start with 10 Gaussian components ($C_0 = 10$) and merge similar Gaussians after the training by the method in [10], resulting in C Gaussians. The parameters of the established C Gaussians are then used to classify pixels into one of the C classes in subsequent frames.

2.4 Classification of Individual Pixels

The color classification of the individual pixels is achieved by a maximum *a posteriori* (MAP) classifier. Once the Gaussian mixture model G for pixel color is obtained, we compute the MAP probability that each pixel in the subsequent frames belongs to each Gaussian component. The class that produces the largest probability value for a pixel v is chosen as the pixel-color class label ω_L for that pixel.

$$\omega_L = argmax_r(log(P(\omega_r|v))), 1 \leq r \leq C \tag{5}$$

2.5 Relaxation Labeling

The pixel color classification results in free-form blobs of different color labels adjacent to one another. Connected component analysis is used to register adjacent blobs. Relaxation labeling [16,13] is performed to remove small blobs and to achieve coherent large blobs according to the color similarity of the adjacent blobs, as follows: two adjacent blobs A_i and A_j are merged together if they are similar in color, where the similarity is defined by the Mahalanobis distance δ_Φ of color feature Φ between A_i and A_j, as follows:

$$\delta_\Phi = (\Phi_i - \Phi_j)^T {\Sigma_\Phi}^{-1} (\Phi_i - \Phi_j) \tag{6}$$

$$\Phi = [\mu_H, \mu_S, \mu_V]^T \tag{7}$$

where Σ_Φ is the covariance matrix of color channels for all blobs in the image. Blobs A_i and A_j are merged if δ_Φ is less than the threshold T_Φ, which was obtained from training data. (Refer [13] for details.) The result of the relaxation labeling is a set of blobs that segment the foreground image into coherent regions based on color and position, which provides view-based initial segmentation and tracking of the foreground human figures. More detailed segmentation and tracking is achieved at the object level that incorporate a 3D human body model projected onto the 2D image plane.

3 Human Body Modeling and Cost Function

3.1 Human Body Modeling

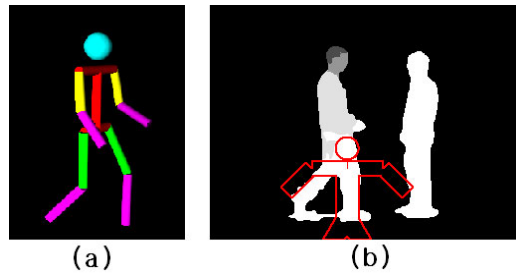


Fig. 1. 3D Model body(a), and overlapping between image and projected model body(b).

As shown in figure 1(a), the body is modeled as a configuration of nine cylinders and one sphere. (Two extra horizontal cylinders are drawn for better illustration, but do not exist in our modeling.) These are projected onto a 2D real projection plane. A sphere represents the head, while the rest of model body is modeled using cylinders of various radii and lengths. Our model body is very similar to that used in [17]. We may use more sophisticated tapered/oval-circular cylinders, but our method is robust enough that a rough body modeling is enough to track foreground images. Currently, we use only nine 1-DOF (1 degree-of-freedom) joints plus body displacement DOF, and a vertical rotation to compensate for a camera view. These are our control variables for the cost function. Body parts are linked together by kinematic constraints in a hierarchical manner. This can be considered to be a tree structure with the base at the pelvis (the bottom of a cylinder represents the torso) of the model body. The methods presented in [8] cannot handle body occlusion, and ignore half of the body. We have overcome the body occlusion problem using computational geometry, by computing the union of the projected model body parts and then computing the intersection with overlapping input image silhouettes.

Due to possible variations in the size of human subjects, our method must be able to handle various magnitudes of height and thickness of the human in input images. We can handle this problem by initially counting the number of pixels from a pixel density map of a preprocessed image to guess the height of a body. Then, we can run an initial parameter optimization, solving by making the thickness of a body an additional variable while using a pre-guessed human height. We assume that humans are standard in their body segment length ratio. After solving the best thickness for a model body, the thickness variable becomes a constant and remains set as a constant for the remaining body tracking over various input images for the same human model. Figure 1(b) shows the initial state of searching for the best overlapping configuration.

3.2 Forward Kinematics-Based Cost Function

The kinematics problem is well explained in Craig's book [3]. A projected model body is assumed to have an affine transformation [8]. This is mathematically acceptable if an orthographic camera model is used and the model body moves parallel to the projection plane. Given a set of joint angles and body displacement values, the forward kinematics function, $h(\cdot)$, where $\cdot$ is a generic variable(s), computes points of body segment boundaries. The P matrix then projects the computed boundary points on a 2D projection plane. The projected boundary points are compared to the foreground image silhouette without using color information from the foreground image to match the image silhouette. $g(\cdot)$ converts projected points to a polygon(s). The input image is preprocessed using $f(\cdot)$. The projection plane is represented in the real numbers. $r(\cdot)$ converts the input image silhouette to a polygon(s) in real number domain. The representation in the real number domain makes the derivative-based parameter optimization possible. For easier tracking of the body, we use head part information from image preprocessing results [13]. The distance between the silhouette head center and the model head center is minimized. However, we can track body motion without head information using the following equation:

$$\begin{aligned} c(I,\theta) = & -w_1 \times [a(r(f(I)) \cap (\cup_l g(P \cdot h_l(\theta,t))))] \\ & + w_2 \times \sum_{xy} (w_d(x,y) \times a(d(x,y) \cap (\cup_l g(P \cdot h_l(\theta,t))))) \\ & + w_3 \times (h_{hc}(\theta,t) - q(f(I)))^2 \end{aligned} \tag{8}$$

Let us explain the notation used in equation 8 in more detail. P is an orthographic camera projection matrix, projecting a 3D model body to the 2D plane. This is a constant matrix, assuming that camera parameters are fixed. $h_l(\cdot)$ is a nonlinear forward kinematics function of an *l-th* body part in terms of joint DOF. $\bar{\theta}$ is a joint DOF vector, represented in a column matrix. (We denote $\bar{x}$ as vector x.) $\bar{\theta}$ is a function of time when a complete sequence is considered. $g(\cdot)$ is a function that takes 2D input points and converts them to a polygon. $r(\cdot)$ is a real function that takes an image as an input and converts its foreground (non-zero value) part to a set of polygons, possibly with holes. Given a raw image, $f(\cdot)$ returns a preprocessed image. Given a preprocessed human image, $q(\cdot)$ computes a head center. I is a raw input image. $I(x,y)$ denotes a grey level pixel value at image location (x,y). $d(x,y)$ is a square polygon of area size 1, representing a pixel located at the (x,y)-*th* position in the distance map [6]. t means time related with frames. $w_d(x,y)$ is a distance map value of $(b(I))(x,y)$, and has a scalar value, where $b(I)$ is background part of input image. $\cap$ is an operation that takes two polygons and returns an intersected polygon(s). $\cup$ is an operation that takes two polygons and returns a unioned polygon(s). $a(\cdot)$ is a function that takes a polygon and returns the size of its area. $c(\cdot)$ is a cost function for parameter optimization that depends on a raw input image I and model DOF variables θ. $h_{hc}(\cdot)$ is the model head center function using forward kinematics, and w_i, $i = 1, 2, 3$ are weighting factors.

Because the input is $\bar{\theta}$, a vector of joint DOF values, this computation is purely forward kinematics-based and thus presents no singularity problem. In deriving forward kinematics equations, we set local coordinates for various parts of the body to make a hierarchical body. We can assign joint limits for model body parts. Only a model-based approach allows easy geometric constraint computation, which is implemented as possible ranges for each variable. Head part information is provided from image preprocessing. Our method is robust enough, regardless of head tracking, that we can find a best match from image silhouette matching. However, the weakness of our approach is in estimating how many humans need to be tracked.

At the start, the model body has initially guessed arbitrary input values. Even though there is no overlapping between the projected model body and the image silhouette, the cost function knows which direction is the best move to minimize the cost function, due to the distance map. The value of our cost function decreases when by moving the projected model body is moved to the foreground image, because overlapping of the foreground image increases while overlapping with background decreases. The parameter optimizer will find the best fitting configuration that produces the optimum value of the cost function. If there is an occlusion between two persons, our method must rely on a distance map [6] for each person to track correctly.

3.3 Computational Geometry for the Cost Function

An image silhouette is one that is converted from the 2D integer pixel domain to a real domain such that the resulting image silhouette becomes a jagged-edge polygon with only horizontal and vertical edges. Each body part is projected on the 2D projection plane. In this process, occlusion occurs among body parts. Basically, projected objects are either polygon shaped or circlular. Because we are concerned with how well the model body overlaps with the foreground image silhouette, it is quite natural that the resulting body parts occlude each other. The projected occluded body parts are unioned using a modified version of Weiler-Atherton's polygon union algorithm [7]. The resulting polygon-like object(s) may even have holes in it, showing our computational geometry works fine. We compute the polygon intersection between the image silhouette and the model silhouette. We use pixel-based computational geometry because it will allow us to compute a cost function based on a distance map, which has different values for each pixel. We found the best fitting configuration using the GRG2 [11] optimization package with the cost function discussed so far.

4 Kinematics and Singularity

We avoid using inverse kinematics due to its singularity problem. We may also work in differential kinematics. In terms of differential kinematics, we usually work either on the velocity level or on the acceleration level. Working on highly differentiated equations does not always have merits. Computed velocities may

have errors in image overlapping computation, which may make our differential kinematics computation unreliable. Many papers presented up to now are based on inverse kinematics and usually work on the velocity level. This type of approach requires computation of joint angular velocities given the end tip velocities. This process involves a matrix, called a Jacobian inversion. If the Jacobian matrix is not square, one must compute a pseudo-inverse Jacobian matrix [2]. Even if the matrix is square, it can be singular. Because they take error-prone difference values computed between two images, the resulting computation is not truly reliable [12]. Our approach, on the other hand, computes in a pure forward kinematics-based way. This is the major difference from others' work. Let $\bar{f}(\bar{\theta})$ be our forward kinematics equation where $\bar{\theta}$ is a body DOF-related vector consisting of joint angles and body displacements. In our method, we control $\bar{\theta}$. Given $\bar{\theta}$, we get the resulting $\bar{f}$ values. If the $\bar{f}$ values are not the values we need, we modify the input $\bar{\theta}$. This process is done by parameter optimization. After this computation, we interpolate $\bar{\theta}$ to get approximate $\dot{\bar{\theta}}$ and $\ddot{\bar{\theta}}$. Differential kinematics-based methods, on the other hand, use differential kinematics equations. By differentiating $\bar{f}(\bar{\theta})$, they can get $\dot{\bar{f}} = \frac{\partial \bar{f}}{\partial \bar{\theta}} \dot{\bar{\theta}}$. $\frac{\partial \bar{f}}{\partial \bar{\theta}}$ is called a Jacobian matrix. They get $\dot{\bar{f}}$ values from input images. In order to compute $\dot{\bar{\theta}}$, they need to compute $\frac{\partial \bar{f}}{\partial \bar{\theta}}^{-1}$. But if this matrix is not square, they must compute a pseudo-inverse matrix [2]. However, even if the matrix is square, there is some chance that it is singular. For singular matrix inversion, they usually use singular value decomposition [14].

5 Experimental Results and Conclusion

We have worked on several different 2D-based human motions. The example shown in figure 2 has a static background with one person (the left) pushing another (right) person away. The white line shows the result of union of every model body part. There are even holes in the resulting body, polygon-like object. We can track every person appearing in the scene, and as long as there is no heavy occlusion between the subjects, motion tracking is satisfactory. Using body part information, our method can handle even heavy occlusion to a limited degree, although the tracking quality is degraded. The graph of figure 2 shows the corresponding pair of joint angles of torso, shoulder, and elbow of the pushing motion of the model body as well as the real human in the scene. All joint angle data on the graph is for the left arm. We measured human joint angles from images and compared it with model tracking data. In the graph labels, **M** means model body data while **R** means real body data. **U** means upper while **L** means lower. Excellent matching was achieved for the Model left upper arm (**M L U ARM**) data to the real left upper arm (**R L U ARM**) data as well as for model torso data (**M TORSO**) to real torso data (**R TORSO**). In order to compute model data, we did individual motion configuration fitting for every frame. Only the left arm of the model body is tracked for the pushing motion. This is quite natural and is the current limitation of using monocular video input, which provides very limited input information. Our method finds

one locally optimum solution from the search space, as can be seen in the graphs, and the tracking is good. In the middle of the pushing motion, there is a body occlusion between the two persons. Because our cost function is pixel-area based, in the third image of figure 2, it appears that the left person's hand is on the right person's face. This is due to not using body part information for tracking motion, and can be fixed if body part information is used. Figure 3 shows the walking (approaching) and shaking hands motions.

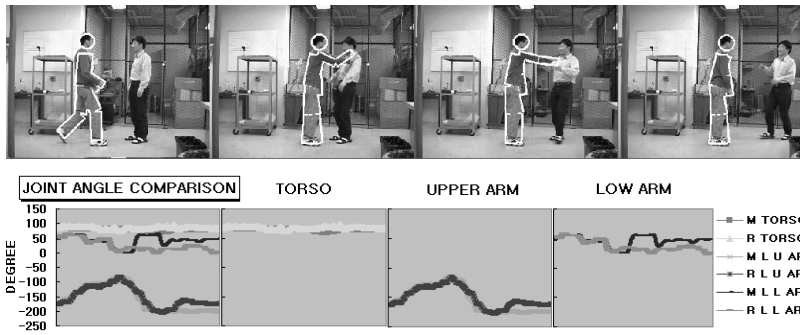


Fig. 2. The subject with the model figure superimposed, shown over a pushing motion. Corresponding joint angle graph of torso, shoulder, and elbow.

In this paper, we presented a new approach to human motion tracking that combines view-based and model-based methods at the pixel level and the object level, respectively. The view based-methods at the pixel level use a Gaussian mixture model and a relaxation labeling technique to achieve initial segmentation of the foreground human figures. The initial segmentation is data-driven, and it significantly reduces the overhead in model initialization and fitting. The model based-method at the object level uses a 3D human body model and parameter optimization techniques to achieve refined segmentation and tracking of the moving humans. Using the human body model achieves robustness of our system. Our forward kinematics-based system overcomes the problem of singularity in inverse kinematics-based systems, and our nonlinear optimization-based fitting does not depend on the number of particles as is the case in stochastic sampling-based approaches. We have presented a solution to the model body part occlusion problem [8] using computational geometry. As demonstrated in figure 3, the motion tracking results from video sequences are very encouraging.

References

1. J.K. Aggarwal and Q. Cai. Human motion analysis: a review. *Computer Vision and Image Understanding*, 73(3):295–304, 1999.
2. H. Asada and J. Slotin. *Robot Analysis and Control*. John Wiley and Sons, New York, NY, 1985.

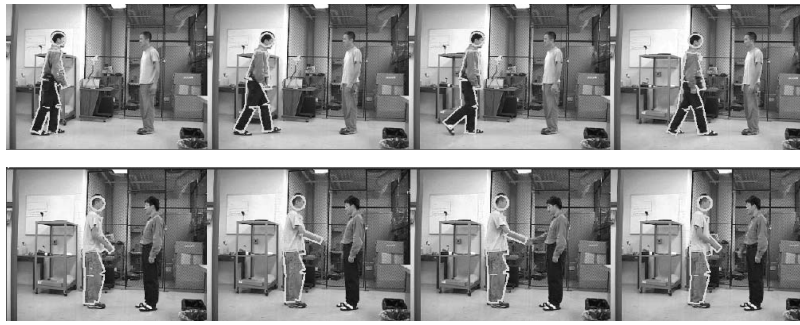


Fig. 3. The subject, with the model figure superimposed, shown over a walking (approaching) motion, and a shaking hands motion.

3. J. Craig. *Introduction to Robotics Mechanics and Control*. Addison-Wesley, Reading, MA, 1986.
4. R.O. Duda, P. Hart, and E. Stork. *Pattern Classisifcation*, chapter Unsupervised Learning and Clustering, pages 517–583. Wiley, New York, 2 edition, 2001.
5. R. Freeman and D. Tesar. Dynamic Modeling of Serial and Parallel Mechanisms / Robotic Systems : Part I - Methodology. In *Trends and Developments in Mechanisms, Machines and Robotics, 20th Biennial Mechanisms Conference*, 1988.
6. D. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.
7. F. Hill. *Computer Graphics*. Macmillan, 1990.
8. Y. Huang and T. S. Huang. Model-based human body tracking. In *International Conference on Pattern Recognition*, 2002.
9. S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated motion. In *International Conference on Automatic Face and Gesture Recognition*, pages 38–44, Killington, Vermont, 1996.
10. S. Khan and M. Shah. Tracking people in presence of occlusion. In *Asian Conference on Computer Vision*, Taipei, Taiwan, 2000.
11. L. Lasdon and A. Waren. *GRG2 User's Guide*, 1989.
12. D. Morris and J. Rehg. Singularity analysis for articulated object tracking. In *Computer Vision and Pattern Recognition*, 1998.
13. S. Park and J.K. Aggarwal. Segmentation and tracking of interacting human body parts under occlusion and shadowing. In *IEEE Workshop on Motion and Video Computing*, pages 105–111, Orlando, FL, 2002.
14. W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes*. Cambridge University Press, Cambridge, England, 1986.
15. R. Rosales and S. Sclaroff. Inferring body pose without tracking body parts. In *Computer Vision and Pattern Recognition*, pages 721–727, Hilton Head Island, South Carolina, 2000.
16. L. Salgado, N. Garcia, J. Menedez, and E. Rendon. Efficient image segmentation for region-based motion estimation and compensation. *IEEE Trans. Circuits and Systems for Video Technology*, 10(7):1029–1039, 2000.
17. H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *ECCV (2)*, pages 702–718, 2000.

Animating Behavior of Virtual Agents: The Virtual Park

F. Luengo[1,2] and A. Iglesias[2⋆]

[1] Department of Computer Science, University of Zulia, Post Office Box #527, Maracaibo, Venezuela
fluengo@cantv.net
[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, s/n, E-39005, Santander, Spain
iglesias@unican.es
http://personales.unican.es/iglesias

(Not blood for oil. Not in our name)

Abstract. This paper proposes a framework for animating the behavior of virtual agents that can reproduce several human behavior features with a minimal input from the animator. While this work is not actually concerned about obtaining a high-quality animation in terms of motion, illumination, textures or any other rendering features, we focus on modeling and simulating the human behavior so that the virtual agents behave in a realistic way from the point of view of a human observer. The framework is illustrated by a simple but still interesting example: a virtual park where the virtual agents can walk, chat, play, rest, do physical exercices, interact with others, etc. The system is flexible enough so that any new goal, feeling, sense, internal state, knowledge or social habit can easily be added without modifying the underlying structure. Some examples of human behavior features such as the competition, anxiety, etc. are also described.

1 Introduction

Imagine a virtual park in which several autonomous agents can behave in a human-like way. For example, children can play whereas an adult is just resting for a while trying to recover himself from the exhausting and stressing task of taking care of them (see Figure 1). At the same time, in other different area of the park a couple of adults are talking to each other while other virtual agents are just taking a walk, sitting down or playing (see Figure 2).

These two simple static pictures are useful to illustrate the huge amount of different and complex tasks the virtual agents must perform as well as the extremely large number of subtle details the animator must consider. Thus, for the virtual agents to finally achieve their goals, they should firstly (1) explore the sorrounding environment and identify the different objects within as well as

⋆ Corresponding author

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 660–669, 2003.
© Springer-Verlag Berlin Heidelberg 2003

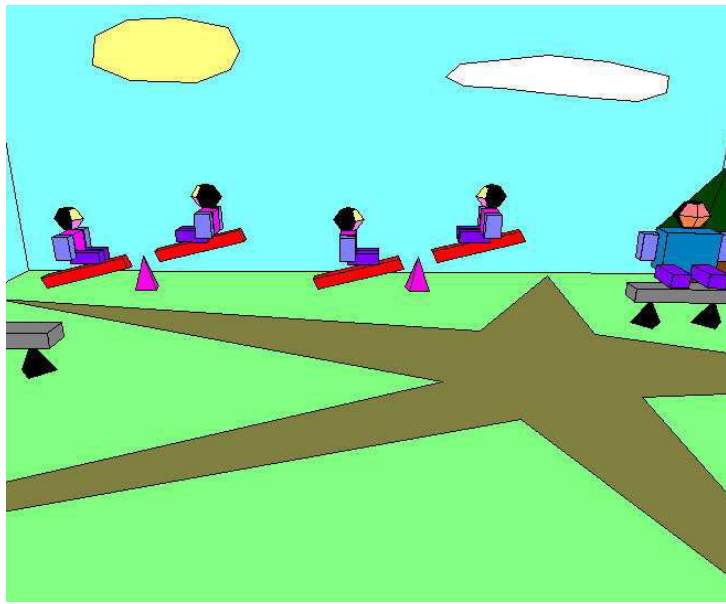


Fig. 1. Four children are playing with the seesaw while the adult on the right is just resting for a while in order to recover from the exhausting and stressing task of taking care of them

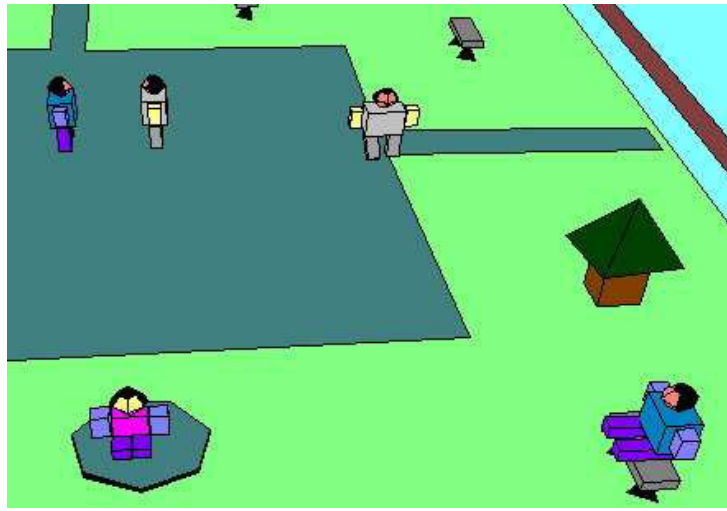


Fig. 2. In this picture, virtual agents carry out different actions: a couple of adults are talking to each other while other virtual agents are taking a walk, sitting down or playing

General Tasks	Specific sub-tasks
[T_1] Explore the environment and identify the objects and other agents	Manage vision sensors Set vision range Classify visible/hidden objects Determine geometry and positions Update the knowledge base Update (short,mid,long)-term memory stack
[T_2] Move towards the objects and/or the agents in order to interact with them	Avoid static obstacles Avoid dynamic obstacles Calculate distances Update the knowledge base Update the internal states
[T_3] Perform the agents' goals	Play alone Play with others Take a walk Sit down and rest Talk to others Do physical exercices Update the knowledge base Update the internal states
[T_4] Take a decision about the next goal to be done	Update the knowledge base Update the internal states Update the goals list Update the goal priority list Talk to others Do physical exercices Update the knowledge base

Table 1. List of general tasks and associated sub-tasks

look for other possible agents, (2) move towards the objects and/or agents in order to interact with them, (3) satisfy their goals and finally (4) take a decision about the next goals to be done once they have already finished their current ones. At its turn, each task may require several associated sub-tasks, as shown in Table 1.

Of course, all these sub-tasks can be input by the animator but simply scripting them into the system. However, this solution does not yield a high degree of autonomy, as the virtual agents are always limited to follow programmer's instructions in a sequential way determined by a list of goals and priorities that can be established *a priori*. In order to provide the virtual agents with a certain degree of autonomy we should allow the agents to behave in a less deterministic way. In other words, we should allow the agents to have different outputs for identical inputs, depending on a number of different internal (physical, mental, psychological, internal states, knowledge, etc.) and external factors (either from the environment or from other virtual agents). These factors will generally decide

the different actions the virtual agents will take with a minimal user intervention. Thus, the role of the animator is just to input some initial values before launching the animation. During the running process, the agents will behave in an autonomous way. Even, virtual agents behavior may depend on a number of stochastic variables so that it becomes totally unpredictable. This is the aim of the present work.

The structure of this paper is the following: in Section 2 we present some previous work in this field. Section 3 describes briefly the proposed framework for the behavioral engine. An illustrative example to show the performance of the system is described in Section 4. Finally, Section 5 closes the paper with the main conclusions and some implementation details.

2 Previous Work

Although several researchers have worked in behavioral animation of virtual agents during the last few years (see, for instance, [1, 2, 3, 4, 5, 7, 8, 10, 13, 14, 15, 17, 18, 19, 20, 25, 26] and references therein) perhaps the works with stronger relationship with the current one are those from Dimitri Terzopoulos [9, 23] and Daniel Thalmann [6, 16, 22]. In particular, Terzopoulos proposed a framework for animating natural ecosystems with minimal input from the animator. In their seminal paper "Artificial fishes" [23] presented at Siggraph'94, Tu & Terzopoulos described an approach to emulate the natural complexity of a virtual marine world inhabited by artificial fishes that can swin hydroynamically in simulated water through a motor control of internal muscles that motivate fins. Their outstanding motion system was combined with a powerful perception and behavior motor incorporating most of the typical behavioral routines exhibited by the real fishes. Other additional remarkable features such as beautiful textures, a physics-based design of the fishes and advanced rendering features allowed the authors to obtain a very realistic simulation of a marine world. However, the number of different routins that fishes can exhibit is not certainly quite large and the system seems to replicate the behavior phenomena over the time.

Undoubtely, human behavior is much more complex than that of fishes (we hope). Hence, the simulation of human behavior is, by large, a much more varied and difficult task. It has been accomplished by, among others, Prof. Thalmann and collaborators at the Swiss Institute of Technology [6, 16, 22]. In this case, emphasis was put on giving the virtual actors a higher degree of autonomy without losing control. Although the results show a realistic simulation of many human behaviors, their virtual actors are still far from being completely autonomous. What we would like to get is virtual agents with the highest degree of autonomy so that the role of the animator is reduced to a simple spectator once the animation of the virtual world is launched.

The approach presented in this work take advantage of the strenghts of the behavioral motor of Terzopoulos and Thalmann's approaches. For example, higher complexity of human behavior routines implies that our behavioral engine must be more sophisticated than Terzopoulos' behavior motor. In addi-

tion, Thalmann's scheme must be modified according to the new requirements for a higher autonomy for the virtual agents. Indeed, the programmer should be involved in the initialization step only, by selecting the environment and setting the initial values of agents parameters such as vision, memory, tiredness, happiness, anxiety, physical resistance, recovering capacity, sociability, etc. Further, the system must be flexible enough to allow the animator to incorporate new features without modifying the underlying structure. Of course, our system has also many weaknesses with respect to Terzopoulos & Thalmann's systems: the most remarkable (but not the only one) is that, since our primary goal is the accurate simulation of human behavior, little effort was devoted to the graphical output and consequently, our rendering and motion subsystems are poorer than those in Thalmann's approach (and of course, much poorer than impressive Terzopoulos' graphical outputs).

3 The Behavioral Framework

A scheme of our behavioral engine framework is shown in Figure 3. Their corresponding subsystems are briefly described in the next paragraphs.

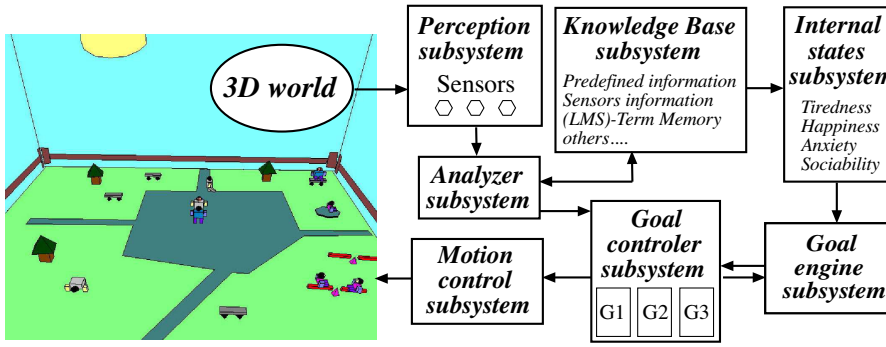


Fig. 3. Behavioral engine framework scheme

First of all, we must consider a *perception subsystem*, whose general task is to capture information from the sorrounding environment, i.e., task T_1. To this aim, the subsystem includes a set of individual sensors so that each virtual agent is able to analyze the environment, capture the most relevant information and send it to the analyzer. At its turn, the *analyzer subsystem* proccesses the world information acquired by the perception subsystem and then updates the *knowledge base* accordingly. Conversely, a new entry in the knowledge base might modify the previous analysis and, consequently, the agent behavior as well. On the other hand, the analyzer can also modify the *goal controler subsystem* when, for example, new information acquired by the sensors or coming from the knowledge base

Subsystem	Specific sub-tasks
Perception	Manage vision sensors Set vision range Manage speech sensors Capture geometry and positions data
Analyzer	Classify visible/hidden objects Calculate geometry and positions Update the knowledge base Update (short,mid,long)-term memory stack Calculate static obstacles Calculate dynamic obstacles Calculate distances Update the goal controler subsystem
Knowledge base	Store the agents' information Store (short,mid,long)-term memory data Update the analyzer subsystem Update the internal states subsystem
Internal states	Update the internal state functions Update the goal engine subsystem
Goal engine	Update the goals list Update the goal priority list Update the goal controler system
Goal controler	Check the goals for feasibility Determine the goal to be done Update the motion subsystem
Motion control	Manage the agents' motion Avoid static obstacles Avoid dynamic obstacles Manage speech features

Table 2. Different subsystems (left column) of our framework and their associated sub-tasks (right column)

is attained. The goal controler is also affected by the *internal states subsystem* via the *goal engine subsystem*. The first one manages some internal states of the virtual agent, such as tiredness T, happiness H, anxiety A and sociability S, which are functions of different variables (see [12] for more details). The second subsystem is the component that updates, sorts and finally stores agent's goals into a priority list. Of course, these lists cannot be empty and therefore, some initial input from the user is required to launch the animation. During the running process, they are updated to adapt to the changing environment and circumstances. Once the lists of goals and priorities are defined, they are sent to the goal controler subsystem. This component will determine which goals the agent is actually going to get (in other words, tasks T_3 and T_4). Firstly, this subsystem check for those goals that cannot be carried out and consequently must be rejected. For example, if the agent is tired, he/she will look for a bank

to sit down. If there is no seats available or they are not free at this time, the agent must modify his/her goals. This leads to the concept of feasibility, that is used to check which goals are actually feasible at each span. From this point of view, the goal controler subsystem acts like a filter modifying the goals and priorities received from the goal engine subsystem.

Once the goals and priorities are defined by the goal controler subsystem, this information is sent to the *motion subsystem*. This component is responsible for task T_2, that is, for all the motion routines, including avoiding static and dynamic obstacles, walking, sitting-down, playing with the seesaw, playing with the wheel and doing physical exercices. Some much subtler motion routines, such as competing routines and others, are also incorporated in our system. However, agents motion is still very simple and further work is currently being developed to create more and better motion routines.

Table 2 shows the different subsystems of our framework (first column) and the specific sub-tasks they perform (second column).

4 An Illustrative Example

In this section the performance of the proposed behavioral engine framework is analyzed through an illustrative example. We should remark, however, the human behavior is too much complex to be illustrated here and hence we restrict ourselves to show only a few behavioral features. In particular, we are interested to discuss the competing routine, which is illustrated in Figure 4. This figure shows sixframes of an animation generated in our framework. A longer animation can be found at:

`http://personales.unican.es/iglesias/animation.avi`.

For this example, we asume that reader is familiar with the playground environment described earlier. Then, we consider four children and an adult who looks after them while doing some physical exercices (frame 1). Initially the children would like to play with the seesaw so they move towards it while the adult is doing physical exercices. The seesaw require two people to play with on the "first come, first served" basis. This implies that the three children closer to the seesaw must compete among them (frames 2 to 4) to get the two seats on the seesaw, while the third child should wait for another one to play with (frame 5). Simultaneously, the adult is getting tired through the tiredness function described in [12]. Finally, the fourth child reaches the available seat at the second seesaw and consequently the two couples of children are playing together. At the same time, the adult is so tired that he decided to sit down for a while in order to rest (frame 6). The final stage is that shown in Figure 1.

5 Conclusions and Further Remarks

This paper introduces a new framework for the simulation of a number of the most typical human behavior features. This new framework allows the virtual

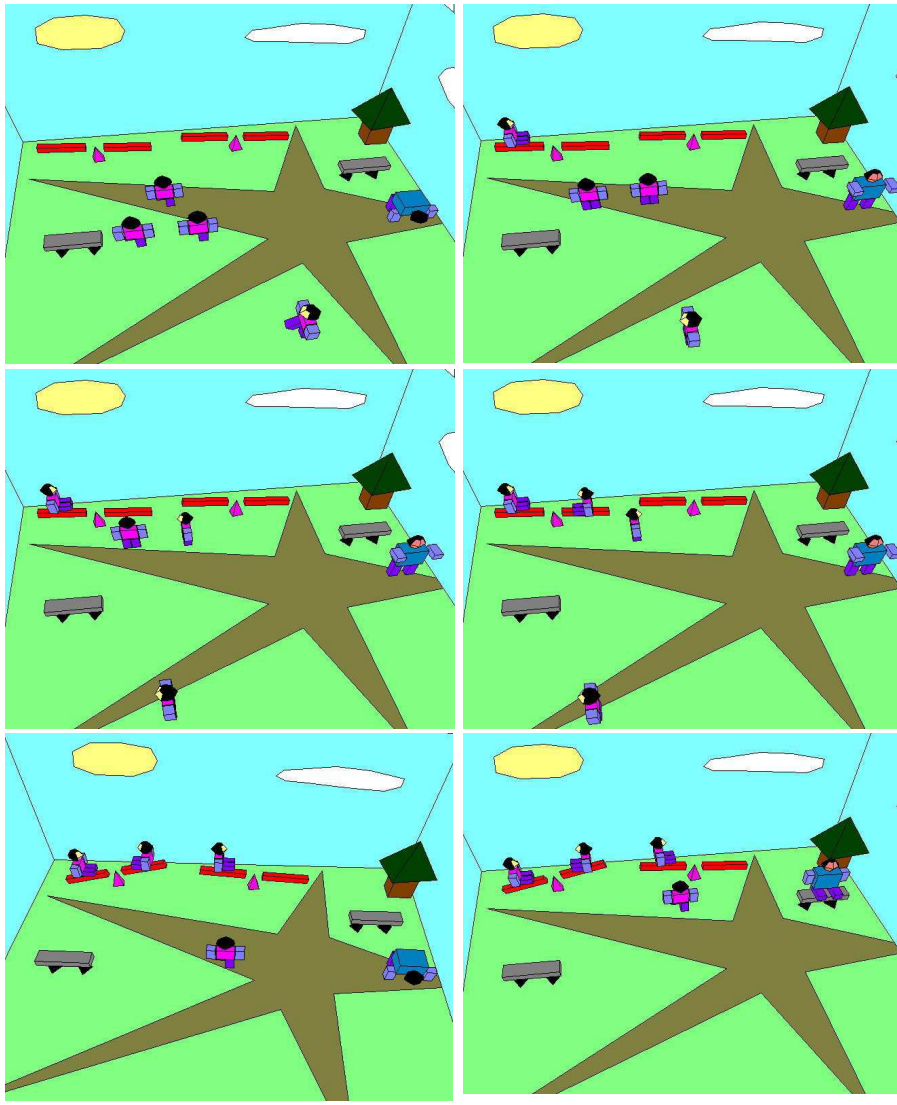


Fig. 4. (left-right, top-bottom) Six different frames of the animation described in Section 4

agents to interact among them and with the environment in a quite realistic way. Some features of this new scheme as well as an illustrative example have been briefly discussed throughout the paper. All the framework (including the graphical module) has been completely implemented in C++ on a Pentium III. No special requirements are needed to run the program.

Finally, we would like to remark that the framework presented here is just a first step of a long way to walk. On one hand, we expect that future versions will include a better graphical output, including textures, shadows, reflections and illumination models. At this time, we are using OpenGL to incorporate all these graphical improvements. On the other hand, the accurate simulation of human behavior requires to modify the system substantially. The list of new functions and parameters to be defined is virtually infinite, and it is still unclear nowadays which are the functions modeling most of the human actions and decisions. These and other tasks, such as the addition of new sensations, feelings, beliefs and capabilities to the virtual agents, the improvement of agents' motion, knowledge base and deduction engine and a more accurate modeling of many human behavior functions are the following steps to be done in this work. In addition, as shown in [11, 21, 24] the use of Artificial Intelligence tools, mainly neural networks and expert systems, can improve the current framework dramatically and it is currently under analysis. The obtained results will be reported elsewhere.

The authors would like to thank to the referees for their suggestions to include a brief discussion on the relation between this work and other related approaches. Finally, we would like to dedicate this paper to all people around the world looking for peace in spite of the unacceptable pressure, coercion and manipulation from some irresponsible and criminal governments claiming for an unjustified and illegal war.

References

[1] Badler, N.I., Barsky, B., Zeltzer, D. (eds.): Making Them Move. Morgan Kaufmann, San Mateo, CA (1991)

[2] Badler, N.I., Phillips, C.B., Webber, B.L.: Simulating Humans: Computer Graphics Animation and Control. Oxford University Press, Oxford (1993)

[3] Blumberg, B.M., Galyean, T.A.: Multi-level direction of autonomous creatures for real-time virtual environments. Proc. of SIGGRAPH'95, ACM, New York (1995) 47-54

[4] Boulic, R., Becheiraz, P., Emering, L., Thalmann, D.: Integration of motion control techniques for virtual human and avatar real-time animation. Proc. of ACM Symposium on Virtual Reality Software and Technology, ACM, New York (1997) 111-118

[5] Brogan, D.C., Metoyer, R.A., Hodgins, J.K.: Dynamically simulated characters in virtual environments. IEEE Computer Graphics and Applications (1998) 58-69

[6] Caicedo, A., Thalmann, D.: Virtual humanoids: let them to be autonomous without losing control, Proceedings of the Fourth International Conference on Computer Graphics and Artificial Intelligence, D. Plemenos (ed.) University of Limoges, Limoges (2000) 59-70

[7] Cerezo, E., Pina, A., Seron, F.J.: Motion and behavioral modeling: state of art and new trends. The Visual Computer, **15** (1999) 124-146

[8] Farenc, N., Boulic, R., Thalmann, D.: An informed environment dedicated to the simulation of virtual humans in urban context, Proceedings of EUROGRAPHICS'99 (1999) 309-318

[9] Funge, J., Tu, X. Terzopoulos, D.: Cognitive modeling: knowledge, reasoning and planning for intelligent characters, Proceedings of SIGGRAPH'99, ACM, New York (1999) 29-38
[10] Granieri, J.P., Becket, W., Reich, B.D., Crabtree, J., Badler, N.I.: Behavioral control for real-time simulated human agents, Symposium on Interactive 3D Graphics, ACM, New York (1995) 173-180
[11] Grzeszczuk, R., Terzopoulos, D., Hinton, G.: NeuroAnimator: fast neural network emulation and control of physics-based models. Proceedings of SIGGRAPH'98, ACM, New York (1998) 9-20
[12] Luengo F., Iglesias A.: A new architecture for simulating the behavior of virtual agents. Proceedings of the International Conference on Computational Science-ICCS 2003. Springer-Verlag, Lecture Notes in Computer Science (2003) (in press)
[13] Kallmann, M.E., Thalmann, D.: A behavioral interface to simulate agent-object interactions in real-time, Proceedings of Computer Animation'99, IEEE Computer Society Press, Menlo Park (1999) 138-146
[14] Maes, P., Darrell, T., Blumberg, B. Pentland, A.: The alive system: full-body interaction with autonomous agents, Proceedings of Computer Animation'95, IEEE Computer Society Press, Menlo Park (1995) 11-18
[15] McKenna, M., Pieper, S., Zeltzer, D.: Control of a virtual actor: the roach, Proceedings of SIGGRAPH'90, Computer Graphics, **24**(2) (1990) 165-174
[16] Monzani, J.S., Caicedo, A., Thalmann, D.: Integrating behavioral animation techniques, Proceedings of EUROGRAPHICS'2001, Computer Graphics Forum, **20**(3) (2001) 309-318
[17] Multon, F., France, L., Cani-Gascuel, M.P., Debunne, G.: Computer animation of human walking: a survey, Journal of Visualization and Computer Animation, **10** (1999) 39-54
[18] Perlin, K., Goldberg, A.: Improv: a system for scripting interactive actors in virtual worlds, Proceedings of SIGGRAPH'96, ACM, New York (1996) 205-216
[19] Renault, O., Magnenat-Thalmann, N., Thalmann, D.: A vision-based approach to behavioral animation, Journal of Visualization and Computer Animation, **1** (1990) 73-80
[20] Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model, Computer Graphics, **21**(4) (1987) 25-34
[21] Sims, K.: Evolving virtual creatures, Proceedings of SIGGRAPH'94, ACM, New York (1994) 15-22
[22] Thalmann, D., Noser, H.: Towards autonomous, perceptive and intelligent virtual actors, Lecture Notes in Artificial Intelligence, **1600** (1999) 457-472
[23] Tu, X. Terzopoulos, D.: Artificial fishes: physics, locomotion, perception, behavior, Proceedings of SIGGRAPH'94, ACM, New York (1994) 309-318
[24] Van de Panne, M., Fiume, E.: Sensor-actuator networks, Proceedings of SIGGRAPH'93, Computer Graphics **27** (1993) 335-342
[25] Wilhelms, J., Skinner, R.: A "notion" for interactive behavioral animation control, IEEE Computer Graphics and Applications, **10**(3) (1990) 14-22
[26] Zeltzer, D.: Motor control techniques for figure animation, IEEE Computer Graphics and Applications, **2**(9) (1982) 53-59

A Curve Design Method with Shape Control

M. Sarfraz and M. Balah

Information and Computer Science Department
King Fahd University of Petrolium and Minerals
Dhahran, Saudi Arabia.
{sarfraz, mbalah}@kfupm.edu.sa

Abstract. A curve design method has been proposed which, in addition to enjoying the good features of cubic splines, possesses interested shape design features too. Two families of shape parameters have been introduced in such a way that one family of parameters is associated with intervals and the other with points. These parameters provide a variety of shape controls like *point* and *interval* tension. This is an interpolatory curve scheme, which utilizes a piecewise rational cubic function in its description. The proposed method enjoy ideal geometric properties and geometric continuity of order two is also achieved.

Keywords. Spline, Bernstein-Bézier, interpolation, tension, shape control.

1. Introduction

Designing of curves, especially those curves which are robust and easy to control and compute, has been one of the significant problems of Computer Graphics and Geometric Modeling. Specific applications including Font Designing, Capturing Hand-Drawn Images on computer screens, Data Visualization, and Computer-Supported Cartooning are main motivations towards curve designing. In addition, various other applications in CAD/CAM/CAGD are also a good reason to study this topic. Many authors have worked in this direction. For brevity, the reader is referred to literature [1-17] in the reference section.

In curve designing, the class of rational cubic functions [3, 5, 6, 10, 11, 17, 16, 15], is one of the most powerful tools. They can define space curves and curves with inflections. This paper presents a simple and effective method for interactive and smooth curve design using a class of rational cubic piecewise functions in [10]. The spline method constructed and proposed is the marriage of the rational spline method [10] and the Nu-spline [13] method. Two families of shape parameters have been introduced in the construction of the proposed spline. These parameters are such that an independent parameter has been associated with each individual piece of spline curve and each joint position. They provide stronger and variety of shape features like interval and point tension control.

The number of shape parameters, occurring in the proposed scheme, are similar to those in weighted *v*-spline [8] except with the difference that interval weights are

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 670-679, 2003.
© Springer-Verlag Berlin Heidelberg 2003

much stronger than those in weighted *v*-spline. In addition, the proposed scheme involves a rational piecewise function whereas in weighted *v*-spline piecewise cubics were used. Furthermore a GC^2 continuity has also been achieved as compared to only C^1 continuity in the weighted *v*-spline.

Section 2 has been reported on a brief review of weighted *v*-splines. The proposed rational spline is based on a rational cubic Hermite interpolant, which is introduced in Section 3 together with some preliminary analysis. Section 4 gives the construction of the geometric rational spline. The analysis of its behaviour with respect to shape parameters is done in Section 5. Demonstration of the proposed scheme has been made in Section 5, which consists of some illustrative examples. Section 6 describes a comparative study and the paper is concluded in Section 7.

2. Review of Weighted Nu-Spline Interpolation

This section gives a brief review of the weighted v-splines. For a detailed description, the reader is referred to [8]. Suppose that we are given data points $F_i = (x_i, y_i, z_i)$ and $v_i \geq 0$ for $i = 1,...,n$ and $w_i > 0$ for $i = 1,...,n-1$. If we let X(t) be the weighted *v*-spline interpolant to the data (t_i, x_i), Y(t) be the weighted *v*-spline interpolant to the data (t_i, y_i) and Z(t) be the weighted *v*-spline interpolant to the data (t_i, z_i), then the parametric curve

$$S(t) = \big(X(t), Y(t), Z(t)\big), \tag{1}$$

where $t_1 \leq t \leq t_n$, is the weighted v-spline interpolant. It is a C^1 piecewise cubic function:

$$S(t) \equiv S_i(t) = F_i(1-\theta)^3 + 3\theta(1-\theta)^2 V_i + 3\theta^2(1-\theta)W_i + F_{i+1}\theta^3, \tag{2}$$

where

$$\theta = \frac{t - t_i}{h_i}, \tag{3}$$

and

$$V_i = F_i + \frac{h_i M_i}{3}, \qquad W_i = F_{i+1} - \frac{h_i M_{i+1}}{3}. \tag{4}$$

It is obvious to see that the piecewise cubic function (2) holds the following Hermite interpolatory properties.

$$S(t_i) = F_i, \quad S(t_{i+1}) = F_{i+1}, \quad S^{(1)}(t_i) = M_i, \quad S^{(1)}(t_{i+1}) = M_{i+1} \Big\}, \tag{5}$$

where $S^{(1)}$ denotes first derivative with respect to t and M_i denote derivative values given at the knots t_i. This leads the piecewise cubic (2) to the piecewise Hermite interpolant $S \in C^1[t_1, t_n]$.

The parametric weighted v-spline can be computed by solving for M_i's. This can be done by imposing the continuity constraints leading to the system of equations as follows:

$$c_{i-1}M_{i-1} + (\frac{v_i}{2} + 2c_{i-1} + 2c_i)M_i + c_i M_{i+1} = 3c_i\Delta_i + 3c_{i-1}\Delta_{i-1}, \quad (6)$$

where $\Delta_i = (F_{i+1} - F_i)/h_i$, for $i = 2,...,n-1$. For given appropriate end conditions, this system of equations is a tridiagonal linear system. This is also diagonally dominant for the following constraints on the shape parameters:

$$v_i \geq 0, \ i = 1,2,....,n, \text{ and } w_i > 0, \ i = 1,2,....,n-1, \quad (7)$$

and hence has a unique solution for M_i's. As far as the computation method is concerned, it is much more economical to adopt the LU-decomposition method to solve the tridiagonal system. Therefore, for the shape parameter constraints (7), the spline solution of the weighted v-spline exists and is unique.

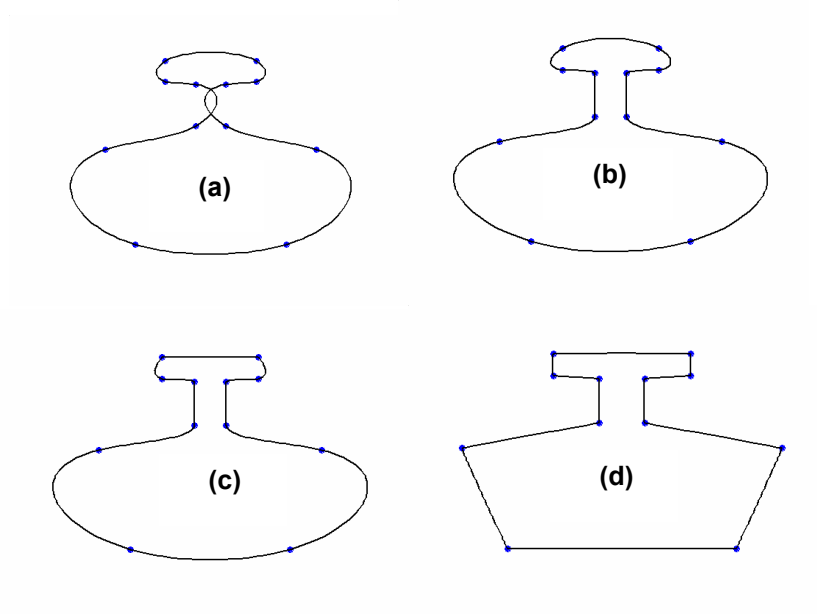


Fig. 1. (a) This is the Parametric weighted v-spline with periodic end conditions using $w_i = 1$ and $v_i = 0$ (a cubic spline case: default curve), (b) around the intervals defining the neck $w_i = 30$, and $w_i = 1$, $v_i = 0$ otherwise, (c) on the top two vertices, $v_i = 50$, around the intervals defining the neck $w_i = 30$, and $w_i = 1$, $v_i = 0$ otherwise, (d) $v_i = 50$ and $w_i = 1$ everywhere.

Demonstration: Figure1 is the parametric weighted *v*-spline interpolant to the points denoted by bullets using periodic end conditions. Interval weights w_i of 40 are used on the top and bottom intervals, while point tension factors v_i of 40 are used on the four vertices defining the 'neck'. The rest of the parameters are taken as $w_i = 1$ and $v_i = 0$.

3. C^1 Piecewise Rational Cubic Hermite Interpolant

A piecewise cubic Hermite parametric function $P \in C^1[t_o, t_n]$, with parameters r_i, $i = 1,...,n-1$, is defined for $t \in [t_i, t_{i+1}]$ by

$$p(t) \equiv P_i(t_i, r_i) = \frac{(1-\theta)^3 F_i + \theta(1-\theta)^2 (r_i F_i + h_i D_i) + \theta^2 (1-\theta)(r_i F_{i+1} - h_i D_{i+1}) + \theta^3 F_{i+1}}{1 + (r_i - 3)\theta(1-\theta)}, \quad \textbf{(8)}$$

where the notations F_i and $D_i \in R^N$ are, respectively, the data points and the first derivative values at the knots t_i, $i = 1,...,n$, with $t_1 <,...,< t_n$, $h_i = t_{i+1} - t_i$, $\theta = (t - t_i)/h_i$ and $r_i \geq 0$.

The function *P*(*t*) has the Hermite interpolation properties that

$$P(t_i) = F_i \text{ and } P^{(1)}(t_i) = D_i, \quad i = 1,...,n \quad \textbf{(9)}$$

The parameters r_i, $i = 1,...,n-1$, will be used as shape parameters to control and fine tune the shape of the curve. The denominator in (8) can be re-written as

$$(1-\theta)^3 + r_i\theta(1-\theta)^2 + r_i\theta^2(1-\theta) + \theta^3.$$

The case $r_i = 3$, $i = 1,...,n-1$, is that of cubic Hermite interpolation and the restriction $r_i \geq -1$ ensures a positive denominator in (8). The interpolant (8) can be re-written in the form

$$P_i(t_i, r_i) = R_o(\theta, r_i)F_i + R_1(\theta, r_i)V_i + R_2(\theta, r_i)W_i + R_3(\theta, r_i)F_{i+1} \quad \textbf{(10)}$$

where

$$V_i = F_i + h_i D_i / r_i, \; W_i = F_{i+1} - h_i D_{i+1} / r_i, \quad \textbf{(11)}$$

and $R_j(\theta, r_i)$, $i = 0,1,2,3$ are appropriately defined functions. These are rational Bernstein-Bézier weight functions, which are non-negative for $r_i > 0$. Thus in R^N, $N > 1$ and for $r_i > 0$, we have:

Proposition 1: (Convex Hull Property): The curve segment P_i lies in the convex hull of the control points $\{X_i, V_i, W_i, X_{i+1}\}$.

Proposition 2 (Variation Diminishing Property): The curve segment P_i crosses any (hyper) plane of dimension $N-1$ no more times than it crosses the control polygon joining F_i, V_i, W_i, F_{i+1}.

Proof: The proof is left for the sake of brevity.

The rational cubic (8) can be expressed in the form.

$$P_i(t, r_i) = l_i(t) + e_i(t, r_i), \tag{12}$$

where

$$l_i(t) = (1-\theta)F_i + \theta F_{i+1}, \tag{13}$$

and

$$e_i(t, r_i) = \frac{h_i\theta(1-\theta)\left[(D_i - \Delta_i)(1-\theta) + (\Delta_i - D_{i+1})\theta\right]}{1 + (r_i - 3)\theta(1-\theta)}. \tag{14}$$

This immediately leads to:

Proposition 3 (Interval Tension Property): For given fixed (or bounded) D_i, D_{i+1}, the rational cubic Hermite interpolant (12) converges uniformly to the linear interpolant (13) on $[t_i + t_{i+1}]$ as $r_i \to \infty$ i.e.

$$\lim_{r_i \to \infty} \|e_i\| = \lim_{r_i \to \infty} \|P_i - l_i\| = 0, \tag{15}$$

($\|\cdot\|$ denotes the uniform norm). Moreover, the component functions of e_i tend to zero monotonically, both uniformly and point-wise on $[t_i, t_{i+1}]$.

4. GC^2 Rational Cubic Spline

We now follow the familiar procedure of allowing the derivative parameters D_i, $i = 0,2,..., n$, to be the degrees of freedom, which are constrained by the imposition of GC^2 continuity conditions. These GC^2 conditions give the linear system of *consistency* equations

$$h_i D_{i-1} + \{h_i h_{i-1} v_i / 2 + h_i (r_{i-1} - 1) + h_{i-1}(r_i - 1)\} D_i + h_{i-1} D_{i+1} = h_i r_{i-1} \Delta_{i-1} + h_{i-1} r_i \Delta_i, i = 0,..., n-1. \quad (16)$$

With appropriate end conditions D_0 and D_n, (16) is a tridiagonal linear system in the unknowns D_i, $i = 0,...,n-1$. Assume that

$$r_i \geq r > 2,\ v_i \geq 0, \quad (17)$$

then the tridiagonal linear system is strictly diagonally dominant and hence has a unique solution which can be easily calculated by use of the tridiagonal *LU* decomposition algorithm. Thus a geometric rational cubic spline interpolant can be constructed with tension parameters r_i, $i = 0,...,n-1$, and v_i, $i = 1,...,n$, where the special cases are can be obtained with special values of parameters.

5. Shape Control Ananlysis

We now examine the behaviour of the geometric rational spline interpolant with respect to the tension parameters r_i and v_i in the following propositions.

Proposition 4 (Global Tension Property): Let $l \in C^0[t_0, t_n]$ denote the piecewise linear interpolant defined for $t \in [t_i, t_{i+1}]$ by $l(t) = l_i(t)$, see (13). Suppose that r_i's are as in (17) and v_i's are assumed fixed. Then the rational spline interpolant converges uniformly to l as $r_i \to \infty$, i.e. on $[t_0, t_n]$

$$\lim_{r_i \to \infty} \|P - l\| = 0. \quad (18)$$

Proof: The proof is left for the reader.

Proposition 5 (Local Tension Property)

Let $v_i \geq 0$, and $r_i \geq r \geq 2, \forall\ i$ and consider an interval $[t_k, t_{k+1}]$ for a fixed $k \in \{0,...,n-1\}$. Then, on $[t_k, t_{k+1}]$, the rational spline interpolant converges uniformly to the line segment l_k as $r_k \to \infty$ and v_k is kept fixed, i.e.

$$\lim_{r_k \to \infty} \|P_k - l_k\| = 0. \quad (19)$$

Proof: The proof is based on in Proposition 5 and is left for the reader.

Remark 6: Increasing r_i tightens the curve both locally and globally (*c.f.* Propositions 4 and 5). For the range $2 < r_i < 3$ the rational spline produces a more flexible, i.e. *looser,* curve than the cubic spline curve, both locally and globally.

Now we look at the effects of the shape parameters v_i and consider the curve as parametric one. One can observe the following tension properties:

Point Tension: For fixed *i=k,* the *k*th equation from the system of equations (16) results as:

$$\lim_{v_{k}\to\infty} D_k = 0 . \qquad (20)$$

Thus the curve at the point P_k will appear to have a *corner.*

Interval Tension: Similarly as above large values of v_k and v_{k+1} (where r_i, $i = k-1, k, k+1$ are regarded as fixed) cause D_k and D_{k+1} to approach zero. This behavior tightens the curve in the interval $[t_k, t_{k+1}]$.

Demonstration

The tension behaviour of the rational cubic spline interpolants is illustrated by the following simple examples for data sets in R^2. Unless otherwise stated, in all the Figures the parameter v_i will be assumed as zero $\forall$ *i* and the parameters r_i as 3 for all *i.*

Figure 2 shows (from left to right) the effect of a progressive increase in global tension with *r* = 3 (the cubic spline case), 5 and 50. The effect of the high tension parameter is clearly seen in that the resulting interpolant approaches piecewise linear form. Figure 3 illustrates the effect of progressively increasing the value of the tension parameter as *r*=3, 5 and 50 in two intervals (around the neck), whilst elsewhere the tension parameters are fixed equivalent to 3. Figures 4 demonstrate the result of Remark 8 regarding the achievement of a *looser* curve than a cubic spline curve; the first curve of the figure is a cubic spline curve whereas the second curve and the third curve show the local behavior (at bottom interval) against the value 2.5 and 2.1 respectively. The same behaviour is also valid for global curves.

Figure 5 illustrates (from left to right) the effect of progressively increasing the values of the point tension parameters v_i 's = 0, 5, 10 at the bottom of the neck. Similarly, Figure 6 shows the global tension effect due to progressive increases v_i 's = 0, 5, 100

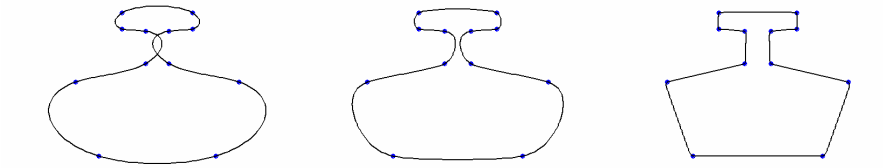


Fig. 2. A progressive increase in global tension with $r = 3$ (the cubic spline case), 5 and 50.

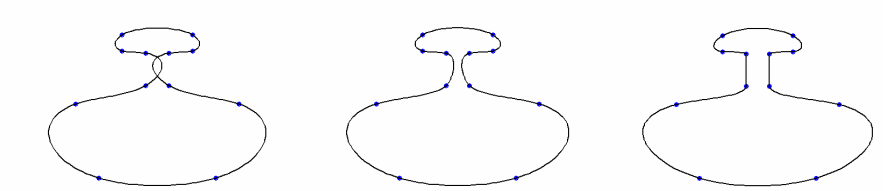


Fig. 3. The effect of progressively increasing the value of two tension parameters as r_4 =3, 5 and 50 in two intervals (around the neck).

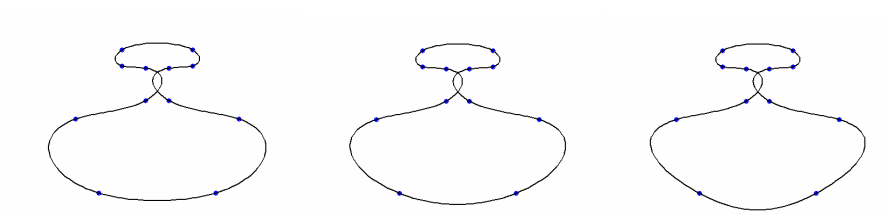


Fig. 4. The achievement of *looser* curves than a cubic spline curve.

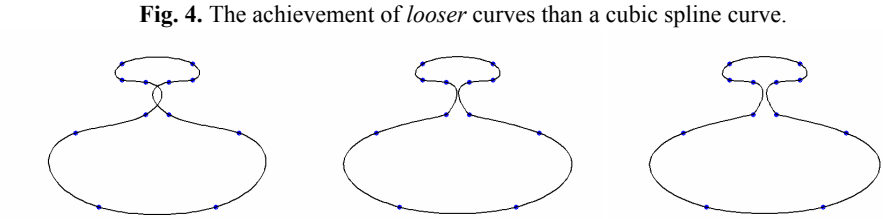


Fig. 5. The effect of progressively increasing the values of the point tension parameters v_i 's = 0, 5, 10 at the bottom of the neck.

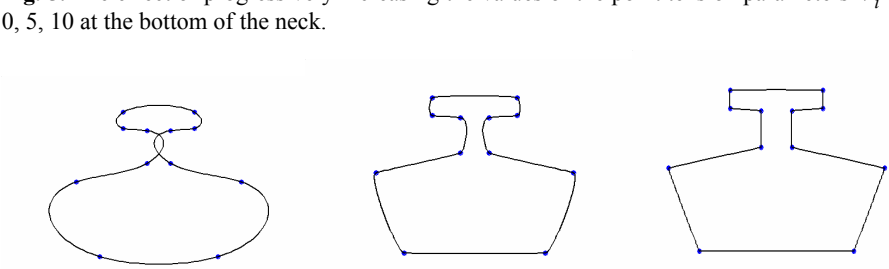


Fig. 6..The global tension effect due to progressive increases v_i 's = 0, 5, 100.

6. Comparative Study

Since the proposed curve design method has been constructed to provide a better alternative to the existing weighted *v*-spline method. Therefore, a comparative analysis would be important to have an overview of both of them. In this study, a curve design method has been formulated which, in addition to enjoying the good features of cubic splines, possesses interested shape design features too. Two families of shape parameters have been introduced in such a way that one family of parameters is associated with intervals and the other with points. These parameters provide a variety of shape controls like *point* and *interval* tension. This is an interpolatory curve scheme, which utilizes a piecewise rational cubic function in its description. These rational functions enjoy ideal geometric properties like convex hull, partition of unity, and variation diminishing. The rational functions are also stitched together with geometric continuity of order two to produce a rational spline curve. A description and analysis of this spline scheme has been made for use in Computer Graphics and Geometric Modeling. The proposed spline method provides not only computationally simple alternative to the exponential based spline under tension but also provides a well controlled and smoother alternative to the well known existing method of weighted *v*-splines. The proposed method also recovers, as special cases, the cubic splines and Nu splines. The method also generalizes the *rational spline with tension*. The spline is presented in interpolatory form and is compared with weighted *v*-splines**.**

7. Concluding Remarks

A GC^2 interpolatory rational cubic spline has been developed with a view to its application in Computer Graphics, Geometric Modeling, and CAGD. It is quite reasonable to construct a rational form, which involves two families of shape parameters in exactly a similar way as in weighted *v*-spline. These parameters provide a variety of local and global shape controls like interval and point shape effects. In particular, it has been found that only one shape parameter per interval is enough when local or global interval tension is required. The shape control features are stronger than those in weighted *v*-spline. Moreover, the visual smoothness is GC^2, which is also superior to C^1 smoothness of weighted *v*-spline. The rational spline method can be applied to tensor product surfaces but unfortunately, in the context of interactive surface design, this tensor product surface is not that useful because any one of the tension parameters controls an entire corresponding interval strip of the surface. Thus as an application of GC^2 rational spline for the surfaces, Nielson's [13] spline blended methods can be adopted. This will produce local shape control, which is quite useful regarding the computer graphics applications.

Acknowledgments

The authors are thankful to the anonymous referees for the valuable suggestion towards the improvement of this manuscript. This work has been supported by the King Fahd University of Petroleum and Minerals under Project No. FT/2001-18.

References

1. Barnhill, R.E., Brown, J.H., and Klucewicz, I.M. (1978), A New Twist in Computer-Aided Geometric Design, Computer Graphics and Image Processing 8. 78-91.
2. Barsky, B.A. (1984), Exponential and Polynomial Methods for applying Tension to an Interpolating Spline Curve, Computer Vision, Graphics, and Image Processing 27, 1-18.
3. Boehm, W., Farin, G., and Kahmann, J. (1984), A Survey of Curve and Surface Methods in CAGD, Computer Aided Geometric Design, 1, 1-60.
4. Dierckx, P. and Tytgat, B. (1989), Generating the Bézier points of β-spline curve, Comp. Aided Geom. Design 6, 279-291.
5. Farin, G.E. (1996), Curves and Surfaces for CAGD, Academic Press, New York.
6. Faux, I.E., and Pratt, M.J. (1979), Computational Geometry for Design and Manufacture, Ellis Horwood, Chichester.
7. Foley, T.A. (1987), Weighted Bicubic Spline Interpolation to rapidly varying Data, ACM Trans. Graph. 6, 1-18.
8. Foley, T.A. (1987), Interpolation with Interval and Point Tension Controls using Cubic Weighted *v*-Splines, ACM Trans. Math. Soft. 13, 68-96.
9. Fritsch, F.N. (1986), The Wilson-Fowler Spline is a *v*-spline, Computer Aided Geometric Design 3, 155-162.
10. Gregory, J.A. and Sarfraz, M. (1990), A Rational cubic Spline with Tension, Computer Aided Geometric Design, North-Holland, Elsevier, Vol. 7, 1-13.
11. Mortenson, M.E. (1985), Geometric Modeling, Wiley, New York.
12. Nielson, G.M. (1974), Some Piecewise Polynomial Alternatives to Splines under Tension, in: R.E. Barnhill and R.F. Riesenfeld, eds., Computer Aided Geometric Design, Academic Press, New York, 209-235.
13. Nielson, G.M. (1986), Rectangular *v*-splines, IEEE Computer Graphics 6, 35-40.
14. Salkauskas, K. (1984), C^1 Splines for Interpolation of Rapidly Varying Data, Rocky Mtn. J. Math. 14, 239-250.
15. Sarfraz, M. (1994), A C^2 Rational Cubic Spline which has Linear Denominator and Shape Control, Annales Univ. Sci. Budapest, Vol. 37, 53-62.
16. Sarfraz, M. (1993), Designing of Curves and Surfaces using Rational Cubics, Computers and Graphics, Elsevier Science, Vol. 17(5), 529-538.
17. Sarfraz, M. (1992), Interpolatory Rational Cubic Spline with Biased, Point and Interval Tension, Computers and Graphics, Elsevier Science, Vol. 16(4), 427-430.

Determination of Cutting Direction for Minimization of Tool Retraction Length in Zigzag Pocket Machining

Byoung Keuk Kim[1], Joon Young Park[1], Hyun Chan Lee[2], Deok-Soo Kim[3]

[1] Department of Industrial Engineering, Dongguk University,
32, Pil-dong, Chung-gu, Seoul, Korea
{kbkim, jypark}@dgu.edu
[2] Department of Industrial Engineering, Hongik University,
72-1, Sangsu-dong, Mapo-gu, Seoul, Korea
hclee@wow.hongik.ac.kr
[3] Department of Industrial Engineering, Hanyang University,
San 17, Haengdang-dong, Sungdong-gu, Seoul, Korea
dskim@hanyang.ac.kr

Abstract. In this paper, we determine a cutting direction which minimizes the tool retraction length in a zigzag pocket machining. A tool retraction, in a zigzag pocket machining, is a non-productive operation in which the tool moves to any remaining regions for machining. We developed an algorithm of generating a tool retraction length in simple polygonal shapes. In the algorithm, we considered a reflex profile of cutting direction in a polygonal shape. Considering reflex profiles of cutting direction, the polygon is decomposed into several subregions, which do not need any tool retractions. Consequently, we propose a new result that the tool retraction can be minimized when a cutting direction is parallel to concave edges. Examples are shown to show the validity.

1 Introduction

In a pocket machining, two strategies are commonly used for a rough cut which are contour milling and zigzag milling. Contour milling uses successive offsets of the pocket boundary as a tool path. Each offset can be computed mainly using a Voronoi diagram, which is widely used for mesh generations, blending solid models and robot path generations. In contrast, zigzag milling is much simpler in computation because it is able to generate a tool path which connects zigzag line segments in a specified cutting direction[3,8,12].

However, there are some disadvantages for the zigzag milling. For example, in case of a complicated part which contains many islands inside of a pocket area, many tool retractions occur to move the tool for machining any remaining regions or for drilling operations. Therefore, in a zigzag milling, the cutting direction for a tool path generation is an important variable[12].

Previous works have significantly advanced the art of zigzag pocket machining. Veeramani and Gau[13] proposed a model which minimizes non-cutting path in a patch by patch machining. They formulated a TSP(Traveling Salesman Problem) to minimize the non-cutting path. Tang *et al.*[11,12] presented an algorithm to minimize

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 680-689, 2003.
© Springer-Verlag Berlin Heidelberg 2003

the number of tool retraction in a zigzag pocket machining, where the pocket area is subdivided into some traceable regions which do not have any tool retraction in a given cutting direction. However, this work contains 3 major constraints: Each of SRP(scan-reflex point) contains 2 scan line segments with respect to a given cutting direction. Every traceable region has some zigzag line segments, and each scan line segment is also a zigzag line segment. They presented a range of optimal cutting directions to minimize the number of tool retractions which can be computed in O(n log n), where n is the number of line or arc segments defining the pocket boundary. Arkin *et al.*[1,2] generated a tool path to minimize the number of tool retraction in a zigzag pocket machining. This work proposed MG(machining graph) which connects intersection points of pocket boundary by a given cutting direction to minimize the number of tool retractions. They showed that the optimization problem related to zigzag pocket machining with holes is NP-hard and indicated that Tang's solution is not practical.

In this paper, we focus on the length of tool retraction. So, we determine a cutting direction which minimizes the tool retraction length. To simplify the problem, we developed an algorithm to calculate the tool retraction length in a simple polygonal shape. The proposed algorithm is composed of two parts: firstly, subdividing a given polygon into several subregions and secondly, calculating the tool retraction length. In our research, we only deal with a rough cutting.

2 Preliminaries

In Fig. 1, a tool retraction is defined as a non-cutting operation in which the tool moves to any remaining regions for machining in a zigzag pocket machining. So, it is a non-productive operation which increases the cutting time. Since the tool retraction occurs at vertices, we need to identify potential vertices where tool retractions might occur. These positions could be found easily if a given region is divided into subregions using SRP(scan reflex point)[1,2,11,12].

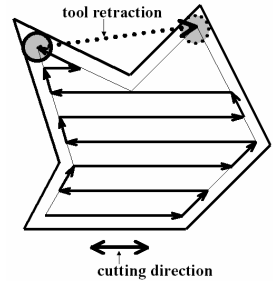


Fig. 1. Tool retraction in a zigzag pocket machining

In Fig. 2, tool retractions can occur as (a) or (b). In Fig. 2(a), the number of tool retraction is one and the drilling operation is necessary. In Fig. 2(b), the number of

tool retraction is two, but the drilling operation is not necessary because the tool enters the machined area. In these cases, we must consider both the tool retraction length and the number of tool retraction with drilling operation. However, to simplify the problem, we only consider the case where the number of tool retraction in a given cutting direction is identical through out the machining. Fig. 3 denotes such cases.

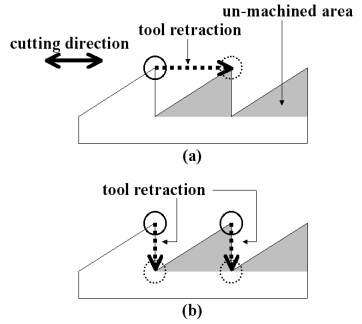


Fig. 2. Relationship of the number of tool retraction and drilling operation

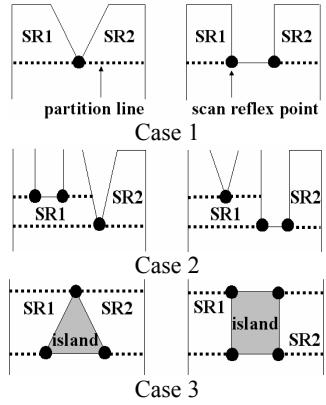


Fig. 3. Cases of two subregions on one side of partition line

3 Proposed Algorithm

3.1 Generation of Subregions(SR)

To divide a given region into several subregions, we firstly generate horizontal lines on the vertices as Fig. 4. Then, we can find vertices of which directly connected the same side of the horizontal line. Among these vertices, it is defined as a RP(reflex

point) if it is contained in a reflex region[3] and otherwise, a EP(end point). And it is defined as a IP(intersection point) when a point is generated by intersecting a horizontal line through RP with boundary edges. Therefore, we can divide a given region into several subregions using EP, RP, and IP.

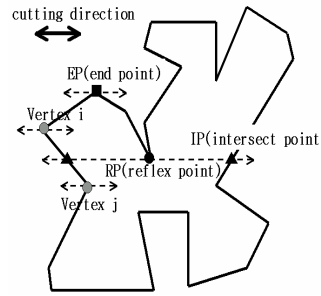


Fig. 4. EP, RP and IP

Consequently, each subregion is composed of EPs, RPs, and IPs. If these points are connected in the order of vertices for the polygon, the shape of each subregion always becomes a triangle or a quadrilateral. Fig. 5 shows 7 subregions which are represented in triangles or quadrilaterals after Fig. 4 is divided. As a result, EPs, RPs and IPs become positions in which tool retractions occur and machining begins.

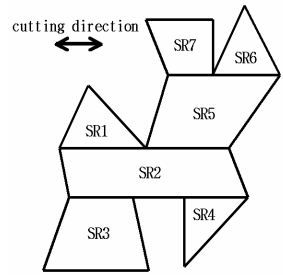


Fig. 5. Subregions of triangle and quadrilateral shape

Fig. 6 shows a graph of subregions. A subregion can be represented as 2 nodes and each node means a set of EP(s), IPs, and IP,RP(RP,IP). For example, because SR1 is composed of EP, IP, and RP, it is represented in EP and IP,RP node. In Fig. 6, a node has single linked relationship or double linked relationship with another node. A single linked relationship means that 2 subregions share a point. Because 2 nodes connected with double linked relationship mean to be contained in an identical subregion, double linked relationship is selected primarily among connecting nodes. A node can have “2 single linked relationships and 1 double linked relationship” or “1 double linked relationship”. So, it is not able to present continuously 2 single linked

relationships or 2 double linked relationships in connecting nodes without any tool retraction. For example, "(EP of SR1)→(IP, RP of SR1)→(IP, IP of SR2)→(RP, IP of SR5)" does not exists, only "(EP of SR1)→(IP, RP of SR1)→(IP, IP of SR2)→(IP, IP of SR2)" exists.

Fig. 7 shows graphs of Fig. 3 in which SRi and SRi′ represent two nodes of a subregion. In Fig. 7, we observe that the number of tool retraction occur only once in each case if tool traverses 4 nodes(SR1, SR1′, SR2, SR2′), previous/next node and next/previous node regardless of ordering. For example, SR2→SR2′→(next)/previous→tool retraction→SR1′→SR1→(previous)/next is a possible route in case 2. Because any graph of subregions can be represented as the combination of 3 graphs in Fig. 7, we can identify the number of tool retraction using the graph.

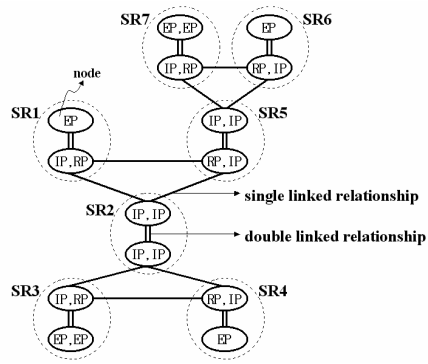


Fig. 6. A graph of subregions

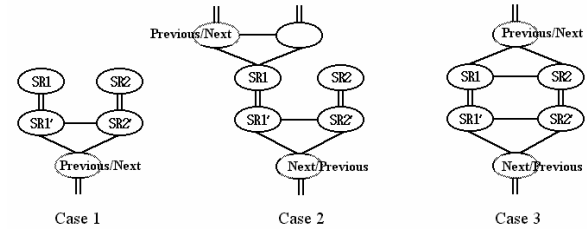


Fig. 7. Basic components for the graph of subregions

3.2 Calculation of Tool Retraction Length

Generally, it is well known that a zigzag tool path generation can be simplified when the machining begins from the boundary of the given region[2,3,8,12]. With this result, we consider EP of each subregion as the starting point for machining. In a

graph of subregions, every node which has only a double linked relationship is a starting point for machining.

Then, we can determine a sequence which enables subregions to be machined without any tool retractions. Using the relationship of subregions, it can be identified which tool retraction does not happen during a sequence[7]. In Fig. 8, the dotted line means a progressing path from SRi to SRj without any tool retraction. So, subregions passed by the line are considered as machined areas. Therefore, tool retraction occurs when the tool moves to any remaining subregions for machining. Drilling operation must be minimized because it increases the cutting time. In the calculation of tool retraction lengths, we regard the drilling operation as a constant length.

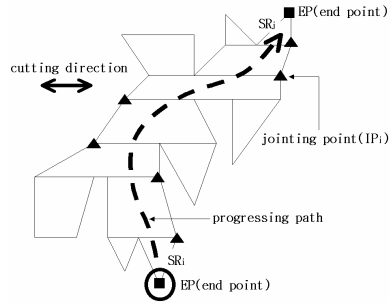


Fig. 8. A progressing path occurred between SRi and SRj

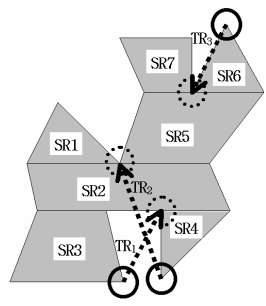
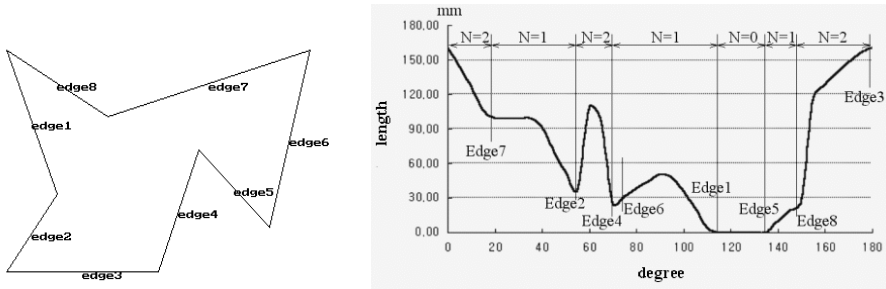


Fig. 9. Three tool retractions(TR1, TR2, TR3) in subregions

The process of tool retraction composes 3 steps: tool lift, rapid move, and tool down. We assume that the length of tool lift and tool down are identical in all phases of tool retraction. Fig. 9 shows tool retractions(TR1, TR2, TR3) occur when the starting point of machining is EP of SR1 in Fig. 6. Because we minimize tool retraction lengths, subregions which do not have any tool retractions are grouped in a sequence of subregions. In this problem, the number of tool retraction turned out to be the same as Tang's results[11,12]. And we observe that the tool retraction length is different even though the number of tool retraction is equal.

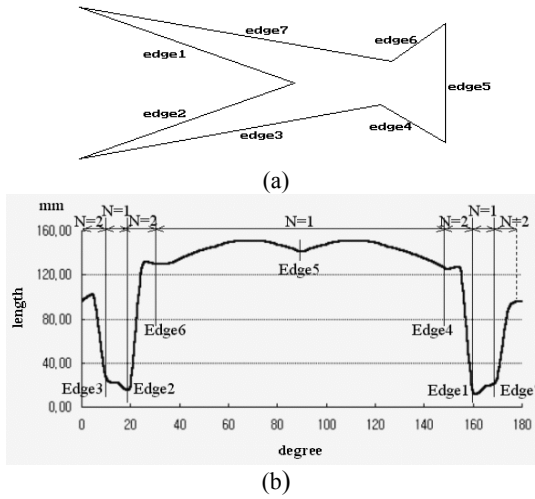
4 Analysis of Tool Retraction Length

In this section, we observe how the tool retraction length changes. We calculate the tool retraction lengths of 5 arbitrary polygonal shapes. In Fig. 10(a), we calculate the tool retraction length when cutting directions change from 0° to 180°. Fig. 10(b) shows the variation graph of the tool retraction length. In the graph, N denotes the number of tool retractions.



(a) (b)

Fig. 10. Concave polygon-1 and a variation graph of the tool retraction length



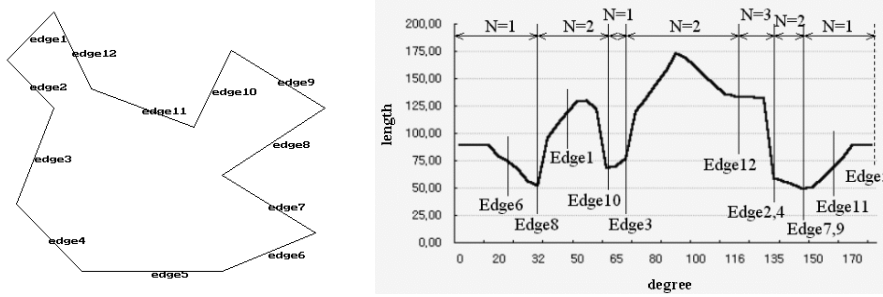
(a)

(b)

Fig. 11. Concave polygon-2 and a variation graph of the tool retraction length

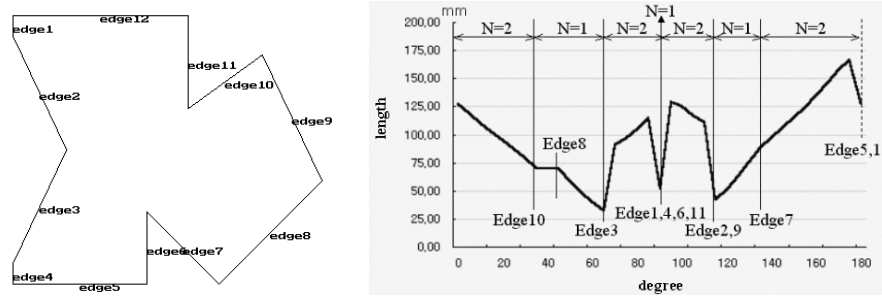
We can notice that the tool retraction length is minimized in boundaries of the range where the number of tool retraction is the same. Namely, the length of tool retraction is minimized in boundaries of ranges, such as edge1, edge2, edge4, edge5, edge7 and edge8 in Fig. 10(b). Fig. 11(b) shows a graph variation of the tool retraction length about Fig. 11(a).

The tool retraction length is minimized in boundaries of ranges, such as edge1, edge2, edge3, edge4, edge6 and edge7 in Fig. 11(b). Fig. 12(b), Fig. 13(b) and Fig. 14(b) show the variation graphs of the tool retraction length for Fig. 12(a), Fig. 13(a) and Fig. 14(a). Also, the length of tool retraction is minimized in boundaries of ranges where N remains the same in Fig. 12(b), Fig. 13(b) and Fig. 14(b).



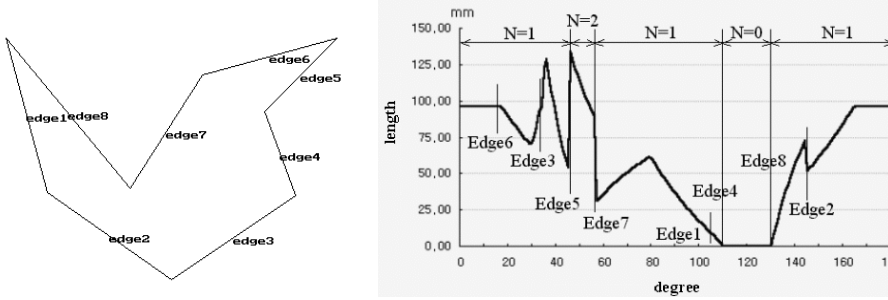
(a) (b)

Fig. 12. Concave polygon-3 and a variation graph of the tool retraction length



(a) (b)

Fig. 13. Concave polygon-4 and a variation graph of the tool retraction length



(a) (b)

Fig. 14. Concave polygon-5 and a variation graph of the tool retraction length

Consequently, a minimum length of tool retractions occurs in one of two edges which are boundaries for each range. These boundaries mean concave edges. Fig. 15 shows cases where a minimum length of the tool retractions exists in boundaries of range.

5 Patterns of Tool Retraction Length

We observe that cause for the variation of tool retraction length relates to change the positions of EP, RP and IP which consists of a subregion. In each range where N remains the same, the length of tool retraction changes as Fig. 15.

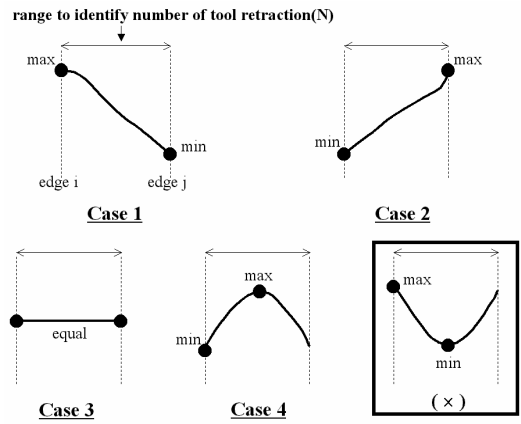


Fig. 15. Cases which a minimum length presents in boundaries of range

So, the shortest length of tool retraction exists in boundaries of the range. Boundaries of the range are concave edges in a given polygon. It means that the cutting direction for reducing the tool retraction length is determined as a direction which is parallel to a concave edge.

6 Conclusions and Future Works

In this paper, we determined a cutting direction which minimizes the tool retraction length in a zigzag pocket machining. We also proposed an algorithm for calculating the length of tool retraction. The proposed algorithm is composed of 2 parts: generation of subregions, and calculation of the tool retraction length. In the proposed algorithm, we formulated a simplified TSP in determining the sequence of the tool paths and restricted drilling operations as possible.

Using the proposed algorithm, we can calculate the tool retraction length in any simple concave polygons and analyze the variation of tool retraction length.

Consequently, we can state that the minimum length of tool retractions presents in one of 2 edges which are boundaries of range where the number of tool retraction remains the same. These boundaries are concave edges in a given polygon. So, the minimum length of the tool retraction exists when a cutting direction is parallel to one of concave edges in a given polygon. In the future, we need to study the tool retraction length and the number of tool retraction together.

References

1. Arkin, E. M., Fekete, S. P., Mitchell, S. B.: Approximation algorithms for lawn mowing and milling. Computational Geometry, Vol. 17 (2000) 25-50
2. Akrin, E. M., Held, M., Smith, C. L.: Optimization problem related to zigzag pocket machining. Algorithmica, Vol. 26 (2000) 197-236
3. Held, M., On the computational geometry of pocket machining. Springer-Verlag, New York (1991)
4. Held, M., Arkin, E.: Letter to the editor: an algorithm for reducing tool retractions in zigzag pocket machining. Computer-Aided Design, Vol. 32 (2000) 617-619
5. Jung, J. Y., Ahluwalla, R. S.: Feature-based noncutting tool path selection. Journal of Manufacturing Systems, Vol. 13 No. 3 (1994) 165-176
6. Kim, B. K., Park, J. Y.: An algorithm for reducing the tool retraction length in zigzag pocket machining. Journal of the Korean Institute of Industrial Engineers, Vol. 28 No. 2 (2002) 128-138
7. Kim, B. K., Park, J. Y.: Determination of cutting direction for tool path minimization in zigzag milling operation. Journal of the Korean Institute of Industrial Engineers, Vol. 27 No. 1 (2001) 69-88
8. Park, S. C., Choi, B. K.: Tool-path planning for direction-parallel area milling. Computer-Aided Design, Vol. 32 (2000) 17-25
9. Park, S. C., Chung, Y. C.: Offset tool-path linking for pocket machining. Computer-Aided Design, Vol. 34 (2002) 299-308
10. Suh, S. H., Shin, Y. S.: Neural network modeling for tool path planning of the rough cut in complex pocket milling. Journal of Manufacturing Systems, Vol. 15 No. 5 (1996) 295-304
11. Tang, K.: On minimum traceable partitioning and minimum tool retractions. Computer-Aided Design, Vol. 32 (2000) 621-624
12. Tang, K., Chou, S. Y., Chen, L. L.: An algorithm for reducing tool retractions in zigzag pocket machining. Computer-Aided Design, Vol. 30 No. 2 (1998) 123-129
13. Veeramani, D., Gau, Y. S.: Model for tool-path plan optimization in patch-by-patch machining, Int. J. Prod. Res., Vol. 36 No. 6 (1998) 1633-1651

Digital Styling for Designers: Sketch Emulation in Computer Environment

Seok-Hyung Bae[1,3], Won-Sup Kim[2], and Eun-Sook Kwon[2]

[1] Virtual Reality Research Center, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
bae@vr.kaist.ac.kr
http://vr.kaist.ac.kr/∼bae

[2] Department of Industrial Design, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
{iron,eskwon}@mail.kaist.ac.kr

[3] Virtual System Laboratory, Gifu University, 1-1 Yanagido, Gifu, 501-1193, Japan
bae@vsl.gifu-u.ac.jp

Abstract. The digitalization of product-styling tools for the early design-development stage has focused on using raster-type graphics S/Ws. These kinds of S/Ws enable designers freely to generate ideas, and enhance them based on their creativity and aesthetic senses. However, resultants from raster-type tools are no more than images, which are hardly converted to digital forms required by downstream processes. Thus, it still remains as an obstacle for both designers and engineers to work harmoniously and simultaneously in the new-product development process through concurrent engineering. This paper proposes a new digital-styling tool for designers by emulating a design sketch. The research process is summarized as: 1) observing designers' sketch behaviors and their works, 2) analyzing designers' sketch behavior patterns for extracting sketch elements, and 3) mapping them to digital forms. The proposed digital-sketch method not only allows designers to perform their creative work, but also provides digital connectivity for concurrent engineering.

1 Introduction

Digitalizing has become a general tendency in every field of the world. As industries make efforts to digitalize all the *product-development process* in view of concurrent engineering, the design and development process is also changing dramatically.

Focusing on *product styling* with special emphasis on external shapes, the digitalizing effort can be categorized with a keyword, *dimension*. For the early phase consists of the *concept selection* and *embodiment design*, where idea generation and hybridization are repeated until a satisfactory styling concept is built up, *raster-type* 2D graphics S/Ws are used. For the *detailed design* where every component specification and technical drawings are made, RE (reverse engineering) and CAD S/Ws are used.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 690–700, 2003.
© Springer-Verlag Berlin Heidelberg 2003

However, the digitalization for the early-phase design development is somewhat controversial especially from a concurrent-engineering point of view. Undoubtedly, raster-type 2D graphics S/Ws are quite compatible with the designer's work in the concept selection and embodiment design stages: 1) the *rapid* generation of various design alternatives, 2) the *flexible* hybridization of previous ideas, 3) the *sensuous* working style, etc. But, raster-type tools are basically *image-based*, and inevitably suffer serious problems as follows: 1) the difficulty of *touch editing*, 2) the hardship of keeping a *touch history*, 3) no *connectivity* to downstream *digital* processes, etc.

Above all, no digital connectivity is the acutest problem for the harmonious concurrent engineering on the whole product-development process. The styling alternatives in the raster-image form must be converted to computer models with CAD S/Ws, separately. Transforming design works into CAD models unavoidably introduces not only numerical (or dimensional) errors but also the unwanted change of the feeling (or impression) of styles. Furthermore, when receiving feedback from downstream processes, the partial modification can hardly be done.

Therefore, this paper proposes a new digital-styling tool for designers being able to express their ideas naturally by emulating a *design sketch* in computer environment while keeping a digital connectivity. In order to identify the designer's sketch, we used several information gathering methods: 1) a *simple observation* of numerous design-sketch works, 2) *disposable-camera* technique to gather designers' working environment, and 3) in-depth *interviews* with professional industrial-designers. Based on the design-sketch elements and behavior patterns extracted from the factor analysis, digital functionalities (or solutions), are mapped with proper assumptions and the *vector-type data handling scheme*. The proposed method not only allows designers to perform their creative work sensually with it by adopting user-friendly interface based on their traditional freehand drawings, but also provides a digital connectivity with downstream processes by vector-type data handling.

This paper is organized as the following order: after related work briefly summarized in Section 2, a design sketch is considered as the designer's versatile tool in Section 3. Based on the analysis of design sketch in Section 4, the solution mapping for the digital-sketch emulation is given in Section 5. The user's experience and feedback about a prototype program are shown in Section 6 followed discussions and conclusions in Section 7 and Section 8, respectively.

2 Related Work

The pen-based interface has been a common human-computer interface. It can highly increase the level of intimacy when combined with 2D graphics tools [13][24] because of their inherent analogy. Many studies tried to use freehand sketch for CAD modeling [8][12][20][21][23][25]. While almost sketch-related research focuses on feature recognition using primitive recognition techniques, there were little studies about sketch stroke itself: Baudel [3] suggested the *mark-*

based spline modification where an initial spline can be updated in real-time by the user's successive strokes. It looks a very intuitive tool for designers or animation creators because of its resemblance to physical behaviors. Arvo and Novins [1] proposed so called the *fluid sketch* based on the continuous recognition of freehand sketches and optimal-shape morphing. It also sounds fascinating, but can distort the user's intension because the number of geometric primitives and constraints is limited. We think the fluid sketch will be useful when adopted into engineering-based S/Ws rather than styling-purposed S/Ws. Roth-Koch [22] made overtures of a *B-spline fuzzy array* concept. However, this method has the difficulty of misunderstanding the intention of designers because of its off-line scheme using scanned images.

3 Sketch: Designer's Principal Means of Thinking

There have been great pains to choose two keywords explain the nature of sketch, *guide* and *freehand*, despite of its definitions widely used as follows: "a rough draught of a composition or part of a composition, made in order to satisfy the artist himself on certain point of scale, composition, lighting, etc. [17]" or "a rough drawing or painting in which an artist notes down his preliminary ideas for a work that will eventually be realized with greater precision and detail [4]". Although the meaning of *rough* could remain in above definitions, we wanted to explore more *tangible* words commonly used in the design field as follows:

Definition 1. *A* design sketch *is a* guide *drawn* freehand *for a finished work.*

In virtue of distinctive advantages of sketch as follows: 1) the *simplicity* of its way and equipment, 2) the *speed*, 3) the *easiness* to catch ideas, etc., sketch has been widely used for creative works–"the designer's principal means of thinking [11]", "The act of sketching has long been embraced by architects as a versatile tool for exploratory design [23]". In real product styling, the practical uses of sketch are as follows: 1) the *concept description* with texts, 2) the *shape description* with color, texture, and shading effects, 3) the *layout description* with a scale, and 4) the *final product description* with full details.

As sketch is versatile, the advantages of digitalizing it are many as: 1) the almost merits of a physical sketch can be inherited, 2) some weak points–the inaccuracy problem caused by delicate hand shaking, imperfect proportion or perspective, annoying cleaning up useless lines in the end–can be complemented, 3) the problems of storage, retrieval, and modification of physical sketches can be settled or improved, and 4) designers can easily fit the new digital tool.

4 Analysis of Design Sketch

To digitalize design sketch, we conducted a wide range of observations for identifying how designers make design sketches, and how their working environment looks like. We collected numerous design-sketch works, videotaped professional

industrial-designers, and interviewed them. Throughout careful observations on enormous materials, we extract several patterns on design-sketch elements and behaviors. The major findings on design-sketch elements are as follows:

- Designers do not carry a specific set of geometry primitives. Because they are always dealing with new shapes, it is impossible to work with a limited numbers of primitives.
- A complicated freeform curve is presented with several simple curves. In most cases, the simple unit curve does not have a convexity change.
- In design sketches in 3D perspective views, arbitrarily rotated ellipses are frequently used. An ellipse is generated through one stroke because it is difficult to construct by connecting open curves.
- Straight-lines are rarely used only but in *orthographic* views or as *support* lines because they can cause *visual illusion* –there has been much research on *visual illusion* or *Gestalt Theory* [15][16].
- An exact circle, which is a special case of an ellipse, is limitedly used in mainly orthographic views.

Based on above observations, two propositions related design-sketch elements are made as follows:

Proposition 1. *There are two design-sketch elements: 1) a smooth curve without an inflection point, and 2) an arbitrarily rotated ellipse.*

Proposition 2. *There are two design-sketch support elements: 1) a straight line, and 2) an exact circle.*

Following are remarks on design-sketch behaviors throughout our observation:

- Drawing a single curve is composed of several consecutive actions as follows: 1) an *air-sketch*, 2) repetitive touches, 3) a decisional stroke, and 4) a cleaning-up
- An air-sketch is the process accustoms the arm's movement to draw the curve intended (it is quite similar to the golfer's putting in the air).
- Designers settle a final curve with a dark line when its shape is embodied throughout touch repetition with a light line.
- Using *see-through* (grid) box is in common for sketching 3D shapes in a perspective view [6].
- A semi-tracing paper is used when creating new shapes starting from existing sketches.

Like the design-sketch element case, the several propositions about design-sketch behaviors are given as below:

Proposition 3. *Drawing* dimensionally *accurate curves by controlling a human body, is primarily difficult.*

Proposition 4. *The designer's sketching process is a kind of adaptation with touch repetition based on visual feedback.*

Proposition 5. *When drawing a new curve, the existing marks of previous scribbles function as guides.*

Proposition 6. *The clean-up process is essential for acquiring an intended curve.*

5 Design-Sketch Emulation in Digital Form

In this section, our solution for a digital design-sketch tool is developed. Two major directions are as follows: 1) the interaction between the system and user has to be emulated as a physical sketch as possible, and 2) all the scribbles are treated in the form of mathematical entities not raster images.

5.1 Design-Sketch Element Mapping

Shown in Fig. 1 is the solution mapping related to design-sketch elements. A smooth curve without an inflection point is interpreted as *simply tensioned-smooth spline*, and mapped to a *cubic* Bezier curve (we thought that the degrees-of-freedom of a cubic Bezier curve are enough for the designer to fully express curves intended). An ellipses is mapped to a quadratic NURBS curve composed of four rational-quadratic Bezier curves [18]. Under the assumption that users themselves explicitly change the mode of elements during sketching for avoiding vagueness, the two support elements, a straight line and exact circle, are mapped to a linear Bezier curve and a quadratic NURBS curve, respectively.

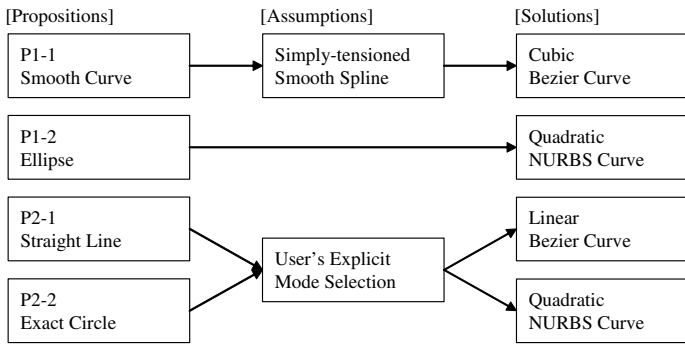


Fig. 1. Design-sketch element mapping

5.2 Design-Sketch Behavior Mapping

In this section, the analyzed designers' sketching behaviors are synthesized into the sketching behavior map. The four propositions related to the designer's behavioral features are mapped to seven solutions based on the vector-type data treatment as shown in Fig. 2.

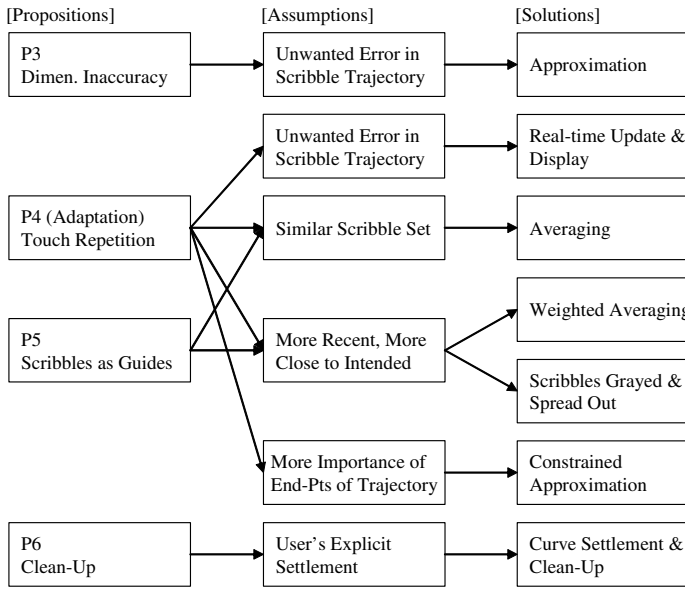


Fig. 2. Design-sketch behavior mapping

Whenever the pen-point changes, the trajectory of a scribble is approximated to the corresponding mathematical model. A trajectory for a smooth curve is fitted to a cubic Bezier curve while the end points are fixed [5][9][18]. The approximation somewhat compensate the dimensional inaccuracy of human drawing by eliminating delicate waves caused by hand shaking–it is a kind of low-frequency filtering or *fairing*. A trajectory for an ellipse is first approximated to general-quadratic curve [7][10], and then converted to a quadratic NURBS curve (only when the general-quadratic curve is discriminated as a *real ellipse* among nine possible types of it [26]). The proposed method enables designers to create a satisfactory curve in their mind by allowing repetitive scribbling. When the designer decides if overlapped scribbles form an embodied shape enough to shrink a final curve, the settlement is executed with his/her explicit order. We assumed that more recent scribble is closer to an intended curve than old ones because the designer can adapt his/her drawing while receiving visual feedback. Hence, the settled curve, $\bar{\mathbf{c}}(t)$, is a *weighted average* of a set of scribble curves, $\{\mathbf{c}_i | i = 0, \ldots, s-1\}$, given as:

$$\bar{\mathbf{c}}(t) = \sum_{i=0}^{s-1} w_i \mathbf{c}_i(t) \tag{1}$$

where $w_i = \frac{b^i(b-1)}{b^s-1}$; $b(\geq 1)$ is the ratio of weights of two consecutive levels. The designer is able to expect the resultant curve before settlement using information

provided in the forms of scribbles' display attributes, *darkness* and *width* –as the number of scribbles increases, the previous scribbles are grayed and spread out. The gray level of the i^{th} scribble, $g[i]$, is calculated as:

$$g[i] = (g_{max} - g_{-\infty})(b_g^{i-s+1} - 1) + g_{max} \tag{2}$$

where g_{max} is for the latest (or current) scribble, $g_{-\infty}$ is the lower-bounded gray level ($g[i] \rightarrow g_{-\infty}\ as\ i \rightarrow -\infty$), and $b_g(\geq 1)$ is the ratio of two consecutive gray levels. Similarly, the width of the i^{th} scribble is,

$$z[i] = (z_{-\infty} - z_{min})(1 - b_z^{i-s+1}) + z_{min} \tag{3}$$

where z_{min} is used for the latest scribble, $z_{-\infty}(z[i] \rightarrow a_{-\infty}\ as\ i \rightarrow -\infty)$, is the upper-bounded width, and $b_z \geq 1$.

Given in Fig. 3 are illustrative examples of sketching a smooth curve and ellipse.

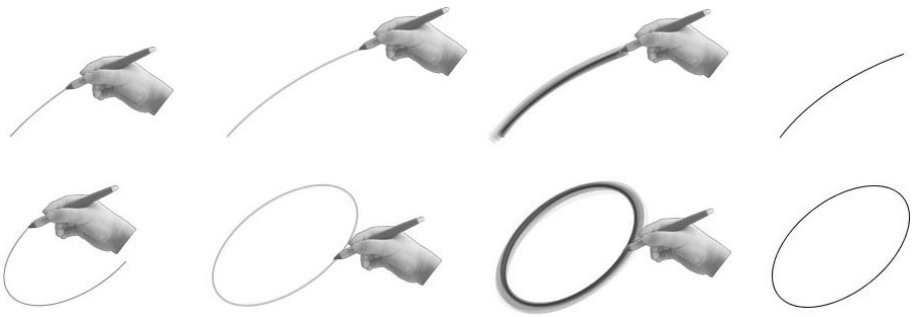


Fig. 3. Sketching simple smooth curve and ellipse

5.3 Implementation

A prototype is implemented as a *Java Applet* program using Java™ 2 Platform Standard Edition (J2SE™), Java 3D™ API 1.3, and JAMA (Java Matrix Package) [14]. The main functionalities are focused on only drawing four design elements: The user can sketch curves using one hand, and input explicit orders (key-in) using the other hand simultaneously such as the change of design-sketch elements, the settlement of a curve, the unwanted scribble removal, the initialization of the program, etc. We tested the program with a commercial tablet (WACOM Intuos™2 $9'' \times 12''$), and our Java Applet is available at http://vr.kaist.ac.kr/˜bae.

6 User Experience

Many professional industrial designers and graduate students tested our prototype. All of them became very skillful to use the program only with several trials (see Fig. 4). Especially they were favorable in following points of view: 1) an intuitive way to create curves with their adaptation, 2) an intimate scribbling guide with existing scribbles grayed and spread out, 3) a natural way to create an ellipse (see Fig. 3), 4) a smooth nature of resultant curves, etc.

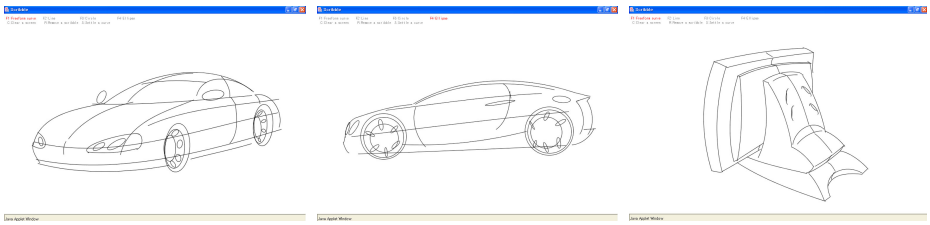


Fig. 4. Selected design-sketches using prototype

They remarked on many features for our program's improvement, which are listed up as below:

- Some mentioned about the direction of scribbling. They wanted our program to allow alternating of the start and end point for a smooth curve, and both clockwise and counterclockwise scribbling for an ellipse.
- Many pointed out that touch pressure is also important for natural sketching (in fact, it is one of our next step's research issues).
- A few users commented an unnecessary explicit settlement. For example, a new scribble is quite different from previous ones, the curve settlement can be automatically executed.
- Some suggested not to throw away the scribbles after curve settlement for using them as guides for another shape.
- Many people commented on editing functionalities and GUIs. These kinds of comments are supposed to be useful when developing regular S/Ws later.

7 Discussions

Our strategy of *vectorizing* a scribble trajectory is a kind of *approximation*. We considered if the vectorized curve can be differ from the designer's intention. In spite of this concern, nobody complained of it. Rather, they liked the smooth appearance of resultant curves throughout approximation. It can be thought that the adaptation based on visual feedback is more important than numerical accuracy.

No matter how the designer tries, the adaptation will not be possible if the degrees-of-freedom the model curve has are not enough. We chose a cubic Bezier curve as the model curve for a smooth curve, and it looks proper so far because there were no complaints among product designers (however, usual graphic designers require more flexible curves for their work). For finding the *optimal* degree, a quantitative study will be followed. Similarly, we are now planning to find optimal parameters for the calculating weighted average, scribbles' gray level and width (in fact, these values can be customized by users as treated as S/Ws' options).

Although we analyzed that one of the major design-sketch elements is a smooth curve *without* an inflection point in Section 4, we did not impose any actual convexity constraint when approximating a scribble trajectory to a cubic Bezier curve (the convexity imposing introduce a NLP problem [19]). We intend to test this strong constraint imposing, but which possibly cause the distortion of the designer's intension (almost designers did not agree with this idea).

8 Conclusions

In this paper, a new digital-styling tool for the designer's creative work, was suggested by emulating a physical design-sketch in computer environment. In order to identify the process of designers' sketch, we performed several information gathering methods including the simple observation of numerous designers' sketch works, disposable-camera techniques, and in-depth interviews. Based on the factor analysis of a design sketch, design-sketch elements and design-sketch behaviors are extracted, and mapped to digital functionalities for developing vector-type graphics S/Ws. The proposed method was implemented as a simple Java Applet program, and set a high value by a professional industrial-designer group.

By digitalizing a design sketch, while the merits of a physical sketch are wholly inherited, the demerits of physical sketch (for example, storage, retrieval, edit, etc.) can be complemented. This new digital media can easily be for the designer's actual work without serious resistance. Because our digital-styling tool was designed on vector-type data treatment scheme, it can resolve the many problems raster-type graphics S/Ws have. Most of all, it can provide a digital connectivity with the downstream process, and allow concurrent engineering throughout the whole product-development process.

The proposed digital-sketch concept has a lot of possibilities to be extended as follows: 1) a new form of regular vector-type 2D graphics S/W by including advanced functions such as editing, filleting, concatenating, neutral file in/out, etc., 2) a plug-in form of existing vector-type graphics or CAD S/Ws as a 2D input device, 3) an alternative user-interface method for most sketch-based applications, 4) a direct 3D freeform-shape creation S/W for design professionals [2].

Acknowledgements

We sincerely thank a lot of designers and the graduate students of Department of Industrial Design, KAIST, who tested our program and gave valuable comments. Especially, we express our gratitude to the designers of Team Design, Design Dream, who permitted us to videotape their sketch work.

References

1. Arvo, J., Novins, K.: Fluid Sketches: Continuous Recognition and Morphing of Simple Hand-Drawn Shapes. The Proceedings of UIST (2000)
2. Bae, S.-H., Kijima, R., Kim, W.-S.: Digital Styling for Designers: 3D Plane-Symmetric Freeform Curve Creation Using Sketch Interface. The Proceedings of CGGM (2003)
3. Baudel, T.: A Mark-Based Interaction Paradigm for Free-Hand Drawing. The Proceedings of UIST (1994)
4. http://www.britanica.com
5. Choi, B.K.: Surface Modeling for CAD/CAM. Elsevier, NY (1991)
6. Doblin, J.: Perspective: A New System for Designers. Whitney Publications, NY (1956)
7. Eberly, D.H.: 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics. Morgan Kaufmann, San Francisco (2001)
8. Eggli, L., Hsu, C.-Y., Bruderlin, B.D., Elber, G.: Inferring 3D Models from Freehand Sketches and Constraints. CAD, 29(2) (1997) 101-122
9. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide 5th Edition. Academic Press, NY (2002)
10. Faugeras, O.D., Herbert, M.: The Representation, Recognition and Locating of 3-D Objects. Int. J. of Robotics Research 5(3) (1986) 27-52
11. Herbert, D.M.: Architectural Study Drawings. John Wiley & Sons, NY (1993)
12. Hwang, T., Ullman, D.: The Design Capture System: Capturing Back-of-the Envelope Sketches. J. for Eng. Design 1(4) (1990) 339-353
13. Igarashi, T., Matsuoka, S., Kawachiya, S., Tanaka, H.: Interactive Beautification: A Technique for Rapid Geometric Design. The Proceedings of UIST (1997)
14. http://math.nist.gov/javanumerics/jama/
15. Koffka, K.: Principles of Gestalt Psychology. Harcourt Brace, NY (1935)
16. Kohler, W.: Gestalt Psychology: An Introduction to New Concepts in Modern Psychology. Liverright Publishing Company, NY (1947)
17. Murray, P., Murray, L.: Penguin Dictionary of Art & Artists. Penguin Books, London (1976)
18. Piegl, L., Tiller, W.: The NURBS Book. Springer-Verlag, NY (1995)
19. Pigounakis, K.G., Kaklis, P.D.: Convexity-Preserving Fairing. CAD 28(12) (1996) 981-994
20. Pugh, D.: Designing Solid Objects Using Interactive Sketch Interpretation. The Proceedings of Symposium on Interactive 3D Graphics (1992)
21. Qin S., Wright D., Jordanov I.: From On-line Sketching to 2D and 3D Geometry: A System Based on Fuzzy Knowledge. CAD 32(14) (2000) 851-866
22. Roth-Koch, S.: Generating CAD Model from Sketches. The Proceedings of Workshop on Geometric Modelling (2000)

23. Schweikardt, E., Gross, M.D.: Digital clay: Deriving Digital Models from Freehand Sketches. The Proceedings of CHI (2002)
24. Tolba, O., Dorsey, J., McMillan, L.: A Projective Drawing System. The Proceedings of Symposium on Interactive 3D Graphics (2001)
25. Zeleznik, R.C., Herdon, K.P., Hughes, J.F.: SKETCH: An Interface for Sketching 3D Scenes. The Proceedings of SIGGRAPH (1996)
26. Zwillinger, D., Kellogg, B.: CRC Standard Mathematical Tables and Formulae 30th Edition. CRC Press, Florida (1996)

Digital Styling for Designers: 3D Plane-Symmetric Freeform Curve Creation Using Sketch Interface

Seok-Hyung Bae[1,2], Ryugo Kijima[2], and Won-Sup Kim[3]

[1] Virtual Reality Research Center, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
bae@vr.kaist.ac.kr
http://vr.kaist.ac.kr/∼bae
[2] Virtual System Laboratory, Gifu University, 1-1 Yanagido, Gifu, 501-1193, Japan
{bae,kijima}@vsl.gifu-u.ac.jp
[3] Department of Industrial Design, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
iron@mail.kaist.ac.kr

Abstract. The 3D evaluation of design shapes is an essential step in product styling. Thus, physical models of final-stage design alternatives have been made in tradition, and recently the effort to substitute them with CAD models has been tried. Whereas, designers in the early phase of the design-development stage where almost design concepts are determined, still use raster-type 2D graphics S/Ws. It causes not only the difficulty of evaluating 3D shapes but also the serious severance of a digital dis-connectivity with downstream processes. This paper presents a method of directly constructing 3D plane-symmetric freeform curves with a sketch interface, as the first step of developing a sketch-based 3D-freeform shape creation S/W for designers. A curve drawn by the designer within the see-through box in an arbitrary perspective view, is simultaneously converted to a real space curve without the 3D ambiguity problem except only special cases to be specified.

1 Introduction

As the customer's respect to products is being transformed from functionality to design, the importance of *product styling* is highly increasing in the whole product-development process. The more aesthetic the style of products is, the more important to evaluate 3D shapes of them before actual production become. For the reason, making physical models made of clay, wood, resin, form urethane, etc., is common in design studios [16]. However, physical models can not always be used because they require a lot of money and time. Thus, recently, the effort to substitute physical models with computer models using CAD S/Ws, has been explored.

So far, this kind of endeavor has focused on only final (or semi-final) stage design alternatives, and designers in the early phase of the design-development

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 701–710, 2003.
© Springer-Verlag Berlin Heidelberg 2003

stage where most of design concept is fixed still uses *raster-type* 2D graphics S/Ws. Although this kind of S/Ws enables designers 1) to *rapidly* generate various ideas, 2) to *flexibly* hybridize existing styles, and 3) to *sensuously* express virtual images in their minds, they have serious problems as follows: 1) the difficulty of *3D evaluation*, 2) *no digital connectivity* with downstream processes.

In this paper, as our first attempt to develop a *direct* 3D freeform-shape creation S/W allows the both of 3D evaluation and digital continuity, a method of constructing *3D plane-symmetric freeform curves* with a *sketch interface* is given. The designer-drawn freeform curves symmetric to the center plane–many products are plane symmetric including automobiles except little *avant-garde* styles–are simultaneously converted to 3D space curves without causing the 3D ambiguity in virtue of its *strong* constraint–the *plane symmetry* (a little special viewing-situations in which our method does not work will be specified in the context).

The method to be proposed has a *great* assumption that *well-trained* designers can *exactly* express 3D shapes on a 2D plane based on *design perspective*. Actually, professional industrial-designers practice and practice so that they can sketch the *accurate* plane images of virtual 3D objects because sketch is not only a means of presentation but also that of idea development [6]. We valued the designer's works based on design perspective as resultants from a *pretty* accurate *human* graphics-rendering pipe line.

The organization of this paper is as follows: related work to the direct creation of computer models including space curves will be given in Section 2. In Section 3, so-called a *perspective 3D sketch* scheme extended from the 2D sketch interface proposed in our companion paper [1] is suggested. In Section 4, the overview of a method converting 2D curves drawn in a perspective view to 3D curves is given, and then the actual calculation procedure of finding 3D points from 2D points is presented with special case treatments in Section 5. The software implementation is given in Section 6 followed by discussions and conclusions in Section 7.

2 Related Work

We categorized research on *direct* 3D computer-model creations as following three based on the user interface: 1) *sketch-based* methods, 2) *suggestive* methods, and 3) *VR-interface* methods. Sketch-based methods have been mostly applied to creating CSG-like models composed of simple primitives [7][15][17] where the most interest was focused on the *primitive recognition* from a rough scribble, and the *topology reconstruction*. There were little studies about directly creating freeform-shapes: one is Teddy [11] for *rounded* freeform models, which, however, is not appropriate for product styling. A suggestive interface is to forecast possible subsequent operations to be executed by users, and to gives action alternatives [4][12]. It shows great possibility for creating simple polygon models, but is limited for freeform-shapes. Recently, much research with 3D interfaces (VR interfaces) has been done. However, VR techniques for the direct creation

of freeform shapes looks still immature in a practical point of view (heavy equipment, precision problem, etc.).

There are two representative studies about 3D space-curve creation: Cohen, et al. [3] suggested a method by sketching a curve and its shadow curve on the floor in a perspective view. Grossman, et al. [10] proposed the reverse way where a depth plane (surface) is constructed first, and the curve drawn on orthographic plane is projected on it.

3 Perspective 3D Sketch

The sketch interface proposed in this paper, *perspective 3D sketch*, is straightforwardly extended from the intuitive 2D sketch interface proposed in our companion paper [1], which enables the designer to freely create accurate curves intended by allowing repetitive scribbling (scribbles already drawn are spread and grayed out as the number of them increases) and inducing the designer's adaptation. Shown in Fig. 1 is the flowchart of the perspective 3D sketch scheme.

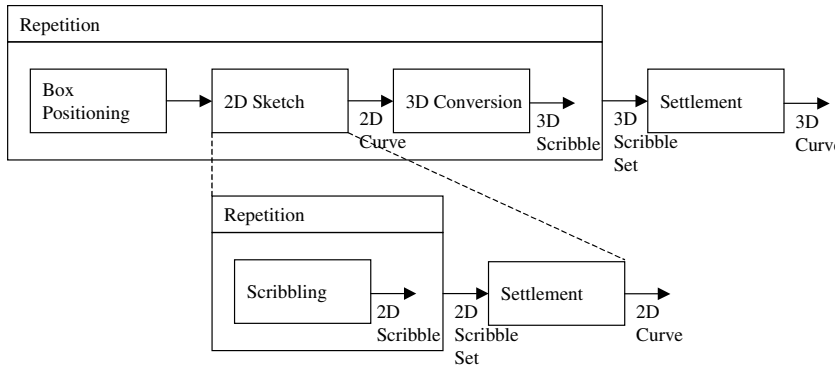


Fig. 1. Flowchart of perspective 3D sketch

The designer can choose an arbitrary perspective view by rotating the *see-through* box provided as a *reference unit* for estimating the dimension of 3D curves to be projected with it [6]. Then he/she draws a curve, $\tilde{\mathbf{c}}(t)$, on 2D image plane (Fig. 2(a)). By repetition of the previous procedure, a set of 3D curves, $\{\mathbf{c}_i(t)\}$, based on the designer's adaptation are generated (Fig. 2(b)), and at last, a final curve, $\bar{\mathbf{c}}(t)$, is settled (Fig. 2(c)).

4 Conversion of 2D Curve to 3D

The *pinhole-camera model* widely used in computer graphics and machine vision is composed of the *optical center*, $\mathbf{e}$, and the *retinal* (or image) *plane*, Π_R [9].

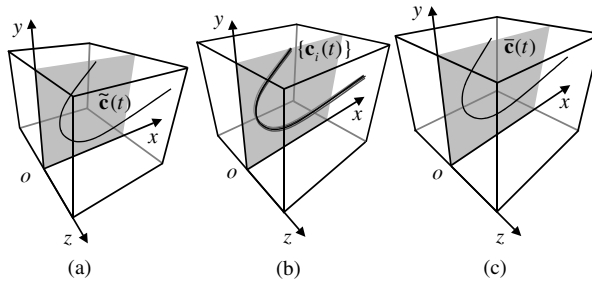


Fig. 2. Illustrative example of perspective 3D sketch

Between an arbitrary 3D point, $\mathbf{p}$, and its projection point, $\tilde{\mathbf{p}}$, there is a relation–a kind of *morphism*–as written as:

$$\tilde{\mathbf{p}}^w = \mathbf{H}\mathbf{p}^w \tag{1}$$

where $\mathbf{H}$ is the *perspective projection matrix* ($rank(\mathbf{H}) = 3$), $\mathbf{p}^w$ and $\tilde{\mathbf{p}}^w$are the *homogeneous coordinates* of $\mathbf{p}$ and $\tilde{\mathbf{p}}$, respectively.

In general, the inverse problem of perspective projection–finding $\mathbf{p}$ from $\tilde{\mathbf{p}}$–is an *under determined* problem, which has an infinite numbers of solutions. In this study, the above *3D ambiguity* is resolved by imposing the *plane-symmetry* condition, and a unique solution can be calculated. Fig. 3 shows our system configuration to be considered where $\mathbf{q}$ is the intersection point of the see-through box and the optical ray passing through $\tilde{\mathbf{p}}$ and $\mathbf{p}$.

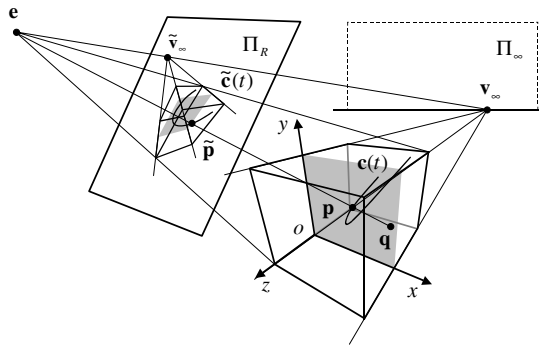


Fig. 3. System configuration for inverse perspective projection

The overall procedure of converting a 2D curve drawn by the designer to a 3D plane-symmetric freeform curve is as follows: 1) polygonizing a 2D parametric-curve, 2) matching a 2D-point pair, 3) finding a 3D-point pair on see-through

box, 4) finding a 3D-point pair looked for, and 5) creating a resulting 3D parametric-curve.

4.1 Parametric-Curve Polygonization

Given a parametric 2D curve, $\tilde{\mathbf{c}}(t)$, is sampled, and 2D point set, $\{\tilde{\mathbf{p}}_i(\tilde{x}_i, \tilde{y}_i)\}$, is obtained by the parametric-curve polygonization algorithm [5][13].

4.2 Matching 2D-point Pair

Using the 3-point perspective [6], generally used by industrial designers, a 2D-point pair expected to symmetric to the center plane in 3D space, can be acquired. First, the vanishing point, $\tilde{\mathbf{v}}_\infty$, is simply found by extending the edges of the see-through box parallel to z-axis, and given point, $\tilde{\mathbf{p}}_1 \equiv \tilde{\mathbf{p}}_i$, is mapped to its corresponding point, $\tilde{\mathbf{p}}_2$, by calculating the intersection point between a line passing through $\tilde{\mathbf{p}}_1$ and $\tilde{\mathbf{v}}_\infty$, and $\tilde{\mathbf{c}}(t)$ (or its linear interpolant obtained by polygonization) (see Fig. 4).

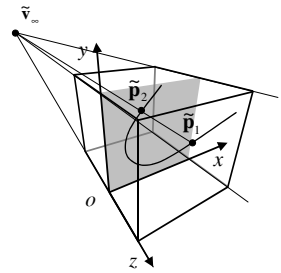


Fig. 4. 2D-point pair matching using vanishing point

4.3 Finding 3D-point Pair on See-Through Box

Instead of directly finding 3D-space points, $(\mathbf{p}_1, \mathbf{p}_2)$–as mentioned earlier, it is impossible to calculate them without the 3D ambiguity, their projection points on see-through box, $(\mathbf{q}_1, \mathbf{q}_2)$ are calculated from 2D points, $(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2)$, as follows (see Fig. 5(a) and refer our viewing system configuration given in Fig. 3) : For each 2D point, 1) finding *two* faces its optical ray pass through, 2) choosing the face between them, on which its 2D image point is closer to the center for more accurate calculation, 3) calculating two parameters of an *affine combination* in the 2D space, (μ, η), using two corresponding vanishing points, and 4) obtaining the 3D projection point on the see-through box using affine combination in the 3D space with the parameters previously calculated. For example, as shown

in Fig. 5(b), between the two projection faces of $\tilde{\mathbf{p}}_1$–$(F_{xy}^{front}, F_{yz}^{rear})$, F_{xy}^{front} is selected, and (μ, η) are calculated using two vanishing points, $(\tilde{\mathbf{v}}_\infty^{'}, \tilde{\mathbf{v}}_\infty^{''})$, so that

$$\tilde{\mathbf{p}}_1 = (1-\mu)(1-\eta)\tilde{\mathbf{v}}_1 + \mu(1-\eta)\tilde{\mathbf{v}}_2 + (1-\mu)\eta\tilde{\mathbf{v}}_3 + \mu\eta\tilde{\mathbf{v}}_4 \tag{2}$$

where $\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \tilde{\mathbf{v}}_4\}$ are the 2D image vertices of F_{xy}^{front}. Then, the 3D projection point on F_{xy}^{front}, $\mathbf{q}_1$, is acquired as:

$$\mathbf{q}_1 = (1-\mu)(1-\eta)\mathbf{v}_1 + \mu(1-\eta)\mathbf{v}_2 + (1-\mu)\eta\mathbf{v}_3 + \mu\eta\mathbf{v}_4 \tag{3}$$

where $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ are the 3D vertices of F_{xy}^{front}.

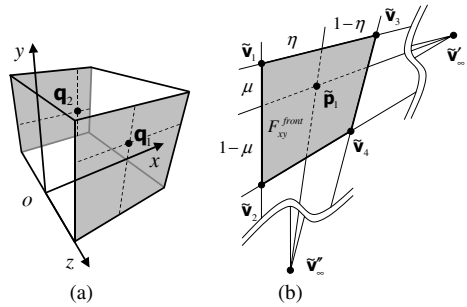


Fig. 5. Finding 3D-point pair on see-through box using vanishing points

4.4 Finding 3D-point Pair in Space

Now, we can define two optical rays, $(\mathbf{l}_1(t), \mathbf{l}_2(s))$, from the optical center, $\mathbf{e}$, using $(\mathbf{q}_1, \mathbf{q}_2)$ as follows:

$$\mathbf{l}_1(t) = \mathbf{e} + t(\mathbf{q}_1 - \mathbf{e}), \tag{4}$$

$$\mathbf{l}_2(s) = \mathbf{e} + s(\mathbf{q}_2 - \mathbf{e}). \tag{5}$$

The 3D points, $\mathbf{p}_1 = \mathbf{l}_1(t^*) = (p_{1x}, p_{1y}, p_{1z})$ and $\mathbf{p}_2 = \mathbf{l}_2(s^*) = (p_{2x}, p_{2y}, p_{2z})$, which we want to find, must satisfy the center-plane symmetry condition. That is, $p_{1x} = p_{2x}$ & $p_{1y} = p_{2y}$ & $p_{1z} = -p_{2z}$. Thus, a solution-parameter pair, (t^*, s^*), can be obtained by solving a system of three linear equations as follows (for more details, see Section 5):

$$t(q_{1x} - e_x) = s(q_{2x} - e_x), \tag{6}$$

$$t(q_{1y} - e_y) = s(q_{2y} - e_y), \tag{7}$$

$$e_z + t(q_{1z} - e_z) = -e_z - s(q_{2z} - e_z). \tag{8}$$

4.5 3D Parametric-Curve Creation

As the final step of converting a 2D curve to a 3D space curve, the approximation of $\{\mathbf{p}_i(x_i, y_i, z_i)\}$ to a parametric curve such as B-spline (or Bezier) curve, $\mathbf{c}(t)$, is performed by applying standard curve fitting methods [2][8][14] keeping a plane symmetry.

5 Solving System of Linear Equations for Finding 3D-point Pair

As given in (6)˜(8), our inverse projection of a 2D-point pair to 3D points under the plane-symmetry condition is an *over-determined* problem having *two* unknowns and *three* equations. Because of the pinhole-camera model assumption $(t, s > 0)$, the solution space is $(t^*, s^*) \in (0, \infty) \times (0, \infty)$. For most cases, (t^*, s^*) can be simply determined by minimizing the sum of least-squares errors, $f = (\mathbf{NF} - \mathbf{G})^T(\mathbf{NF} - \mathbf{G})$, where $\mathbf{NF} = \mathbf{G}$ is the matrix form of (6)˜(8). However, there exist special viewing conditions in which the proposed method does not work. Thus, the analysis of these special cases will be followed with the consideration of the number of equations.

First of all, let us consider the cases that one of three equations vanishes as shown in Fig. 6. The case that only (6) vanishes, that is, $q_{1x} - e_x \approx 0$ & $q_{2x} - e_x \approx 0$, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are all on the same plane parallel to the yz-plane (see Fig. 6(a)). In the case, it is possible to calculate (t^*, s^*) with remaining two equations. Similarly, if (7) vanishes, the solution exists (Fig. 6(b)). However, when (8) vanishes, that is, $q_{1z} - e_z \approx 0$ & $q_{2z} - e_z \approx 0$ & $e_z \approx 0$, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are all on the xy-plane because $p_{1x} \approx p_{2x}$ & $p_{1y} \approx p_{2y}$ & $p_{1z} \approx -p_{2z}$ or $\mathbf{p}_1 \approx \mathbf{p}_2$ & $\mathbf{q}_1 \approx \mathbf{q}_2$. Thus, there are an infinite numbers of solutions (Fig. 6(c)) (it is so called the *impossible-to-solve* case).

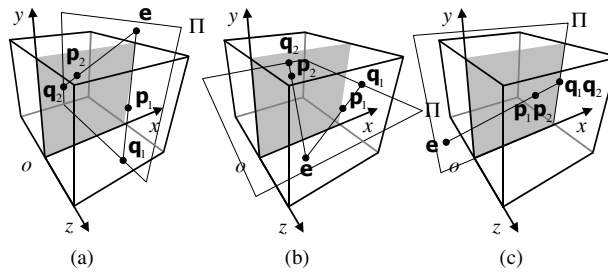


Fig. 6. Special cases only one equation vanishes

Now, let us consider the only one equation remains. They are all the impossible-to-solve cases (see Fig. 7). If (6) and (7) vanish, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$

are all on the line parallel to the z-axis as shown in Fig. 7(a). If only (6) or (7) remains, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are all on the same line parallel to the x-axis, or y-axis (on the xy-plane or center plane), respectively.

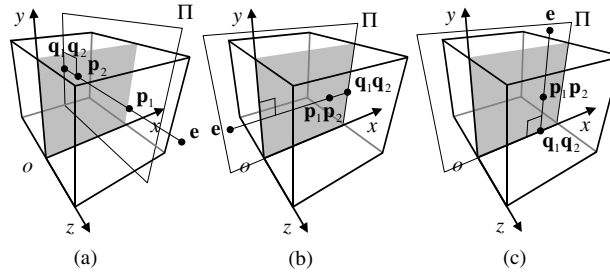


Fig. 7. Special cases only one equation remains

If all three of (6)˜(8) vanish, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are the coincident point (the *trivial-solution* case).

In summary, there are *seven* special viewing-cases: *one* trivial-solution case, *four* impossible-to-solve cases, and *two* possible-to-solve cases. In fact, four impossible-to-solve cases can be re-categorized as following two: 1) the optical center is positioned on the center plane, and 2) the optical rays are parallel to the z-axis.

For above possible-to-solve cases, (t^*, s^*) can be explicitly calculated. For the case only (7) vanishes ($q_{1x} - e_x \neq 0$ & $q_{2x} - e_x \neq 0$ & $q_{1y} - e_y \approx 0$ & $q_{2y} - e_y \approx 0$), by substituting $t = (q_{2x} - e_x)/(q_{1x} - e_x)$ into (8), we can obtain the following equation, $s\{(q_{1z} - e_z)(q_{2x} - e_x) + (q_{1x} - e_x)(q_{2z} - e_z)\} = -2(q_{1x} - e_x)e_z$, and then (t^*, s^*) except the case that the optical center is located at the infinite point where $(q_{1z} - e_z)(q_{2x} - e_x) + (q_{1x} - e_x)(q_{2z} - e_z) \approx 0$ (see Appendix A).

6 Implementation

A simple program for the proposed method is implemented as a *Java Applet* using Java$^{\text{TM}}$ 2 Platform Standard Edition (J2SE$^{\text{TM}}$) and Java 3D$^{\text{TM}}$ API as shown in Fig. 8. For a pen-based user interface, WACOM Intuos$^{\text{TM}}$2 tablet ($9'' \times 12''$) is used. Our Java Applet is available at http://vr.kaist.ac.kr/˜bae.

7 Discussions and Conclusions

In this paper, we proposed a method of constructing 3D plane-symmetric curves using an intuitive sketch interface, which was inspired from a fact that a lot of industrial products have plane-symmetric forms. The designer can arbitrary choose

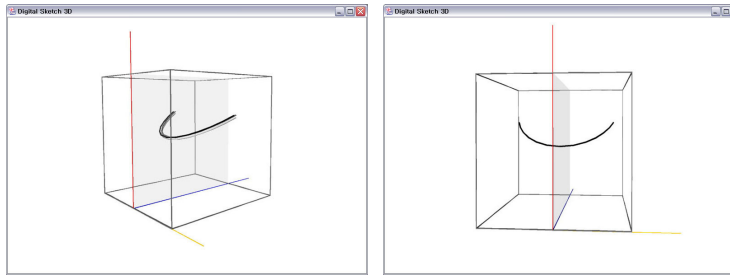


Fig. 8. Software implementation: repetitive 3D scribbling and settled freeform curve

a 3D perspective view, and draw a projected form of the curve intended. By repetition of the perspective 3D sketch (visual feedback and behavioral adaptation process), the designer can obtain the final form of a 3D space curve. Because of the assumption of a plane-symmetry, the proposed method does not suffer the 3D ambiguity problem unlike general inverse-projection of converting 2D image objects to 3D space objects (the exceptional viewing cases–only two–were addressed in Section 5). A prototype for the proposed method was made as simple Java Applet program, and tested by a group of product designers.

One important point is that the proposed method is relies on the designer's accurate space sense. The assumption can be thought as reasonable remembering that projective geometry, which computer graphics is based on, was started from Renaissance painters' effort to correctly reproduce the perspective effects in images of the world that they were observing. Actually, many designers took a great interest in our program, and agreed it has much possibility to be developed as a powerful tool for design specialists.

References

1. Bae, S.-H., Kim, W.-S., Kwon, E.-S.: Digital Styling for Designers: Sketch Emulation in Computer Environment. The Proceedings of CGGM (2003)
2. Choi, B.K.: Surface Modeling for CAD/CAM. Elsevier, NY (1991)
3. Cohen, J., Markosian, L., Zeleznik, R., Hughes, J., Barzel, R.: An Interface for Sketching 3D Curves. The Proceedings of SI3DG (1999)
4. Cypher, A.: Eager: Programming Repetitive Tasks by Example. The Proceeding of CHI (1991)
5. de Figueiredo, L.H.: Adaptive Sampling of Parametric Curves. In: Paeth, A.W. (eds): Graphics Gems V. Academic Press, Boston (1995)
6. Doblin, J.: Perspective: A New System for Designers. Whitney Publications, NY (1956)
7. Eggli, L., Hsu, C.-Y., Bruderlin, B.D., Elber, G.: Inferring 3D Models from Freehand Sketches and Constraints. CAD 29(2) (1997) 101-122
8. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide 5th Edition. Academic Press, NY (2002)

9. Faugeras, O., Luong, Q.-T.: The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications. The MIT Press, Cambrigde (2001)
10. Grossman, T., Balakrishnam, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., Buxton, B.: Creating Principal 3D Curves with Digital Tape Drawing. In Proceedings of CHI (2002)
11. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: Sketch Interface for a 3D Freeform Design. The Proceedings of SIGGRAPH (1999)
12. Igarashi, T., Hugh, J.F.: A Suggestive Interface for 3D Drawing. The Proceedings of UIST (2001)
13. Lane, J. M., Riesenfeld, R.F.: A Theoretical Development for Computer Generation and Display of Piecewise Polynomial Surfaces. IEEE Transactions on PAMI 2(1) (1980) 35-46
14. Piegl, L., Tiller, W.: The NURBS Book. Springer-Verlag, NY (1995)
15. Schweikardt, E., Gross, M. D.: Digital Clay: Deriving Digital Models from Freehand Sketches. The Proceedings of CHI (2002)
16. Yamada, Y.: Clay Modeling: Techniques for Giving Three-dimensional Form to Idea. Car Styling Extra Issues 93(1/2) (1993)
17. Zeleznik, R. C., Herdon, K. P., Hughes, J. F.: SKETCH: An Interface for Sketching 3D Scenes. The Proceedings of SIGGRAPH (1996)

Appendix A: Optical Center at Infinite Point

Let us consider the equality, $(q_{1z}-e_z)(q_{2x}-e_x)+(q_{1x}-e_x)(q_{2z}-e_z)\approx 0$, when only one equation, (7), vanishes as shown in Fig. 9(a). From the pinhole-camera assumption $(0<t<\infty)$, $(q_{1x}-e_x)(q_{2x}-e_x)>0$. Thus, $(q_{1z}-e_z)(q_{2z}-e_z)<0$, and there are two cases satisfying it: $q_{2z}<e_z<q_{1z}$ *or* $q_{1z}<e_z<q_{2z}$. Without the loss of generality, we can choose the case of $q_{2z}<e_z<q_{1z}$ (see Fig. 9(b)). The modified form of the given equality can be written as $(q_{1z}-e_z)/(q_{1x}-e_z)\approx(e_z-q_{2z})/(q_{2x}-e_x)$, and it means two triangles, $(\triangle\mathbf{e}^*\mathbf{q}_1^*\mathbf{r}_1^*, \triangle\mathbf{e}^*\mathbf{q}_2^*\mathbf{r}_2^*)$, must be have almost *foldaway* forms. Because the case now treated is one of the possible-to-solve cases, $\mathbf{e}$ can not be positioned on the center plane or $e_z\neq 0$. Therefore, $e_z\rightarrow\pm\infty$ or $(t^*,s^*)=(\infty,\infty)$.

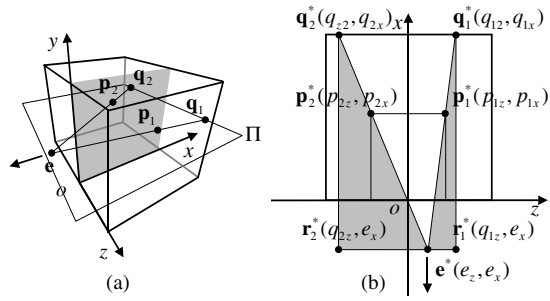


Fig. 9. Optical center at infinite point

Directional Offset of a Spatial Curve for Practical Engineering Design

Hayong Shin[1], Seock K. Yoo[1], Su K. Cho[2], Won H. Chung[1]

[1] KAIST, Dept. of Industrial Engineering, 305-701, Daejeon, S.Korea,
{hyshin, jcircle, skyoo}@vmslab.kaist.ac.kr
[2] DaimlerChrysler Corp., Technical Computing Center, Auburn Hills, MI, USA
sc211@dcx.com

Abstract. In this paper, we propose a 3D curve offset method, named *directional offset*, motivated from the observation of the needs in many engineering design practices such as flange of sheet metal parts. Since the normal vector of a 3D curve at a point is not unique, a 3D curve offset definition is about selecting the offset direction vector on the normal plane of the curve. In directional offset, the offset direction vector is chosen to be perpendicular to the user-specified projection direction vector as well as the curve tangent vector. Directional offset is a natural extension of planar curve offset, in the sense that they produce the same results when applied to planar curve. An overall procedure to compute a directional offset for a position-continuous NURBS curve is described with an emphasis on avoiding self-intersection loop.

1. Introduction

An offset of a curve is defined as a set of points having a constant distance from the original curve along the offset direction. The offset curve $\mathbf{r}^o(u)$ of a parametric curve $\mathbf{r}(u)$ can be simply described as :

$$\mathbf{r}^o(u) = \mathbf{r}(u) + d\,\mathbf{n}(u) \qquad (1)$$

where $\mathbf{n}(u)$ is the offset direction and d is the offset distance.

Being an important topic, there have been a large number of literatures on curve and surface offsetting, as surveyed in [7, 11]. In 2D curve case, the offset direction $\mathbf{n}(u)$ is typically defined as the normal vector of $\mathbf{r}(u)$. However, in 3D curve case, there is no commonly agreed definition of $\mathbf{n}(u)$. Most of the researches on curve offsetting are devoted to 2D curve offsetting, and literatures on spatial curve offsetting are surprisingly rare. Since the normal vector of a spatial curve at a point is not unique, a curve offset definition is to select the offset direction vector on the normal plane $\pi(u)$ of the curve. Obviously, the principal normal vector $\mathbf{n}_p(u)$ is not a good choice for the offset direction, since the resulting curve is not very useful in engineering design practices as shown in Fig.1 and Fig.2. (For the definition of principal normal, the readers are referred to differential geometry textbooks such as [6].) Coquillart [1] proposed to use a *quasi-normal* $\mathbf{n}_q(u)$ for the offset direction.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 711-720, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Basically quasi-normal maintains a constant angle θ with the principal normal $\mathbf{n}_p(u)$ on $\pi(u)$ as shown in Fig.1 and Fig.3, with exceptions at inflection points and line segments. The idea of quasi-normal seems to be more thoughtful selection of offset direction. Nevertheless, offsetting with quasi-normal direction is not quite suitable in many practical engineering design practices such as illustrated in Fig.15 and Fig.16. From the observation of such design examples, we obtained the following required properties of spatial curve offset :

- R1: When viewed from the user-specified direction, the offset curve should be the same as (or close to) the 2D offset, while inheriting the height value in the given direction of the original curve.
- R2 : The resulting curve should be self-intersection free.
- R3 : It should be applicable to G^0-continuous parametric curves, which are very common in engineering design.
- R4 : Let **S** be the ruled surface between the input curve and the offset curve where the ruling line is the offset direction. It is desirable that **S** be G^1-continuous as much as possible. However, this requirement is optional depending on the user's choice.

Presented in this paper is a 3D curve offset method, called *directional offset*, which satisfies the above-mentioned requirements. The idea of directional offset was presented by the same authors in Solid Modeling & Applications conference [15]. This paper is a shortened, compact, and rearranged version of the original paper.

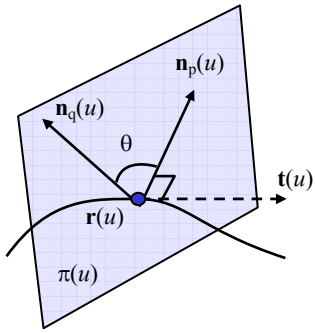


Fig. 1. Principal normal and quasi-normal.

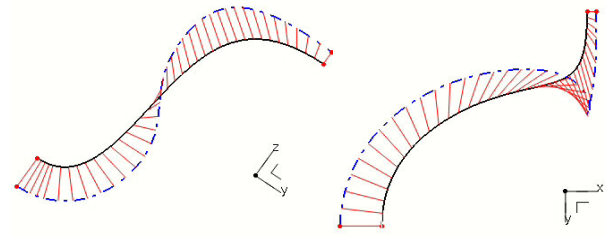


Fig. 2. Offset using the principal normal. (solid line : the original curve, dashed line : the offset curve)

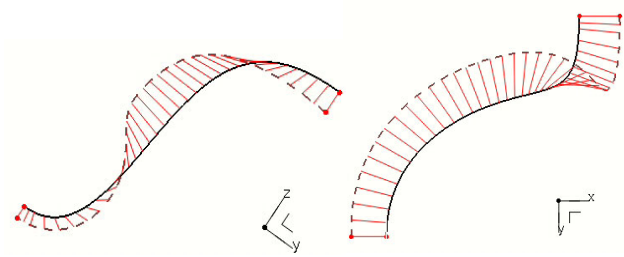


Fig. 3. Offset using the quasi-normal direction. (solid line : the original curve, dashed line : the offset curve)

2. Offset Direction Vector

The inputs of the directional offset are a parametric curve $\mathbf{r}(u)$, offset distance d, and a projection vector $\mathbf{k}$. The role of the additional parameter $\mathbf{k}$ is to help selecting the offset direction vector $\mathbf{n}(u)$, so as to meet the requirement R2 when viewed from $\mathbf{k}$. For the sake of simplicity, hereafter, we assume that $\mathbf{k}$ be the Z-axis direction without loss of generality, since $\mathbf{r}(u)$ can be rotated so that $\mathbf{k}$ coincides with the Z-axis. Let ${}^{xy}\mathbf{r}(u)$ denote the 2D curve obtained by projecting $\mathbf{r}(u)$ onto XY-plane in the direction of $\mathbf{k}$. We also assume that $\mathbf{r}(u)$ is represented in NURBS and that ${}^{xy}\mathbf{r}(u)$ does not have a self-intersection loop, for the simplicity again.

In directional offset, $\mathbf{n}(u)$ is defined as the unit vector on $\pi(u)$ orthogonal to both $\mathbf{k}$ and the unit tangent vector $\mathbf{t}(u)$ of $\mathbf{r}(u)$ as illustrated in Fig.4. For the most points on $\mathbf{r}(u)$, $\mathbf{n}(u)$ can be easily computed by

$$\mathbf{n}(u) = (\mathbf{t}(u) \times \mathbf{k}) / | \mathbf{t}(u) \times \mathbf{k} | \qquad (2)$$

When $(\mathbf{t}(u) \times \mathbf{k})$ vanishes momentarily, $(\mathbf{r}''(u) \times \mathbf{k})$ can be used instead, by applying L'Hopital's rule. If $\mathbf{r}(u)$ contains a vertical line segment for $u \in (a,b)$, this also fails and $\mathbf{n}(u)$ should be determined to be compatible with $\mathbf{n}(a)$ and $\mathbf{n}(b)$. This case is further classified into 3 sub-cases :

- If $\mathbf{n}(a) = \mathbf{n}(b)$, we can set $\mathbf{n}(u) = \mathbf{n}(a)$ for any $u \in (a,b)$
- If the vertical line $\{\mathbf{r}(u);\ u \in (a,b)\}$ forms a convex corner toward the offset direction when viewed from $\mathbf{k}$, a helix-like segment should be inserted for $u \in (a,b)$. Section 4 will describe the case in detail including the definition of 'convex (and concave) corner'.
- If the corner is a concave one, section 5 will handle it.

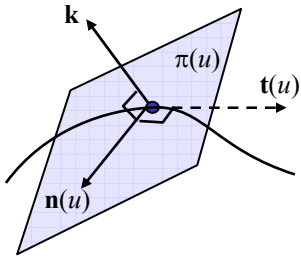


Fig. 4. Offset direction.

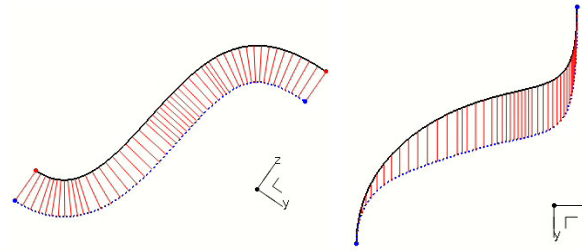


Fig. 5. Raw offset curve obtained by applying the directional offset to the curve in Fig. 2.

3. Overall Algorithm of Directional Offsetting

The overall algorithm to implement the directional offset is simply described below. Step 1 & 2 are briefly described in this section, while Step 3 and Step 4 are explained in the subsequent sections.

Step 1. //segmentation into G^1 continuous pieces

- Identify tangent discontinuous points of ${}^{xy}\mathbf{r}(u)$ and vertical line segments of $\mathbf{r}(u) \rightarrow \{ u_i : i = 0,..,n \}$ where u_0 and u_n are the start and the end parameter of $\mathbf{r}(u)$, respectively.
- Subdivide $\mathbf{r}(u)$ at $\{ u_i \}$ into $\{ \mathbf{r}_i(u) : i = 1..n \}$, where $\mathbf{r}_i(u)$ is defined for $u \in [u_{i-1}, u_i]$

Step 2. // raw offsetting for each segment

- Compute the directional offset curve $\mathbf{r}^o_i(u)$ for each $\mathbf{r}_i(u)$ using the offset direction vector described in the previous section.

Step 3. // convex corner handling (explained in section 4)

- If ${}^{xy}\mathbf{r}(u_i)$ forms a convex corner toward the offset direction, the gap-filling segment is inserted in order to connect $\mathbf{r}_i^o(u_i)$ and $\mathbf{r}^o_{i+1}(u_i)$ (or $\mathbf{r}^o_{i+2}(u_{i+1})$).

Step 4. // self-intersection handling (explained in section 5)

- Identify all self-intersections among $\{ {}^{xy}\mathbf{r}^o_i(u) \}$
- For each self-intersection, delete the invalid portion and the self-intersection handling procedure is invoked.

Step 1 is to subdivide the input curve $\mathbf{r}(u)$ into $\{ \mathbf{r}_i(u) \}$, to which the corresponding ${}^{xy}\mathbf{r}_i(u)$ is G^1-continuous for $u \in (u_{i-1}, u_i)$. Note that ${}^{xy}\mathbf{r}(u)$ can be easily obtained by just ignoring the Z-ordinate of $\mathbf{r}(u)$. The singular points can be found by examining the multiple knots or the overlapping control points. The singularity in tangent vector may also occur in the middle of any knot span, depending on the control point arrangement. For the details on this issue, the readers are referred to [5,12]. Nevertheless, a robust implementation of segmentation is not a trivial task.

Step 2 utilizes the offset direction definition of the previous section. There have been a number of researches on the offset curve approximation [1,3,4,10,16] as well as a comparison between the approximation methods summarized in [2]. A recent paper by Piegl and Tiller [13] provides a well-established scheme of offset curve approximation, which consists of the four major steps : (1) special shape identification and handling, (2) offset point sampling, (3) interpolation of offset points, and (4) approximation by knot removal. This scheme can be applied to the raw offset curve approximation, with the minor modification of using the offset direction described in the previous section when sampling offset points. Fig.5 shows a raw offset curve from step 2 using the offset direction described in Section 2.

4. Convex Corner Handling

$\mathbf{r}(u_i)$ is called a *simple convex corner* if ${}^{xy}\mathbf{t}(u_i\text{-}0) \times {}^{xy}\mathbf{t}(u_i\text{+}0)$ is non-zero and is in the same direction with $\mathbf{k}$ (ie $\mathbf{k} \bullet ({}^{xy}\mathbf{t}(u_i\text{-}0) \times {}^{xy}\mathbf{t}(u_i\text{+}0)) > 0$). If $\mathbf{r}_{i+1}(u_i)$ is a vertical line segment and ${}^{xy}\mathbf{t}(u_i\text{-}0) \times {}^{xy}\mathbf{t}(u_{i+1}\text{+}0)$ is non-zero and in the same direction with $\mathbf{k}$, $\mathbf{r}(u_i)$ is called a *vertical convex corner*. Otherwise, it is called a *concave corner*. The raw offset curves obtained in the previous step would have gaps in-between. For the concave corners, the gaps will disappear during the self-intersection handling step. However, the gaps around the convex corners should be filled with extra curve segments to form a continuous offset curve. A naive approach is to insert a circular arc for a simple convex corner or a helix segment for a vertical convex corner.

However, this approach results in a tangent discontinuous offset curve around the convex corner, since the tangent vectors of the neighbor segments are not taken into consideration in 3D sense when constructing the gap-filling segment.

Basically the convex corner gap-filling problem is to construct a curve segment on a cylinder (centered at the corner with the radius d and the axis vector $\mathbf{k}$) interpolating position and tangent direction at the end points. Since a simple convex corner can be handled as a special case of vertical convex corner case, we will explain the vertical convex corner case only. As depicted in Fig.6, the convex corner gap-filling problem is to find a curve segment interpolating two end points and the tangency condition at the end points, while staying on the cylinder. Since this is a Hermite interpolation on cylindrical surface, we can get a rational Bézier curve of degree 6, which interpolates the given end conditions and is exactly lying on the given cylinder. For the detailed solution procedure for this problem, the readers are referred to [15].

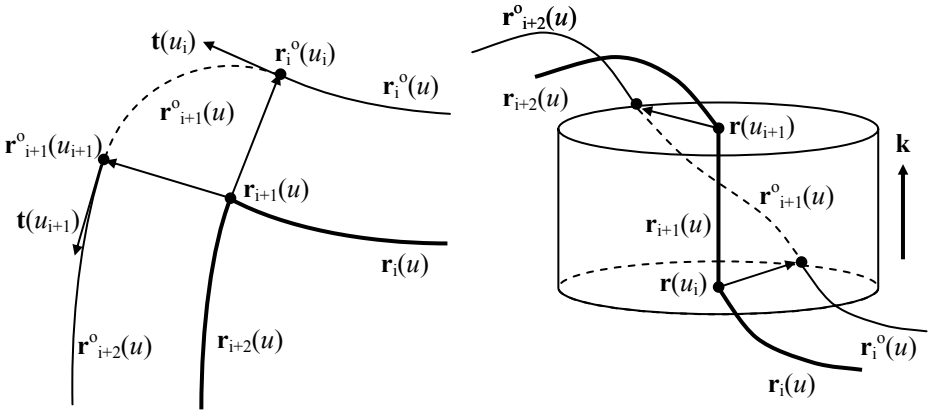


Fig. 6. Vertical convex corner handling.

5. Self-intersection Handling

Once raw offset curves are obtained and the gaps are filled, the next step is to find the self-intersections among $\{ {}^{xy}\mathbf{r}^o_i(u) \} \cup \{$ gap-filling segments $\}$. Though finding self-intersection is a non-trivial task, we would like to avoid going in deep into the issue, since it goes beyond the focus of this paper. Instead, the readers are referred to [8,9,17]. Briefly speaking, first, each offset curve is approximated to a polygon with vertices obtained in offset point sampling of Step 2. Then, the polygonal chain intersection algorithm explained in the above references, which is based on an efficient sweep line algorithm, is applied to find the self-intersections within and between the polygons. The intersection points computed from polygons are not necessarily on the offset curves. A Newton iteration method is applied to find more exact intersection point, using the polygon intersection points as the seeds. From the self-intersection points, the invalid portions of the offset curves can be easily detected

and removed. However, in general, there is a gap in **k** direction at a self-intersection point. In Fig.7b, **a** and **b** are the same self-intersection point in XY plane, but they represent a vertical line segment in 3D view (Fig.7c). We propose 3 different options to fill a vertical gap.

- Option SIH-1 : This option is to simply connect **a** and **b** with a vertical line segment, shown as the vertical broken line segment in Fig.7c. With this option, all vertical drop (or elevation) happens at a single point in XY plane. This option is simplest and easiest to implement. Obviously, the resulting curve is not G^1 continuous and neither is the ruled surface connecting the original curve and the offset curve. Hence this option may not be very useful in many engineering design cases.

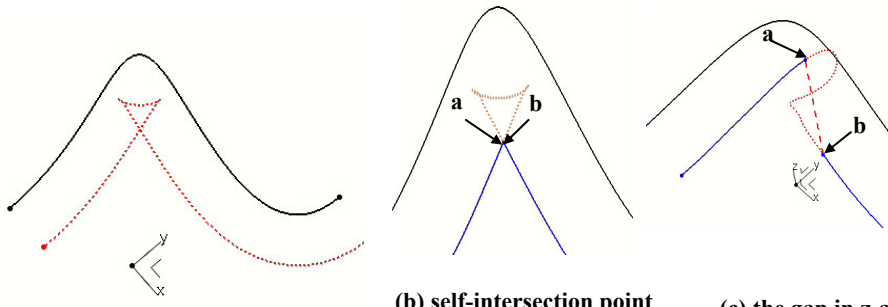
- Option SIH-2 : This option takes the *influence distance D* as an extra input parameter from the user. Let $\mathbf{r}^o_i(u)$ and $\mathbf{r}^o_j(u)$ be the two offset curves trimmed at the intersection point pair (**a, b**). And let *V* denote the vertical distance between **a** and **b**. The idea is to absorb the vertical drop (or elevation) *V* over the neighbor region with length *D* at the both side of the intersection point, as illustrated in Fig.8.
  - Trim $\mathbf{r}_i(u)$ at the point of arc length (or chord length) *D* measured from **a.** Let $\mathbf{p}(u)$ be the sub-segment of $\mathbf{r}_i(u)$ close to **a** as shown in Fig.8.
  - Let **S** be the surface obtained by extruding $\mathbf{p}(u)$ along **k** direction, and let **m** be the mid-point of **a** and **b**. Let $\mathbf{p}_0$ denote the start point of $\mathbf{p}(u)$ and $\mathbf{t}_0$ denote the start tangent vector at the start point of $\mathbf{p}(u)$.
  - Construct $\mathbf{q}(u)$ on S interpolating $\mathbf{p}_0$ and **m** with the tangent constraints $\mathbf{t}_0$ and **k** (or −**k**). Cylindrical Hermite interpolation will give us a rational Bezier curve segment of degree 6 for this problem [15].
  - Replace $\mathbf{p}(u)$ by $\mathbf{q}(u)$. Do the same to $\mathbf{r}_j(u)$.

- Option SIH-3 : This option is to modify the shape of the trimmed offset curves in XY view by inserting a circular arc around the intersection, as shown in Fig.9. This option also takes an extra input parameter : the filleting radius *R*. Steps to implement this option are :
  - Compute the fillet contact points **a**′ and **b**′ by finding the fillet circle with radius R tangent to both of $^{xy}\mathbf{r}_i(u)$ and $^{xy}\mathbf{r}_j(u)$. Trim $\mathbf{r}_i(u)$ and $\mathbf{r}_j(u)$ at **a**′ and **b**′, respectively.
  - Imagine a cylinder **S** by extruding the fillet circle along **k**. Construct the gap-closing segment on **S**, which interpolates **a**′ and **b**′ with the tangency condition to $\mathbf{r}_i(u)$ and $\mathbf{r}_j(u)$. Again, the procedure for cylindrical Hermite interpolation can be applied for this purpose [15].

  Strictly speaking, the resulting curve obtained by this option is not an offset as defined by eq. (1). Nevertheless, this option is very helpful in many engineering design cases which require smooth curves.

To help understanding the effect of options, Fig.10 ~ Fig.14 show the result of above-mentioned options. Though SIH-3 option generates most smooth offset curve,

the choice of self-intersection handling options depends on the specific need of the design case, so that it is desirable to leave the choice at user's hand.

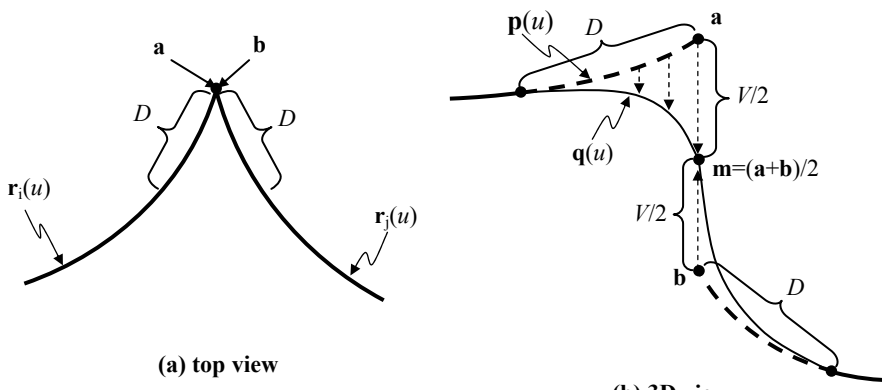


(a) raw offset curve

(b) self-intersection point and invalid portion (top view)

(c) the gap in z-axis direction (3D view)

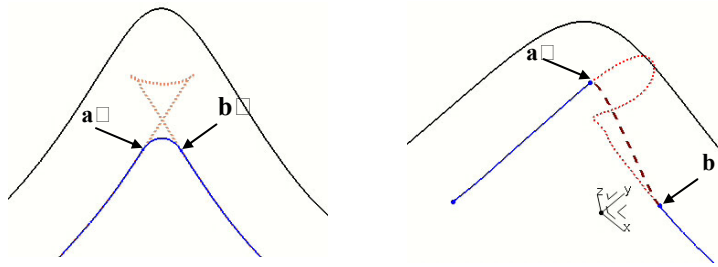
Fig. 7. A gap after removing the invalid portion.



(a) top view

(b) 3D view

Fig. 8. Self-intersection handling option SIH-2.



(a) top view

(b) 3D view

Fig. 9. Self-intersection handling option SIH-3.

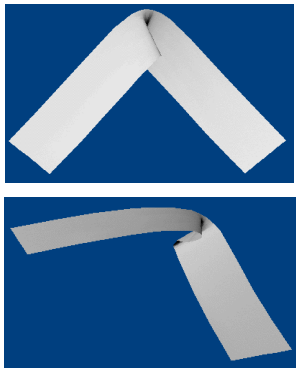


Fig. 10. Ruled surface connecting the input curve and raw offset curve.

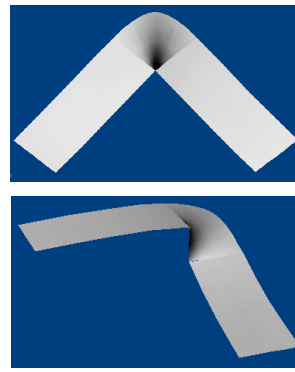


Fig.11. Result of SIH-1 option.

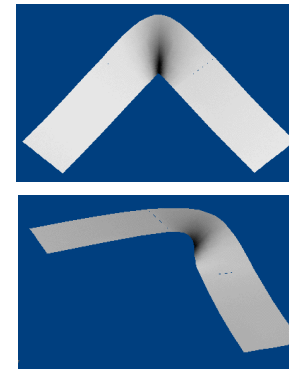


Fig.12. Result of SIH-2 option with *D=d.*

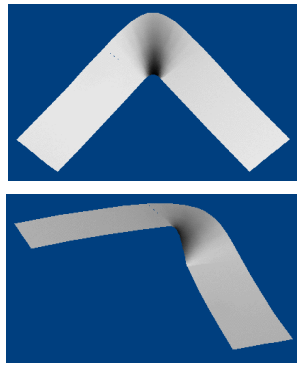


Fig.13. Result of SIH-3 option with *R=0.2d.*

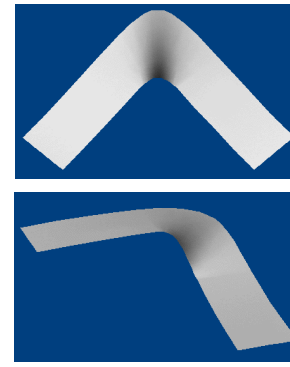


Fig.14. Result of SIH-3 option with *R=0.5d*

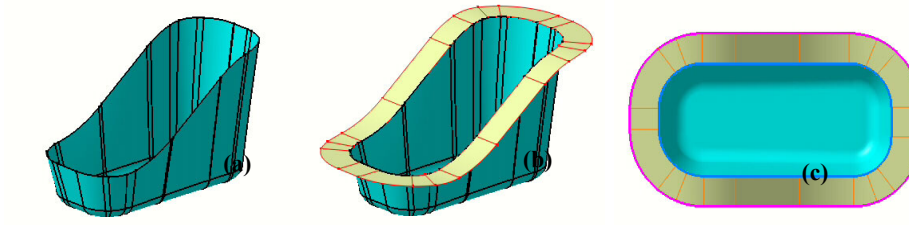


Fig. 15. A sheet metal part with a flange.

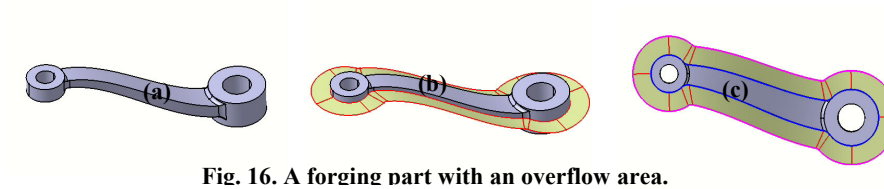


Fig. 16. A forging part with an overflow area.

6. Design Examples

In this section, we would like to show some practical engineering design examples modeled with directional offset. Fig 15 shows a sheet metal product with a flange. Note that the outer boundary curve of the flange is obtained by directional offsetting the inner boundary, and the flange itself is modeled as a ruled surface connecting the the inner boundary, and the flange itself is modeled as a ruled surface connecting the two. Shown in Fig 16 is an example design of a forging die for a simplified automotive part with a spatial parting curve. Fig.16a is the original part, while Fig.16b and Fig.16c are different views of the overflow area attached to the part. In a forging die, an overflow area around the parting line is required to capture the surplus material. Again, the outer boundary of the overflow area in Fig.16 is defined by directional offsetting the spatial parting curve, similar to the flange design case.

7. Summary and Discussion

The examples in Fig.15~Fig.16 are only a few engineering design cases, and we believe there are plenty of cases to which the directional offset can be applied. To the best of our knowledge, most of the major CAD systems do not provide such functionality at the point of writing this paper. In this paper, we proposed a 3D curve offsetting method, which has a wide variety of application and seems to be a natural extension of 2D curve offsetting. Detailed options were also discussed.

Determining the offset direction vector **k** is purely application specific. For example, in the forging die case in Fig.16, **k** is set to the die opening direction.

Directional offset can be explained in a different way. For a given 3D curve, consider a surface obtained by extruding the curve in the direction of **k**. It is obvious that the directional offset curve is identical to the boundary of the offset surface. Though this view does not provide an efficient way of obtaining the directional offset curve, it opens up a line of thinking in generalizing the concept of directional offset. For example, instead of an extrusion (cylindrical) surface passing through the given curve, a conical surface (with a given apex) can be considered. In this case, the direction vector **k** varies along the curve.

Acknowledgements

This research was partly supported by the Ministry of Science and Technology of Korean government.

References

[1] Coquillart S, "Computing offsets of B-spline curves", Computer-Aided Design, 1987, v19(6), pp.305-309.

[2] Elber G, Lee I-K, Kim M-S, "Comparing Offset Curve Approximation Methods", IEEE computer graphics and applications, 1997, v17(3), pp.62-71
[3] Hoschek J, Wissel N, "Optimal approximate conversion of spline curves and spline approximation of offset curves", Computer-Aided Design, 1988, v20(8), pp.475-483.
[4] Hoschek J, "Spline approximation of offset curves", Computer Aided Geometric Design, 1988, v5(1), pp.33-40
[5] Kim D-S, "Hodograph Approach to a Geometric Characterization of Parametric Cubic Curves", Computer Aided Design, 1993, v25(10), pp.644-654
[6] Lipschultz M, Differential Geometry, 1969, McGraw Hill
[7] Maekawa T, "An overview of offset curves and surfaces", Computer-Aided Design, 1999, v31, pp.165-173
[8] Park SC, Choi BK, "Uncut free pocketing tool-paths generation using pair-wise offset algorithm", Computer-Aided Design, 2001, v33(10), pp.739-746
[9] Park SC, Shin H, "Polygonal Chain Intersection", Computers & Graphics, in press.
[10] Pham B, "Offset approximation of uniform B-splines", Computer-Aided Design, 1988, v20(8), pp.471-474
[11] Pham B, "Offset curves and surfaces: a brief survey", Computer-Aided Design, 1992, v24(4), pp.223-229
[12] Piegl L, Tiller W, The NURBS book, 2nd ed, New York: Springer, 1997.
[13] Peigl L, Tiller W, "Computing offsets of NURBS curves and surfaces", Computer-Aided Design, 1999, v31(2), pp.147-156
[14] Peternell M, Pottmann H, "A Laguerre geometric approach to rational offsets", Computer Aided Geometric Design, 1998, v15, pp.223-249.
[15] Shin H, Cho S, "Directional offset of a 3D curve", Proceedings of 7th ACM Symposium on Solid Modeling and Applications (SM02), 2002, pp.329-335.
[16] Tiller W, Hanson EG "Offsets of two-dimensional profiles", IEEE Computer Graphics and Applications, 1984, v.4 (9), pp.61-69
[17] Wallner J, Sakkalis T, Maekawa T, Pottmann H, Yu G, "Self intersections of offset curves and surfaces", Intl. Journal of Shape Modeling, 2001, v1, pp.1-22

Task-Level Assembly Modeling in Virtual Environments

Bernhard Jung

University of Bielefeld, Germany
Faculty of Technology, Knowledge-Based Systems Group
Laboratory for Artificial Intelligence and Virtual Reality

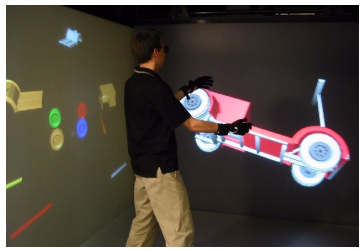
Abstract. This contribution introduces a new framework for assembly modeling in Virtual Reality. Aiming at an easy instructability of the virtual environment, the framework provides a task-level interface which directly maps logical assembly commands to corresponding changes in the geometry scene. For example, the visual assembly of two parts is achieved given only a single command 'connect(a,b)'. This is in contrast to the assembly modeling style of conventional CAD systems which forces the designer to break down each conceptual assembly task into a series of lower-level subtasks. The proposed framework consists of two parts: (1) A knowledge-based model of connection-sensitive part features ("ports") and the connections between them; and, (2), a set of algorithms that define the task-level interface for assembly, disassembly, and adjustment operations. All algorithms are computationally efficient and easily meet the real-time requirements of virtual environments. At the user interface, both direct manipulation and directive interfaces, e.g. based on natural language instructions are supported. A family of implemented VR-systems, including CAVE and Internet-based applications, demonstrates the feasibility of the approach.

1 Introduction

One of the commercially most relevant application areas of Virtual Reality (VR) are CAD-tasks that involve the design and evaluation of so-called virtual prototypes. A main goal is to enable the testing of different aspects of part design and behavior without the need to manufacture a physical prototype first [1,9]. Whereas in most of today's industrial applications, VR is still used as mere visualization tool for prototypes modeled in external CAD systems, recent years have seen an intensified research interest in *immersive assembly modelers* which could ultimately enable the design of virtual prototypes from within the virtual environment. Our main interest are immersive assembly modelers that not only provide powerful modeling functionalities but also – and in particular – can be easily instructed, e.g. by means of intuitive language- and gesture-based instructions. Figure 1 shows examples of the kinds of virtual environments considered.

While conventional CAD systems provide powerful assembly modeling functionalities attempting to transfer the CAD-based assembly modeling style to VR

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 721–730, 2003.
© Springer-Verlag Berlin Heidelberg 2003



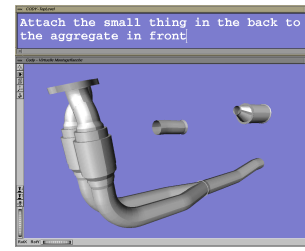


Fig. 1. Virtual environments involving the assembly of CAD-based parts.

seems rather misguided. Two of the CAD systems' inadequacies concern their their insufficient real-time and collision handling capabilities. For the present discussion, a further aspect is central: In CAD systems, the conceptually atomic process of adding a part to an assembly is modeled by the successive definition of geometric relations (constraints) between part surfaces. For example, to insert a screw into a nut usually three constraints such as `concentric`, `coincident`, and `parallel` need to be defined. Thus, when adding a part to an assembly, the cognitive load is on the user who has to mentally break down the task into a series of lower-level constraint definitions. Moreover, the constraints can be defined between arbitrary surfaces of the parts (where typical industrial models are easily composed of hundreds of surfaces). This surface selection process increases the mental load of the user even more. In desktop-based CAD systems, an assembly relation browser helps the user to understand the structural constraints of the assembly. In VR, in contrast, the internal structure of the assemblies is usually hidden from the user. This, in turn, requires the availability of alternative, higher-level assembly modeling methods.

The remainder of this contribution, after reviewing related work, introduces such an alternative approach to assembly modeling based on the notion of *task-level control* [22]. Building on a knowledge-based model of part connectivity, these methods achieve e.g. the visual assembly of two parts in the virtual environment given only a single command *connect(a,b)*. Besides assembly operations, similar methods for disassembly and adjustments of assemblies are supported. The main goal of the proposed framework is, however, not so much a full reconstruction of the powerful CAD-based modeling functionalities in VR (although it is in fact more powerful than many of the existing immersive assembly modelers); instead, it's main purpose is to support the easy instructability of VR-based virtual prototyping environments.

2 Related Work

VR systems labeled in the literature as "virtual assembly" can be roughly divided into two categories (of which only the latter provides modeling functionalities).

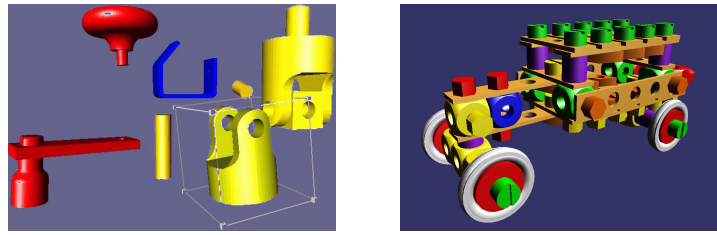


Fig. 2. Left: Direct manipulation based desktop user interface for task-level assembly. Right: A car model built from 124 parts.

A first kind of virtual assembly systems enables a VR-based assembly, typically using one or two data gloves, of prototypes defined in external modelers. The purpose of such systems is e.g. to check the general (dis-)assemblability of the design, part accessability, tool usage, generation of sequences and trajectories of assembly operations, and so on. Examples include the VADE system [10], Zachmann's work with Virtual Design II [6], and applications of DBView developed at DaimlerChrysler [4]. Empirical studies confirm the usefulness of VR-based systems for assembly planning and verification as compared to traditional blueprints or nonimmersive desktop systems [2].

The second kind of virtual assembly systems, immersive assembly modelers, additionally supports the combination of CAD-based parts to novel assemblies. Typically, such systems implement some sort of snapping mechanism which automatically completes an assembly operation when two parts are moved close enough in the virtual environment. Some systems are restricted to assemblies composed of a few types of primitives, e.g. boxes or cylinders [5] or Lego bricks [13]; more recently, the SeamlessDesign system used a larger set of more complex primitives [14]. At Volkswagen, an immersive VR system for interactive cable layout has been developed [20]. Other systems explicitly model part connectivity by means of simple reference (snap) points on the object's surfaces [8]. The approach of [7] uses an automatic alignment based on the objects' bounding boxes for snapping two parts together. ISAAC implements snap-to-grid and snap-to-orientation methods for precision placement of parts [17]. The MAESTRO system uses "sensitive polygons" to constrain the objects' movement during the coarse positioning phase [19]. Another class of immersive assembly modelers uses constraint-based approaches very similar to those of modern CAD systems, e.g. ICBAM [21] or the constrained virtual environment developed in Salford [16]; here, snapping can be implemented by a combination of automatic detection and assertion of geometric constraints. Snapping addresses some of the issues raised in the introduction concerning the transfer of CAD-style assembly modeling to VR; however, as the constraint detection mechanism constantly has to check for spatial relations between arbitrary part surfaces, such systems appear to be restricted to assemblies consisting of a small number of industrial-complexity parts. It is further unclear, how conflicts between asserted constraints and object

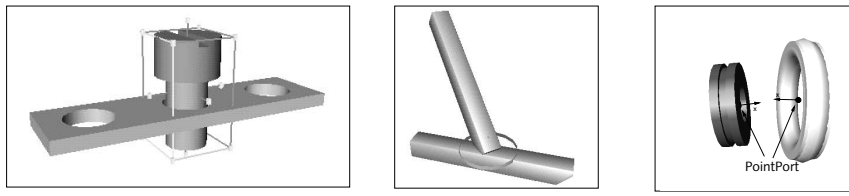


Fig. 3. Extrusion, Plane, and Point port connections.

collisions could be resolved. In all of the above systems, the interactive assembly simulation is triggered by direct manipulation, e.g. using one or two datagloves.

Our approach to immersive assembly modeling aims to overcome some of the limitations of existing systems by combining the following properties: (1) it supports not only assembly and disassembly but also adjustments in the relative positioning of parts; (2) it scales well for assemblies consisting of a large number (e.g. in the order of hundreds) of parts; and, (3) it supports not only direct manipulation, but also natural language instructions. The last aspect, and even more so the combination of these features makes our approach unique.

3 A Knowledge-Based Model of Ports and Connections

The proposed framework for task-level assembly modeling utilizes a knowledge-based model of connection-sensitive part subvolumes or *ports*, e.g. the shaft of a screw or the inside thread of a nut, and constraints imposed by different kinds of *connections*, e.g. screwed, inserted, or welded. Taxonomies with predefined port and connections concepts capture the most common mating properties of CAD-based parts which facilitates the reuse of once developed models.

The *port* taxonomy is organized around geometric and mechanical properties which differentiate the port concepts at the higher and, resp., lower levels of the taxonomy. At the top-level, the taxonmoy is divided into the following concepts:

- *Extrusion ports* model connection properties of object subvolumes with extrusion geometries. For example, extrusion ports model connection properties of objects involved in peg-in-hole type assembly operations. In general, connections between extrusion ports afford one translational and one rotational degree of freedom. Extrusion ports are differentiated into *male ports*, such as the shaft of a screw and *female ports*, e.g. the hole of a nut. Matings between two extrusion ports generally invole one male port and one female port. For two extrusion ports to be connected, they must be collinear, parallel (in same or opposed direction), and their capacities must overlap.
- *Plane ports* model connection properties of planar object surfaces. Connections between two plane ports afford two translational and one rotational degree of freedom. For two plane ports to be connected, they must be coplanar, anti-parallel, and their capacities must overlap.

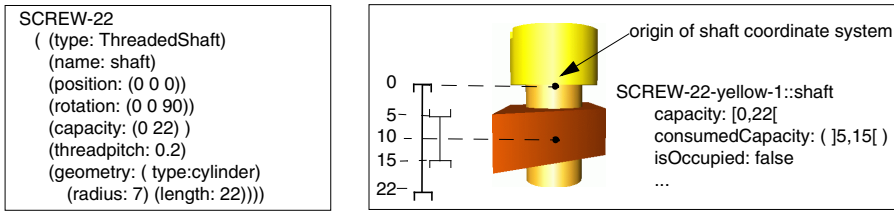


Fig. 4. Left: Port definition for a screw with a 22mm shaft. Right: In concrete assembly tasks, among other dynamic attributes, a list of consumed capacities is maintained.

– *Point ports* model point-like object connections that induce no translational and one rotational degree of freedom when objects are connected. Connected point ports are coincident and parallel. Point ports can be understood as borderline cases of both extrusion and plane ports but introduce a separate concept that has conceptual and computational advantages.

The port taxonomy currently consists of 20+ concepts; it is complemented by a taxonomy of connection concepts that define further constraints concerning the relative movement of connected objects. E.g. screw-in-hole insertions can be defined with either independent or dependent rotational and translational degrees of freedom (in the latter case the screw needs to be rotated in order to be inserted into the hole). Or, a welding-type of connection can be defined as one that allows no transformational degree of freedom at all. The connection taxonomy mirrors a taxonomy developed by Roth in the field of mechanical engineering [18].

Figure 4, left, shows how the CAD model of a screw is enriched with port information. The port definition makes reference to the port taxonomy – the screw's shaft thus inherits e.g. the property to take part in screwing-type operations – and defines additional part-specific, dimensional information. During an on-going assembly task, among other dynamic attributes, a list of consumed capacities is maintained for each port. The consumed capacity of a port is used e.g. by the assembly algorithms to prevent matings with third parts at those positions. A detailed description of the port and connection representations can be found in [11].

4 Algorithms for Task-Level Assembly Modeling

Aiming at a task-level instructability of virtual prototyping environments, a comprehensive set of algorithms has been developed that transforms high-level specifications of various assembly-related tasks to corresponding changes of the virtual environment's lower-level graphics representation. In particular, these algorithms accomplish the assembly and disassembly of parts and aggregates as well as adjustments of already connected parts along their remaining degrees of

connect(a,b)

1. Heuristically refine the input data, i.e. select ports, hotspots, assembly direction (if not already present in input descriptions) based on current snap policy
2. Perform preliminary mate of the two parts, account for already consumed capacities of ports being mated
3. Handle collisions possibly resulting from step 2
4. Infer the new connection relationships and assert them in the connection graph

Fig. 5. The connect algorithm for task-level assembly.

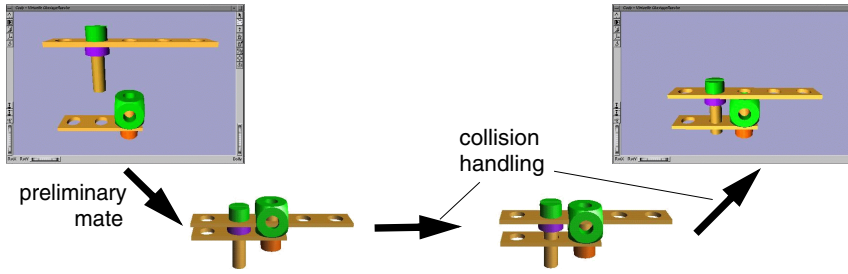


Fig. 6. Phases of the connect algorithm.

freedom. Building on the port and connection representations described above, the algorithms operate on and manipulate several part, port, and connection representations at once. The algorithms further incorporate polygon-precision collision handling methods and thus ensure physical realism to a large extent. Due to space limitations, the present discussion is focussed on the algorithm for task-level assembly. A more detailed pseudo code description of all algorithms can be found in [11].

The connect algorithm achieving a task-level assembly is shown in Figure 5. A first and important issue concerns the data structures of its input arguments. Just like the other task-level algorithms, the connect algorithm expects only information as input that can be derived from natural language (NL) instructions. NL instructions, however, can specify the task at very different levels of granularity. E.g. in the assembly task shown in Figure 6, the objects to be connected could have be described as individual parts or as aggregates (*insert the screw into the bar* vs. *insert the upper aggregate into the lower one*); or, a specific port may have been specified (*...into the middle hole of the bar*); further, an assembly direction *...from above*) or the amount of the insertion (*insert the screw fully* ...) may have been provided. Correspondingly, the connect algorithm is called with parameters of a flexible data-structure. At a minimum, the set of involved parts must be specified. Optionally, further fields describing e.g. a specific port or

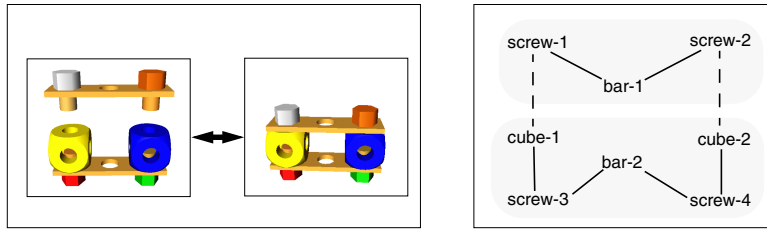


Fig. 7. Each assembly operation may establish multiple connections; in the connection graph, two components are merged. Likewise, each disassembly operation may require the resolution of multiple connections; in the connection graph, the component representing the original aggregate is split in two components.

a hotspot (snap point) on that port can be supplied. In step 1 of the algorithm, the unfilled fields of the two input parameters are heuristically filled, based on a globally defined *snap-policy*. In Figure 6, e.g., a snap-policy is assumed that tries to mate the closest ports, in an assembly direction that matches best the moved objects' current orientation, and with a preference to insert extrusion ports as far as possible into another.

Steps 2 and 3 of the connect algorithm are illustrated in Figure 6. In step 2, the parts are preliminarily mated based on the locally available information about consumed port capacities. While the mating conditions are locally satisfied for the two ports, a collision occurs between two other parts in the assembly. Therefore, step 3 performs a collision avoidance procedure. The screw is pulled out (in inverse assembly direction) of the bar until a collision-free state is reached. Collision avoidance operates at polygonal precision by utilizing detailed geometry information about the CAD-based parts. The final step 4 of the connect algorithm tests the resulting assembly for new connection relations and asserts them in the *connection graph*. Figure 7 shows an example, where multiple connections result from one assembly operation.

Figure 7 can also be read as an example of an disassembly operation. Just as a single assembly task may result in the establishment of multiple new connections, a single disassembly task may involve the breakup of several connection relationships. Put slightly differently: While assembly operations merge two components of the connection graph into one component, disassembly operations split one component of the connection graph into two. Accordingly, the disconnect algorithm, shown in Figure 8, first splits the connection graph into two components, and then fully separates the two components in the graphics scene. Collision avoidance ensures that the disassembled parts do not penetrate any other objects in the virtual environments.

Finally, besides assembly and disassembly operations, the present framework for task-level assembly modeling further supports adjustment operations that modify the relative placement of some parts in an assembly w.r.t. other, stationary parts. Both rotational and translational adjustments are supported, cor-

disconnect(a,b,$dirhint$)

1. Split the connection graph of the aggregate containing a and b into exactly two components a' and b' by retracting one or more connection relations. All retracted connection relations must support the same disassembly direction dir, which must approximately match the optional $dirhint$, if provided.
2. Graphically separate the parts a' from b' in direction dir, such that their distance exceeds a threshold and the removed parts a' do not collide with any other parts in the environment.

Fig. 8. Disconnect algorithm for task-level disassembly.

responding to the degrees of freedom of port connections; see Figure 9 for an example of a translational adjustment. The adjustment algorithms are similar to the disassembly algorithm in that they first (although temporarily) split the connection graph of the modified aggregate into two components of the transformed and stationary parts. The following graphical displacement, however, is limited such that in the resulting state the mating conditions of the modified connections still hold and no parts are separated from the assembly.

5 Conclusions

We have presented a novel framework for task-level control of virtual environments for assembly modeling. As representational foundation, a knowledge-based model of ports and connections was devised that captures stereotypical mating properties of CAD-based parts. Compared to conventional lower-level CAD descriptions, port and connection representations provide a more abstract modeling means closer to the human conceptualization. Building on the port and connection representations, a set of algorithms was developed that accomplishes a task-level control of various assembly related operations such as assembly, disassembly, and adjustments. Desirable properties of the algorithms include their real-time capabilities and built-in collision avoidance. Further, the algorithms

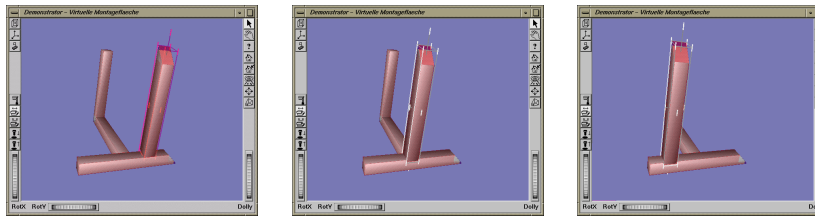


Fig. 9. Translational adjustment by direct manipulation: The selected part is moved, constrained by the mating conditions of its connection to another part.

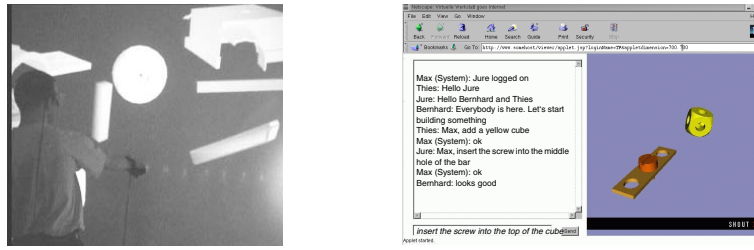


Fig. 10. Left: Immersive assembly modeling in a large-screen environment using speech and gesture. Right: A web-browser interface based on typed natural language.

can accommodate a variety of information present in verbal (or gestural) instructions of the user. They are, however, also able to perform the task when only minimal information is provided in the instruction.

The framework for task-level assembly modeling has been integrated into several complete virtual assembly systems. In the large screen environment shown on the left of Figure 10, natural language and gesture instructions enable an interaction with both close and distant parts [15]. In web-browser based environments, as the one shown on the right of Figure 10, the updates rates of the visualization are particularly slow; here, the natural language interface proves advantegeous as it allows for instantaneous updates of the virtual environment as opposed to direct manipulations that require continuous updates [12]. Of course, as illustrated in Figures 2 and 9, the task-level interface to virtual assembly also supports user interfaces based on direct manipulation. Current work aims at extending the task-level interface to scaling operations that preserve the shape of certain subparts [3].

Acknowledgment. This work is partly supported by the Deutsche Forschungsgemeinschaft (DFG).

References

1. J. A. Adam. Virtual reality is for real. *IEEE Spectrum*, 30(10):22–29, 1993.
2. A. Banerjee, P. Banerjee, N. Ye, and F. Dech. Assembly planning effectiveness using virtual reality. *Presence*, 8(2):204–217, 1999.
3. P. Biermann, B. Jung, M. E. Latoschik, and I. Wachsmuth. Virtuelle Werkstatt: A plattform for multimodal assembly in VR. In *Proceedings Virtual Reality International Conference (VRIC 2002)*, pages 53–62, 2002.
4. M. Buck and E. Schömer. Interactive rigid body manipulation with obstacle contacts. In *Proceedings 6th International Conference in Central Europe on Computer Graphics and Visualization*, pages 49–56, 1998.
5. J. Butterworth, A. Davidson, S. Hench, and T. M. Olano. 3DM: A three dimensional modeler using a head-mounted display. In *Procedures of the Symposium on Interactive 3D Graphics*, pages 134–138. ACM Press, 1992.

6. A. G. de Sa and G. Zachmann. Virtual reality as a tool for verification of assembly and maintenance processes. *Computer Graphics*, 23(3):389–403, 1999.
7. P. Drews and M. Weyrich. Interactive functional evaluation in virtual prototyping illustrated by an example of a construction machine design. In *IECON-98 – Proc. 24th Annual Conference of the IEEE Industrial Elctronics Society*, volume 4, pages 2146–2151. IEEE, 1998.
8. J. Gausemeier, M. Grafe, and R. Wortmann. Interactive planning of manufacturing systems with construction sets. In *IECON-98 – Proc. 24th Annual Conference of the IEEE Industrial Elctronics Society*, volume 4, pages 2146–2151. IEEE, 1998.
9. R. Gupta, D. Whitney, and D. Zeltzer. Prototyping and design for assembly analysis using multimodal virtual environments. *Computer-Aided Design*, 29(8):585–597, 1997.
10. S. Jayaram, U. Jayaram, Y. Wang, K. Lyons, and P. Hart. VADE: A virtual assembly design environment. *IEEE Computer Graphics and Applications*, 19(6), 1999.
11. B. Jung. Intelligent virtual prototyping environments: Foundations and applications of task-level CAD in virtual reality. Habilitation Thesis, Faculty of Technology, University of Bielefeld, 2002.
12. B. Jung, T. Pfeiffer, and J. Zakotnik. Natural language based virtual prototyping on the web. In *Proceedings Structured Design of Virtual Environments and 3D-Components*. Aachen: Shaker, 2002.
13. K. Kiyokawa, H. Takemura, Y. Katayama, H. Iwasa, and N. Yokoya. Vlego: A simple two-handed modeling environment based on toy blocks. In *Proceedings ACM Symposium on Virtual Reality Software and Technology (VRST '96)*, 1996.
14. K. Kiyokawa, H. Takemura, and N. Yokoya. SeamlessDesign: A face-to-face collaborative virtual/augmented environment for rapid prototyping of geometrically constrained 3-d objects. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Vol. 2*, pages 447–453, 1999.
15. M. Latoschik, M. Fröhlich, B. Jung, and I. Wachsmuth. Utilize speech and gestures to realize natural interaction in a virtual environment. In *IECON'98 - Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2028–2033, 1998.
16. L. Marcelino, N. Murray, and T. Fernando. A constraint manager to support virtual maintainability. In *1st Ibero-American Symposium in Computer Graphics*, 2002.
17. M. A. Milne. ISAAC: A meta-CAD system for virtual environments. *Computer Aided Design*, 29(8):547–553, 1997.
18. K. Roth. *Konstruieren mit Konstruktionskatalogen*, volume I. Springer-Verlag, Berlin, 2. edition, 1994.
19. R. Steffan and T. Kuhlen. Maestro - a tool for interactive assembly simulation in virtual environments. In B. Fröhlich, J. Deisinger, and H.-J. Bullinger, editors, *Immersive Projection Technology and Virtual Environments*, pages 141 – 152. Springer Verlag, 2001.
20. M. Symietz. *Echtzeitbasierte Generierung und Verlegung von Leitungsobjekten in einem digitalen Fahrzeugmodell mit einem Virtual-Reality-System*. PhD thesis, University of Bielefeld, Faculty of Technology, 2000.
21. M. R. Thompson, J. H. Maxfield, and P. M Dew. Interactive virtual prototyping. In *Proceedings of Eurographics UK '98*, pages 107–120, 1998.
22. D. Zeltzer and S. Gaffron. Task-level interaction with virtual environments and virtual actors. *International Journal of Human-Computer Interaction*, 8(1):73–94, 1996.

Design of a New Test Part for Benchmarking the Accuracy and Surface Finish of Rapid Prototyping Processes

Hong-Seok Byun and Kwan Heng Lee

Department of Mechatronics, Kwangju Institute of Science and Technology, Gwangju, 500-712, Korea
{hsbyun, lee}@kyebek.kjist.ac.kr

Abstract. This paper presents a new test part that can benchmark various RP techniques. Using the proposed test part, geometrical and dimensional characteristics, as well as the surface finish, can be evaluated. The test part is designed with consideration given to RP process mechanisms, a questionnaire and existing test parts. It includes many features, such as holes, columns, spheres, steps and walls. Some features are included with varying details to determine the finest feature size that can be manufactured by a specific RP process. The test part was made on several major RP machines and measured by a coordinate measuring machine (CMM). The evaluation of the accuracy, as well as the surface finish, is discussed for different RP processes.

1 Introduction

Fabrication of a physical model from CAD data, usually by material addition using a layered manufacturing method, is usually referred to as rapid prototyping (RP). RP is relatively a new technology, which was first commercialized by 3D Systems in 1987. Generally, RP processes begin with a STL file that describes a model with many triangular facets. The fabricated RP parts can be used to visualize or to verify designs, to check form, fit and function, or to produce master patterns for mold-making.

Over a period of time, RP systems have emerged with scores of different techniques. Regarding their implementation principles, RP systems can be largely classified into several basic layered manufacturing technologies [1]; polymerization of liquid resin, sintering of powdered materials, lamination by bonding, extrusion of thermoplastic materials, and jetting of material or binders. It is obvious that these RP processes have grown to become an integral part of new product development in various areas of manufacturing industry such as automobiles, electric home appliances and aerospace. Especially, they have been used as a key technology in concurrent engineering to reduce the cycle time of product development.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 731-740, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Related Works

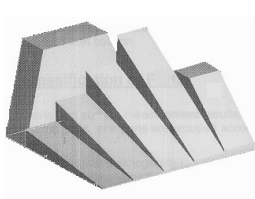
Studies related to RP processes tend to focus on many aspects, including benchmarking the capability of the system and quality assurance. For benchmarking, a test part is usually required, and many studies have developed test parts prior to this study.

The project funded by the European Community reported to determine the levels of dimensional accuracy and surface finish achievable with various layer-manufacturing processes. The test part geometry, shown in Fig.1(a), has planar surfaces that include various angles in the x, y and z directions. The surfaces are disposed to facilitate measurement. However, it is impossible to evaluate various geometric characteristics since it includes no features other than angles [1,2].

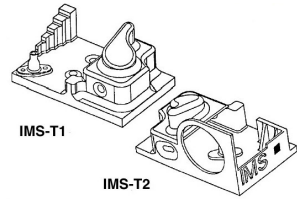
The Intelligent Manufacturing Systems (IMS) project reported the capabilities of RP systems, including build time, build volume, system cost and accuracy expectation, through IMS test parts built by many companies. Although the two IMS test parts shown in Fig.1(b) were created to provide a variety of geometrical features, it is difficult to quickly acquire measurement data to evaluate geometric characteristics or dimensional accuracies on the account that they have many freeform shapes and fillets or rounded features that are not easily measured by a coordinate measuring machine (CMM) [3]. Juster et al. proposed a new benchmark part (Fig.1(c)) for evaluating the differences in the accuracy of major machines. Using this part, which includes various features, it is possible to evaluate the geometric and dimensional characteristics according to features of different sizes. However, the base plane is so big that curling of the base surface, which then affects the measurement, can occur. Since it also has freeform features like the IMS parts, it is not simple to obtain and estimate the measurement data rapidly. The essential primitives, such as a sphere and a large cylinder, were excluded as well [4].

The standard part proposed by 3D Systems (Fig.1(d)) was designed to facilitate measurement since features on the base surface are almost aligned symmetrically right and left. Due to the lack of necessary features to satisfy various evaluation lists, however, evaluation is very restrained, and measurement data in each of the x, y and z directions cannot be obtained because most of the features are arranged in one direction. Ippolito et al. compared the accuracy and surface roughness of major RP techniques by using this part [5].

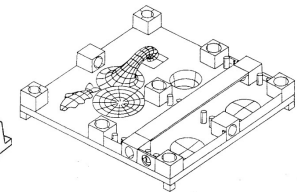
Zhou et al. developed a standard sample (Fig.1(e)) to provide for the benchmarking of some features. Even though the sample part has the essential primitives for the evaluation of features' dimensional accuracy, since their dimensions are so small and the axes of all the features are aligned in one direction, dimensional characteristics based on features of different sizes and measurement data in other directions cannot be acquired. Besides, the distance between features is very small, which makes it difficult to obtain measurements by a CMM [6]. Nakagawa et al. developed the benchmark part in Fig.1(f) to evaluate the performance of commercial RP systems. Unlike other conventional test parts, small features, such as walls and holes, were included in the benchmark part to assess the ability of the RP systems to make fine details. However, the walls and holes are aligned along one axis and the benchmark part consists of only several simple features, and most of all, its size and features' size are so small that we cannot acquire data with respect to differences in their size [7].



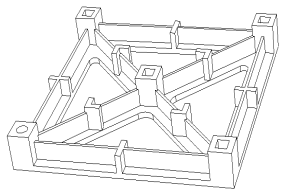
(a) 71mm x 75mm



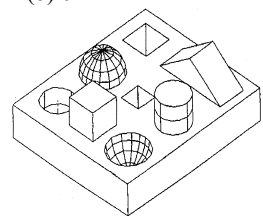
(b) 6in x 4in



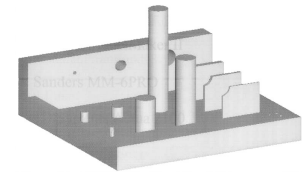
(c) 240mm x 240mm



(d) 240mm x 240mm



(e) 36.5mm x 31mm



(f) 50mm x 50mm

Fig. 1. Existing test parts (length x width)

Table 1. Geometric items evaluated by existing test parts

Geometric item \ Test part		(a)	(b)	(c)	(d)	(e)	(f)
True position		○	○	○	○	○	○
Cylindricity				○	○	○	○
Circularity			○	○	○	○	○
Angularity		○	○	○		○	
Concentricity				○			
Perpendicularity			○	○	○	○	○
Flatness			○	○	○	○	○
Parallelism			○	○	○	○	○
Sphericity	In					○	
	out					○	

The drawbacks of existing test parts used for respective studies can be summarized as follows. They are inefficient in the viewpoint of the i) Part size: the parts are too big or too small. For the large parts, they will lead to an increase in consumption of expensive materials and the possible curling of the part, while the small parts present a difficulty in the measurement and a lack of various features of different sizes. Thus, it is desirable that an appropriate part size be selected based upon the process parameters chosen at the beginning of its design. ii) Robustness of data: a CMM has difficulty in obtaining exact measurement data of features set up on a base surface because curling of the base surface has not been taken into account, iii) Deficiency in the arrangement of features: it is not enough to acquire measurement data along only the x, y or z direction, *i.e.*, features can be set up in all directions, iv) Redundancy: it is wasteful to evaluate and obtain measurement data from redundant features, and v) Measurability: freeform shapes are included in some standard parts, but they are difficult to be measured and evaluated. Table 1 shows the important geometric tolerance items that can be evaluated from a test part. It can be seen that none of the existing test parts satisfy all the items. In this paper, a new test part that compensates for the drawbacks of existing test parts is proposed.

3 A New Test Part Design

3.1 Overall Design Procedure

The test part, including some dimensional features, needs to be designed in order to estimate the geometric characteristics and surface roughness of the model. The ideal test part should have the following properties [8]:

1. Include small, medium and large features,
2. Include both inside and outside components,
3. Should not take too long to build,
4. Should not require a large amount of material to be built,
5. Easily and quickly measurable by a CMM, and
6. Include many real features (cube, hole, wall, cylinder etc).

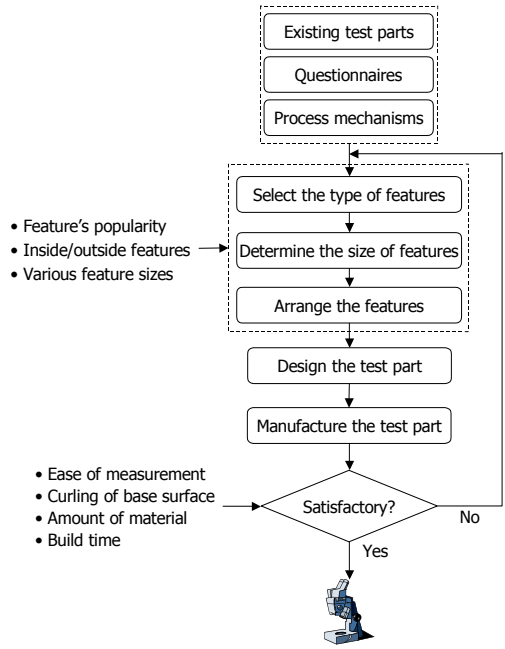


Fig.2. The overall design procedure of the test part

Besides, the test part should also have evaluation features to assess whether or not it is possible to manufacture fine features under a specific RP process. Accordingly, these fine features should be set up in all axes. The minimum dimensional size of fine features can be varied considering the improvement in process mechanisms and the resolutions of RP machines. All these properties are considered in the design of the new test part. In addition, data obtained by questionnaire are utilized in designing the test part. The overall design procedure is illustrated in Fig.2.

3.2 Arrangement of Features in the Test Part

When designing the test part, the arrangement of features should be determined with respect to the x, y and z directions after selecting the type and size of features. However, the features do not have to be arranged in all directions. It is desirable to remove any unnecessary arrangement of features. The arrangement of features is determined according to the following design rule.

$$\begin{aligned} &CF_i \supset (CF_j \cup CF_k) \\ &i = x, y \text{ or } z \\ &j = x, y \text{ or } z \\ &k = x, y \text{ or } z, \text{ but } i \neq j,\ j \neq k,\ i \neq k\,. \end{aligned} \tag{1}$$

It means that if the union of CF_j and CF_k is the subset of CF_i, the direction of the arrangement of a feature is in *i*-direction, and if it does not, the direction is set to *i*, *j* and *k*. Each cross sectional shape CF_i, CF_j, or CF_k is obtained by cutting a feature perpendicular to the z (or build) direction when it is located on the *jk*-, *ik*- and *ij*-plane, respectively. If cross sectional shapes of a feature satisfy eq.(1), the feature needs to be arranged only in the *i*-direction; for example, a sphere or a rectangular hole is arranged only in one direction (*e.g.*, z direction). If they do not satisfy eq.(1), it is arranged in all directions, *i*, *j*, and *k*. The direction of the arrangement of a feature can easily be determined using this design rule.

Finally, after determining the directions of the features, they are arranged on the base surface. Arrangement of features on the base surface plays an important role in facilitating measurement. Also, it is important to consider curling. It may occur due to inappropriate temperature distribution on a part or the differences in the coherence of material in printing of binders, curing or sintering. It often leads to problems in measurement. Therefore, care must be taken to prevent it. In this study, curling was reduced to a great extent by providing small steps at the bottom of the base surface and a uniform distribution of the features on the base surface. Fig.3 shows the arrangement of features determined according to the design rule.

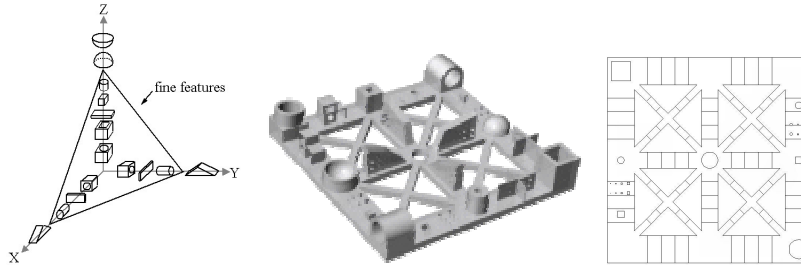


Fig.3. (a) Features to be included in the test part, (b) the proposed test part and (c) small steps at the bottom of the base surface

A test part illustrating the dimensions of constituent features has been designed as shown in Fig.3. The size of test part was determined to be 150mm x 150mm x 25mm, and the included features were designed considering the properties mentioned in

section 3.1. Table 2 shows the components included in the test part and corresponding evaluation items.

Table 2. Included component features and corresponding evaluation items

Component		Evaluation items
Square		External linear accuracy, parallelism, perpendicularity
Cylinder		External roundness, cylindricity, concentricity
Angle		Angularity, surface roughness
Wall		Small linear accuracy (thickness)
Sphere		Internal or external sphericity
Base surface		Flatness, surface roughness
Fine features		The ability to make details
Hole	Square	Internal linear accuracy, parallelism, perpendicularity
	Round	Internal roundness, cylindricity, concentricity, aspect ratio

4 Experimental Results

4.1 The Relative Percent Errors of Accuracy

The measurement data of diverse features set up in the x-, y- and z-directions that cannot be obtained with conventional test parts can be acquired with the newly proposed test part. In this paper, the dimensional accuracy, fabrication of fine features, and surface roughness are evaluated. The dimensional accuracy, which is one of the more important parameters for evaluating the quality of the final product, is described in terms of the relative percent error of accuracy (RPEA). Mean values obtained through the measurements are used to calculate the RPEA using eq.(2).

$$\bar{x} = \sum_{i=1}^{n} x_i / n \quad \text{and} \quad RPEA(\%) = | \bar{x} - x_0 | / x_0 \ . \tag{2}$$

Where, index i is the i-th dimension data obtained when the same features are repeatedly measured, n is the number of random coordinate points for a feature, $\bar{x}$ is the mean value of measurement, and x_0 is the original CAD data. $|(\ )|$ indicates the Euclidean distance. The RPEA is calculated from each feature included in the test part with respect to the x-, y- and z-directions.

Machines selected for evaluation, the materials used and the slice thickness are listed in Table 3. An appropriate slice thickness was usually chosen after considering the other parameters. Most of the layered manufacturing technologies described in the introduction were selected. Fig.4 shows the relative percent errors of the linear accuracy in each direction. It can be seen that for the smaller dimensions, a larger error resulted. In Z402, the percent error increases rapidly due to the bonding of a large excess material during printing. In the FDM process, the error of small features in z direction is larger than that in x- or y-direction, as compared to the other processes. This is due to the fact that the layer is thicker and its composition is different from that of the other processes. However, the accuracy in the x- and y-directions turns out very good. With features having a size of greater than 14mm, almost all RP processes produced them with less than 1% error.

Table 3. Rapid prototyping techniques for benchmarking

Technique	Material	Vendors	Slice thickness
SLA3500	SL5510	3D Systems	0.1mm
FDM8000	ABS	Stratasys	0.25mm
SLS2500	Duraform P/A	DTM	0.1mm
Z402	ZP100	Z-Corp	0.09mm
LOM1015	LOM Paper	Cubic Tech	0.1mm

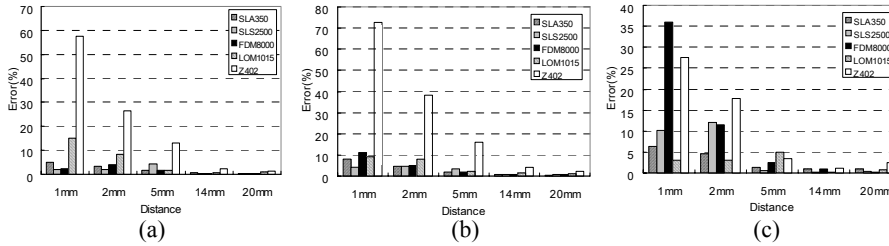


Fig.4. Relative percentage errors of accuracy(RPEA) about x-, y- and z-direction; (a) Error in the x-direction, (b) Error in the y-direction, and (c) Error in the z-direction

4.2 Fabrication of Fine Features

The capability of RP processes is closely associated with its ability to produce very small features. Therefore, the ability of different RP processes to produce very fine details was evaluated using very small features included in the test part and aligned along the x-, y-, and z-axes. Table 4 shows the dimensional limitations of fine features that can be manufactured by a specific process. In Table 4, it can be seen that all RP processes, regardless of the direction of the feature set up, can manufacture fine features having a size of greater than 1mm. It should also be noted that the formation of fine features, under 0.5mm, varies with each RP process and the direction of the feature set up. This results from differences in the resolution along the x-, y- and z-axes. Both SLA and SLS processes were able to make squares and cylinders of 0.2mm in diameter along the build direction, but their measurement data were larger than those of the original CAD data. This can be caused by the laser diameter and beam offset. The LOM process is not included in Table 4 since fine features can be hardly maintained during the decubing process. These results demonstrate that the test part allows for the assessment of dimensional limitations of very small features.

4.3 Surface Roughness

Surface roughness is essential in many applications. In this study, the surface roughness was evaluated by measuring planar surfaces at various angles. The inclined surfaces are set up on a base surface along the x- and y-axes. Inclined surfaces, constructed to have an inclination of 10°, 20°, 40° and 70° from the base surface, can

Table 4. Formation of fine features

	Dir.	Dim.(mm)	SLA 3500	SLS 2500	FDM 8000	Z402
Wall	x, y, z	2.0	Y	Y	Y	Y
	x, y, z	1.0	Y	Y	Y	Y
	x, y	0.5	Y	Y	N	Y
	Z				Y	
	x, y	0.2	N	N	N	Y
	Z		Y	Y	Y	
Square	x, y, z	2.0	Y	Y	Y	Y
	x, y, z	1.0	Y	Y	Y	Y
	x, y, z	0.5	Y	Y	N	Y
	x, y	0.2	N	N	N	N
	Z		Y	Y		
Rectangular Hole	x, y, z	2.0	Y	Y	Y	Y
	x, y, z	1.0	Y	Y	Y	N
	x, y	0.5	Y	Y	Y	N
	Z		N	N		
	x, y, z	0.2	N	N	N	N
Circular Hole	x, y, z	2.0	Y	Y	Y	Y
	x, y, z	1.0	Y	Y	Y	N
	X	0.5	N	Y	Y	N
	Y			N		
	Z				N	
	x, y, z	0.2	N	N	N	N
Cylinder	x, y, z	2.0	Y	Y	Y	Y
	x, y, z	1.0	Y	Y	Y	Y
	x, y, z	0.5	Y	Y	N	Y
	x, y	0.2	N	N	N	N
	Z		Y	Y		

provide sufficient roughness data. The arithmetic average surface roughness is defined as the arithmetic mean of the deviations of the surface profile from the central line (l_m) along the measurement, as expressed in eq.(3), where *y(x)* is the surface profile and *l* is the range of the evaluation.

$$R_a = \frac{1}{l}\int_0^l |y(x)|dx \cdot \tag{3}$$

We assumed that an inclined surface has sharp edges as shown in Fig.5 so as to estimate and compare the experimental results with the theory. If the surface profile is symmetric, as shown in Fig.6, the arithmetic surface roughness has to be calculated for *l* as set out in eq.(4), where $l = t/\sin\theta$.

$$R_a = \frac{t^2}{4l}\cdot\frac{\cos\theta}{\sin\theta} = \frac{t}{4}\cos\theta,\ 0<\theta<\frac{\pi}{2},\ \text{but}\ \theta \neq \frac{\boldsymbol{n}\pi}{2},\ \boldsymbol{n}=0,1,2$$
$$R_a = 0,\ \theta = \frac{\boldsymbol{n}\pi}{2},\ \boldsymbol{n}=0,1,2\cdot \tag{4}$$

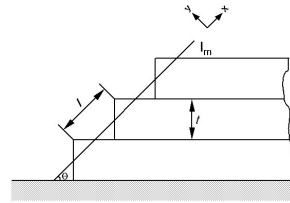
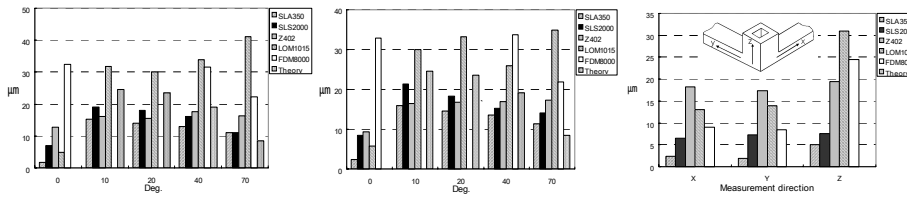


Fig.5. Surface profile of inclined surface

For the measurement, the wavelength cut-off (λ_c) was selected as 2.5mm with an evaluation length of 12.5mm (ISO standard), which is longer than $3*\lambda_c$. Each of the angled surfaces was measured from three different positions. Fig.6 graphically shows the comparison of the theoretical values and the average R_a value for the unpolished sections of the inclined surfaces (angle=10°, 20°, 40°and 70°) and a planar surface (angle=0°). The results show that the planar surfaces are the smoothest and the surface quality of inclined surfaces gets better as the angle increases. For the FDM and LOM processes, the angle did not affect the roughness, which was very large, especially in FDM where the data for 10° and 20° could not be obtained because it exceeds the measurement range of the roughness tester. It should be noted in Fig.6(a) and (b) that, for SLS and SLA, the values of the roughness for the same slope or surface along x and y directions varied to some extent, although the inclined surfaces were manufactured by the same RP process. The differences seen in these values were due to the scanning direction of the laser. Likewise, in the printing process, the planar surface was better, as compared to the inclined surfaces. However, unlike the results of inclined surface in other processes, they gave a similar average roughness because of the porous characteristics of the material. The roughness of side surfaces obtained by different processes is presented in Fig.6(c). It can be noted that there is no difference in the roughness of the side surfaces in the direction of x and y.



(a) (b) (c)

Fig.6. Surface roughness of inclined and planar surfaces; (a) Angle aligned in the x direction, (b) Angle aligned in the y direction, and (c) Angle aligned in the z direction

5 Conclusions

To evaluate dimensional accuracy and the surface finish of a product in rapid prototyping (RP) processes, a new test part that is different from conventional test parts was designed. Unlike the existing test parts that can be applied to a specific RP proc-

ess or only used to estimate several items, the proposed test part includes diverse features of varying sizes in the x, y and z directions and can be applied to different RP processes. It can assess both the various geometrical and the dimensional characteristics with respect to all directions considered. In addition, the dimensional limitations of fine features, namely, the ability to produce fine details, can be evaluated. The test part is designed taking into account ease of measurement, material consumption, build time, curling of the base surface, and size and type of features. Major primitives, such as squares, cylinders and holes, are included in the test part. Experiments are performed with this test part to determine the relative percent error of accuracy, and surface finish, as well as the smallest dimensions that can be attained to evaluate the ability of any system to generate fine details. Using the results of these evaluations, an appropriate RP process can be chosen to manufacture a part of high surface quality. However, it is still difficult to select a suitable RP process to verify designs, to check for form, fit and function, or to produce master patterns from above evaluations. Since each process has its own strength in different applications along with drawbacks for others, additional factors for selecting an appropriate RP process should be considered such as the cost of the part to fabricate, the speed of the RP process, the material properties used, and the machine size and cost. A decision-making program that helps in selecting a RP process utilizing the above factors is currently under development.

References

1. Shellabear, M.: Benchmark Study of Accuracy and Surface Quality in RP Models, Brite-EuRam project BE-2051: Process chains for Rapid Technical Prototypes (RAPTEC), June (1999)
2. Reeves, P. E., Cobb, R. C.: Surface deviation modeling of LMT process – a comparative analysis, Proceedings of 5th European Conference on Rapid Prototyping and Manufacturing, Helsinki, 4-6th June (1996)
3. Aubin, R. F.: A World Wide Assessment of Rapid Prototyping Technologies, Report No. 94-13, United Technologies Research Center, 411, Silver Lane, MS 129-48, East Hartford, CT 06108, January (1994)
4. Childs, T. H. C., Juster, N. P.: Linear and Geometric Accuracies from Layer Manufacturing, Annals of the CIRP, Vol. 43 (1994) 163-166
5. Ippolito, R., Luliano, L., Torino, P. D.: Benchmarking of Rapid Prototyping Techniques in Terms of Dimensional Accuracy and Surface Finish, Annals of the CIRP, Vol.44 (1995) 157-160
6. Zhou, J. G., Herscovici, D., Chen, C. C.: Parametric process optimization to improve the accuracy of rapid prototyped stereolithography parts, International Journal of Machine Tools & Manufacture Vol.40 (2000) 363-379
7. Loose, K., Nakagawa, T.: Benchmarking Various Methods of Layer-Manufacturing Systems in Rapid Prototyping, the 15th Rapid-Prototyping Symposium in Japan, November (1998) 90-100
8. Jacobs, P. F.: Rapid Prototyping & Manufacturing Fundamentals of StereoLithography, First Edition, Society of Manufacturing Engineers (1992)

Automated Scan Plan Generation Using STL Meshes for 3D Stripe-Type Laser Scanner

Seokbae Son and Kwan H. Lee

CAD/CAM Laboratory, K-JIST, 1 Oryong-dong, Buk-gu, Gwangju 500-712, Korea
{sbson, lee}@kyebek.kjist.ac.kr
http://kyebek9.kjist.ac.kr/

Abstract. Reverse engineering and computer-aided inspection technologies play an important role in reducing the lead-time and improving the quality in the production of products containing complex and aesthetic surfaces. In this paper, we propose a laser scan planning system for complex freeform surfaces that is based on STL (STereoLithography) meshes. The developed laser scan planning system consists of three steps and it is assumed that the original CAD model of the part exists. Firstly, the surface model is approximated by STL meshes. The meshes are generated considering the curvature of the model so that more vertices are sampled in high curvature regions. From the mesh model, the region for which the scanning will be performed is selected and the normal vector of each node point is estimated. Secondly, scan directions and regions are determined through *the region growing method*. Also, scan paths are generated by generating a minimum-bound rectangle of points to be scanned. Finally, the generated scan directions and paths are validated by checking optical constraints and the collision avoidance.

1 Introduction

While a conventional engineering process starts with a design concept, in reverse engineering, a product is designed by capturing the shape of a real part. The parts that have complex freeform surface are usually developed through the reverse engineering process. Acquiring the shape data of a physical part is an essential process in reverse engineering and computer-aided inspection [1,2].

As the accuracy and the scanning speed of the laser scanners improved drastically in recent years, they are now getting used in the production floor. However, it is difficult for the human operator to generate a correct measuring plan for a complex freeform part by considering both the geometric characteristics of a part and optical constraints of a laser scanner simultaneously. For CNC-driven laser scanners, it is important to automate the scanning operations to improve the accuracy of the captured point data and to reduce the scanning time as well. It is also a difficult problem to deal with multi-patched freeform surface models. Few research works focused on this topic [3,4].

In this paper, we propose a STL (STereoLithography) mesh based laser scan planning system for multi-patched complex freeform parts and a 3-axis motorized stage

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 741-750, 2003.
© Springer-Verlag Berlin Heidelberg 2003

with a high accuracy for automated positioning and registration. The developed system consists of two modules: a scan plan generation module and a positioning hardware module. It is assumed that the CAD model of the part to be scanned exists. The scan planning system generates the scan plan in three steps. Firstly, points and normal vectors are extracted from the selected region of a CAD model presented in STL format. The STL mesh has been the de facto standard of the input format for rapid prototyping devices. The points are sampled non-uniformly considering the curvature of a part due to the characteristics of STL mesh. Secondly, the scan directions and regions are determined by *the region growing method*. The view angle can be varied to control the quality of the captured point data. Thirdly, the scan paths are generated for each scan direction and the optical constraints such as the depth of field and the occlusion are verified using solid models. To minimize the length of the scan path, *the minimum-bounding rectangle* is adopted. Finally, the developed system is tested and verified using a fan model, which is machined with aluminum.

2 Background and Research Scope

2.1 Laser Scanning System

The laser scanner consists of two major parts: the laser probe and the positioning system. The laser probe is mounted on a 3-axis transportation mechanism and moves along the scan paths that consist of series of predetermined line segments (Fig. 1).

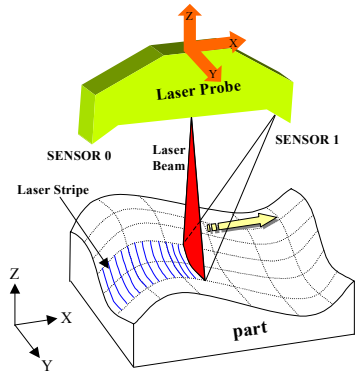


Fig. 1. Laser Scanning Mechanism

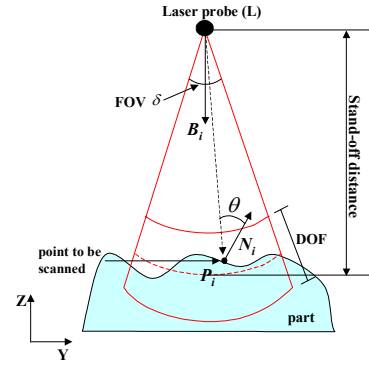


Fig. 2. Constraints for Laser Scanning

During scanning, a series of laser stripes is projected onto a surface patch and the reflected beam is detected by two cameras. Through image processing and optical triangulation, three-dimensional coordinates of the scanned points on the surface of the part are acquired.

When the laser scanner captures an image, the system automatically finds an optical focus while keeping the stand-off distance. Since the laser scanner consists of

optical sensors and mechanical moving parts, various constraints must be satisfied when measuring a point on the part (Fig. 2). In order to capture a point on the part's surface, the following constraints should be satisfied [5]:

(1) View angle: The angle between the incident laser beam and the surface normal of a point being measured should be less than the view angle γ.

$d_i \cdot N_i \geq \cos(\gamma)$, where $d_i = (L - P_i) / |L - P_i|$

(2) Field of View (FOV): The measured point should be located within the length of a laser stripe.

$(-d_i) \cdot B_i \geq \cos(\delta / 2)$, where δ is the FOV angle.

(3) Depth of Field (DOF): The measured point should be within a specified range of distance from the laser source.

$l_{STAND} - l_{DOF} / 2 \leq \|L - P_i\| \leq l_{STAND} + l_{DOF} / 2$

where l_{STAND} and l_{DOF} denotes stand-off distance and DOF length.

(4) Occlusion: The incident beam as well as the reflected beam must not interfere with the part itself.

(5) The laser probe should travel along a path that is collision-free

2.2 Research Scope

The overall procedure of the STL mesh based laser scan planning and automated scanning and registration system for multi-patched freeform surfaces is presented in Fig. 3. The proposed system consists of four sub-steps as follows:

Step 1: Generation of an STL model, selection of regions to be scanned and estimation of normal vector at each node point.

Step 2: Grouping of regions and generation of scan directions

Step 3: Generation of scan path and checking of collision and optical constraints

In step 1, an STL model is generated with a specific tolerance from the given CAD model. Through the STL mesh generation, non-uniform triangles are obtained due to the curvature of a surface model. Then, some regions to be scanned are selected from the STL model and, for the selected region, normal vector of each vertex is calculated. In step 2, the scan directions and the scan regions are generated by analyzing normal vectors. In order to generate the best suitable scan direction and the related region, *the region growing method* is developed. The scan directions and regions can be varied by changing the size of the view angle within the maximum angle given by the machine. In step 3, the scan paths for each scan direction and region are generated using *the minimum-bound rectangle*. They should be verified by checking optical constraints such as the depth of field (DOF) and occlusion. Also, collision avoidance issues are analyzed. When the generated scan plan does not meet the optical constraints, the system modifies the scan plan or generates additional scan direction for the problematic region.

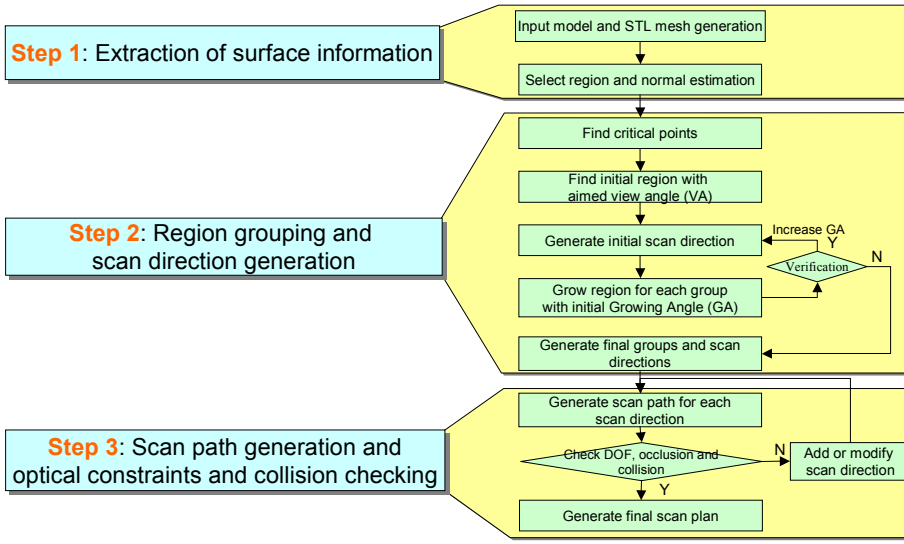


Fig. 3. Overall procedure

3 Scan Plan Generation

3.1 STL Mesh Generation

Although the format of the STL mesh is fixed, STL meshes can be generated through various methods with different parameters. Even though the models are generated with the same tolerance, the number of points and the pattern of the generated triangles are quite different (Table 1).

Table 1. Results of STL mesh generation using commercial CAD systems

Tools	Tolerance	Number of points	Number of triangles	Mesh type
CATIA	0.01 mm	38,773	77,542	Regular
I-DEAS		10,602	20,515	Regular
SolidEdge		15,036	30,068	Irregular
SolidWorks		14,861	29,718	Irregular

As shown in these experimental results, CATIA generates the most points and I-DEAS generates the least, while SolidEdge and SolidWorks reflect the shape of the surface patch very well. For scan plan generation, the data sampled based on the curvature is more effective for some cases. Therefore, among these four programs, SolidEdge, which considers the curvature of the part, is selected for STL model generation.

3.2 Estimation of Normal Vector

After generating an STL mesh model, the regions to be scanned should be determined and the normal vectors should be estimated for each vertex of the selected regions. In the original STL data structure, the normal vector of each triangle exists but there is no normal vector information for each vertex. Hence, the normal vector at each vertex should be calculated from the neighborhood of each vertex. According to the location of each vertex, three types of normal vector estimation methods are presented in Fig. 4. Especially, the normal vector of the corner of the selected region is same with the normal vector of the triangle (Fig. 4 (c)). The estimated normal vector can be used for further applications after the normalization process.

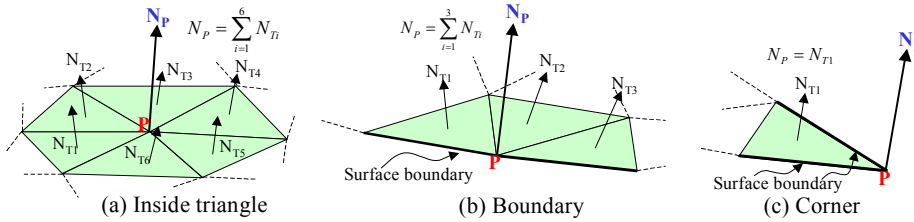


Fig. 4. Normal vector estimation methods.

3.3 Calculation of Laser Scanning Directions and Regions

After selecting regions to be scanned from the STL mesh and calculating normal vector at each vertex, the scan directions and regions are then generated using the sampled points and the normal vectors. From the normal vectors, only two points that have the largest angular deviation are extracted. They are called *the critical points.* For instance, CP_1 and CP_2 in Fig. 5 are the critical points and S_i represents each surface patch. If the angle between the critical points is less than the given view angle, all the points can be scanned in one direction [6].

When the angle between the critical points is larger than the given view angle, more than two scans are required; the scan directions and the scan regions are calculated by *the region growing method*. For the region growing process, the initial value of the view angle, which is the user defined angle value and is smaller than the maximum view angle, is defined. Firstly, from the critical points CP_1 and CP_2, the points that have the angle with a critical point less than the given view angle are grouped respectively. Two initial point groups, *initial group 1* and *initial group 2*, are generated.

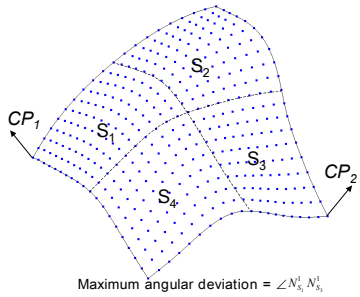


Fig. 5. Finding the critical points

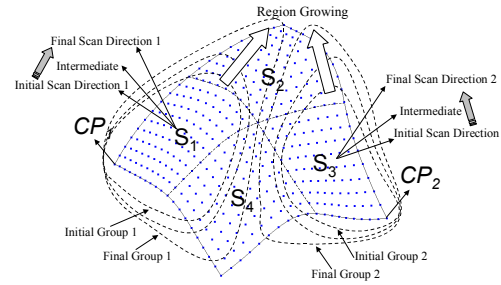


Fig. 6. Region growing process

The two initial scan directions are also generated by calculating the global mean vector of normals of each group (Equation 1).

$$Initial\ \ Scan\ \ Direction\ \ \ ISD_i = \sum_{j=1}^{n} D_j \qquad (1)$$

where D_j is a unit normal vector of each sampling point and n is the number of points in region i.

Secondly, the points that have smaller angular deviation between the initial scan direction and the normal vector of each sample points than the initial view angle are grouped and the scan directions are newly generated by calculating the global mean vector of the normals of the grouped points. After finding these new point groups, the initial view angle is increased and the system finds new point groups and scan directions repeatedly. During the region growing iteration, the initial view angle can be increased up to the maximum view angle while each new group should include the critical point. There are two types of scan regions in terms of the data overlapping. When overlap region exists between two scanning regions in the region growing process, two choices to determine the second scanning region exist and are referred to as *with overlap* and *without overlap*. If it is necessary to avoid the overlapped region, the second point group must exclude the points that already belong to the first point group. However, if an overlapped region is required for further application such as the registration process, the second point group will include some of the points in the first group.

The final point groups and scan directions generated through the iterative region growing process are shown in Fig. 6. For the missed points, the proposed method is applied again in a recursive manner.

3.4 Scan Path Generation

To operate an NC-driven laser scanning system, scan paths are required. A scan path consists of a sequence ID, a starting point, and an ending point. From the scan directions and sampling points for each direction determined in the previous section, the scan paths are calculated.

Firstly, the sampling points for each scan direction are loaded and projected onto a 2D plane that is orthogonal to the scan direction (Fig. 7). After projection, the minimum-bounding rectangle is found by rotating the points set about the scan direction to generate the shortest scan paths (Fig. 8). By using the minimum bounding box, the shortest scan paths can be generated. The rectangle should then be subdivided along the y-direction by the length of a stripe to reduce scanning time.

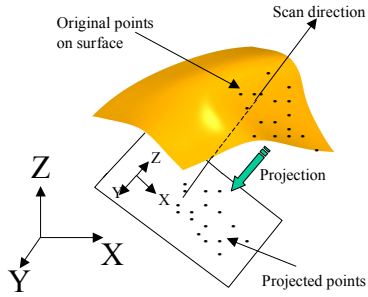


Fig. 7. Projection of the sampling points

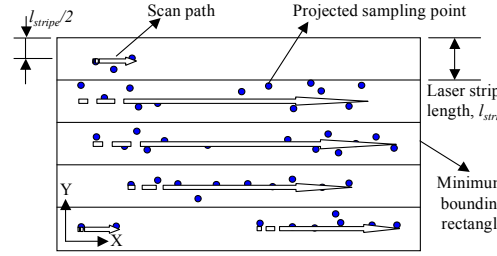


Fig. 8. Generation of the scan paths

Next, for each subdivided rectangle, the scan path is generated by finding the first point and the last points. Finally, the coordinates of the generated scan paths are transformed into the global 3D coordinate system.

3.5 Checking of Optical Constraints and Collision

To implement automated scanning, the scan path must satisfy the optical constraints such as the depth of field (DOF) and the occlusion. The generated scan plan is verified using the DOF and the occlusion checking model as described in Fig. 9 and Fig. 10, respectively. Also, collision between the probe and the part or fixture units should be investigated for safe measure.

Since the scan plan is generated from the sampled points, some regions can be skipped when the surface model is very curved. When all the points on the laser stripe are located in the DOF region, the surface can be completely scanned. To verify the DOF constraints, five points from each line of intersection with the part surface are sampled. The location of each point is verified whether or not they lie in the DOF region (Fig. 9). In this study, since the stripe length is 15mm and the DOF is 40 mm, there is no DOF problem when the view angle is less than 69.4 degrees.

Similarly, in occlusion checking, whether the reflected beam reaches the camera or is reflected or absorbed by another portion of the part is tested. In this study, two cylinders that connect the two CCD sensors and a point on the surface are designed as illustrated in Fig. 10 (a). While moving the probe model along the given scan paths, any interference is checked between the cylinder and the part or the machine itself. If a collision occurs with the cylinder, this indicates that the corresponding CCD camera cannot read the reflected rays of the real system since they are deflected.

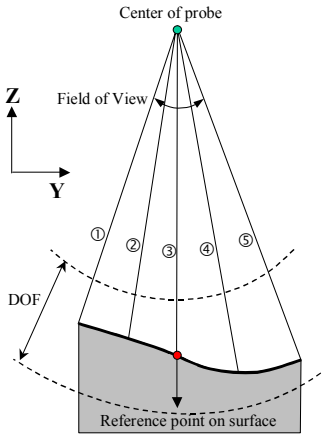


Fig. 9. DOF checking model

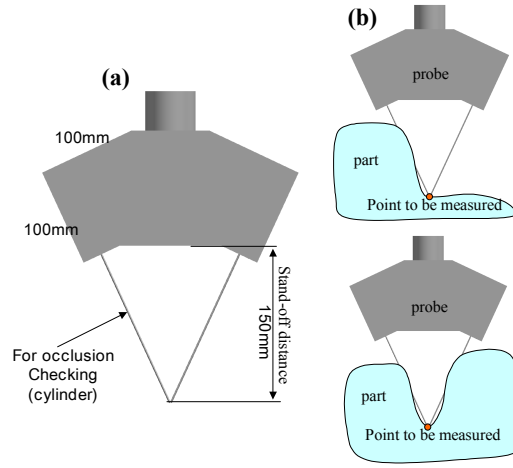


Fig. 10. Occlusion and collision checking model

However, as shown in the upper case of Fig. 10 (b), the left sensor cannot capture the point but the right sensor can capture the point. When one of the two sensors can capture the point, it can be said that there is no occlusion problem because the sensor can automatically be changed when an occlusion problem occurs. But, in the lower case of Fig. 10 (b), the point cannot be read with the given direction because both sensors are blocked.

Finally, a collision-free path is verified using solid models of the probe, the motorized stage, the fixture units and the scanner itself. The probe head is used as a moving part for collision checking. For this process, the Assembly module of SolidEdge is used. Through the collision checking process, we can simulate the whole scanning procedure and can generate an errorless scan plan.

4 Experimental Test

In order to evaluate the proposed algorithms, a fan model, as shown in Fig. 11, is used. The fan model consists of various trimmed and untrimmed freeform surfaces. In the scanning of this type of surface model, it is difficult to determine the correct scan directions and scan regions. For the fan model, the surface patches marked ①, ②, and ③ are selected for the measurement plan generation and automated scanning. Also, the acquired point data will be used for geometric inspection because the selected surface patches have aerodynamic functionalities.

In order to sample points differently and effectively, STL meshes are introduced. For the fan model, STL meshes are generated (Fig. 12 (a)). Since the meshes are sampled based on tolerance, more points are sampled in high curvature regions. After generating STL meshes, the regions to be measured are selected and the normal vector at each point is estimated (Fig. 12 (b)).

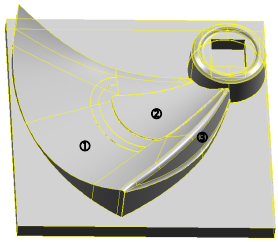


Fig. 11. The fan model

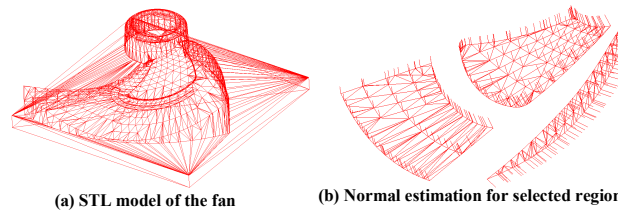


Fig. 12. STL model generation of the fan model

Using the sample points and normal vectors, *the region growing algorithm* which is used for the CAD-directed sample points is applied again. In this case, two scanning regions and scan directions are generated (Fig. 13).

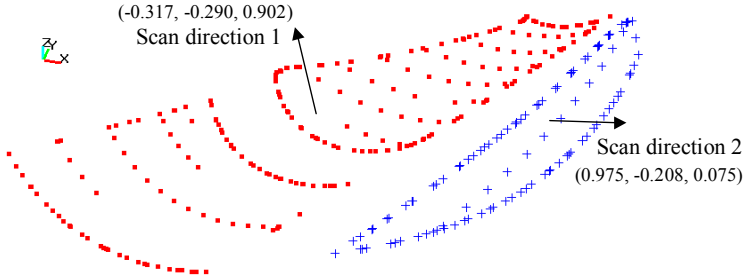


Fig. 13. Result of scan direction generation

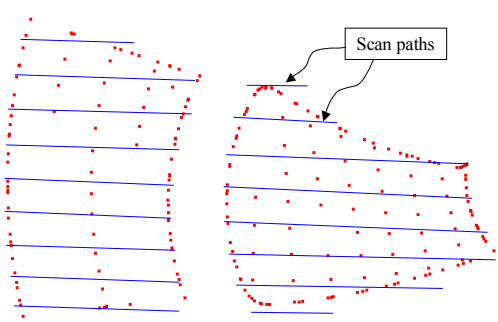


Fig. 14. Scan path generation for the *direction 1*

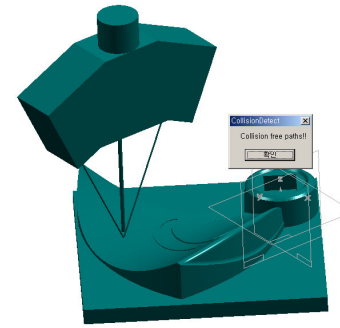


Fig. 15. Collision and occlusion checking

Fig. 14 shows the generated scan paths for *scan direction 1*. In STL meshes, the gap between each point is not restricted because it only depends on the tolerance value. Therefore, the user should divide the points grouped for each surface patch and only the first point and the last point are used for the generation of the scan path for each sub-rectangle.

Finally, the optical constraints and collision are verified in Fig. 15. The fan and the probe are modeled by solid and two cylinders are designed for checking of the occlusion of two sensors. For each scan path, the occlusion and collision violations are

evaluated. In this example, there is no DOF problem because the view angle is less than 69.4 degrees and no occlusion problem occurs since the shape of the given surface patches is not too much curved. Therefore, the initial scan direction can be maintained for this example.

5 Conclusion

In this study, an STL mesh based laser scan planning system for freeform surfaces is presented. The generated scan plan includes the scan direction, region and corresponding scan path. The system developed here is more applicable to inspection than genuine reverse engineering tasks. However, there are several noteworthy characteristics of the newly developed system.

- The system can automatically generate scan plans for the part that has multiple freeform surface patches can be applied to various types of parametric surfaces, including trimmed and untrimmed surfaces after generating STL meshes.
- The quality of the point data can be controlled by changing the size of the view angle while considering the required accuracy based on the characteristics of an optical sensor.
- The optical constraints and collision checking modules can generate a more complete and safe scan plan.

Acknowledgements

This work was supported by grant No. R01-2000-000-00298-0(2002) from the Basic Research Program of the Korea Science & Engineering Foundation.

References

1. Varady, T., Martin, R.R., Cox, J.: Reverse Engineering of Geometric Models-An Introduction, Computer Aided Design. 29 (4) (1997) 255-268
2. Kwan, L., Hyunpung, P., Son, S.: A Framework for Laser Scan Planning of Freeform Surfaces, International Journal of Advance Manufacturing Technology. 17. (2001) 171-180
3. Xi, F., Shu, C.: CAD-based path planning for 3-D line laser scanning, Computer-Aided Design. 31(1999) 473-479
4. Bernard, A., V□ron, M.: Analysis and Validation of 3D Laser Sensor Scanning Process, Annals of the CIRP. 48 (1) (1999) 111-114
5. Zussman, E., Schuler, H., Seliger, G.: Analysis of the Geometrical Feature Detectability Constraints for Laser-Scanner Sensor Planning, International Journal of Advanced Manufacturing Technology. 9 (1994) 56-64
6. Seokbae, S., Hyunpung, P., Kwan, L.: Automated laser scanning system for reverse engineering and inspection. International Journal of Machine Tools and Manufacture. 42 (2002) 889-897

An Efficient Algorithm for Real-Time 3D Terrain Walkthrough

M. Hesse and M. L. Gavrilova

Dept. of Computer Science, University of Calgary
Calgary, AB, Canada T2N1N4
marina,hesse@cpsc.ucalgary.ca

Abstract. The paper presents an efficient and easy to implement algorithm based on ROAM for visualization of large scale terrain models in real-time. The quality and smoothness of terrain data visualization within a 3D interactive environment is preserved, while the complexity of the algorithm is kept on a reasonable level. The main contribution of the paper is in introducing a number of efficient techniques such as *implicit coordinates method* within the patch array representing ROAM and *the viewpoint dependent triangle rendering method* for dynamic level of detail (LOD) updates. In addition, experimental comparison of a number of culling techniques, including a newly introduced method, is conducted. These techniques are incorporated in order to achieve more realistic terrain representation and the real-time level of detail reduction.

1 Introduction

High quality rendering and meshing techniques for displaying complex geographical data, such as terrain models, play an important role in the fast growing domain of CAD oriented towards Geographic Information Systems (GIS). According to a recent review, conducted as a part of Virtual Terrain Project on convergence of the fields of CAD, GIS and visual simulation [13], even today the vast majority of CAD is only 2D blueprints, not a 3D models. The conversion from 2D to 3D is generally a difficult process requiring highly efficient algorithms, and there are practically no freely available tools for 3D CAD that could be useful in GIS. In addition, for exploring different kinds of geographic-based data sets on screen it is necessary to display data at interactive frame rates. Because of the inherent geometric complexity, this goal is often hard to achieve, unless the original data is approximated in order to reduce the number of geometric primitives that need to be rendered. This problem is particularly prevalent in applications dealing with large polygonal surface models, such as digital terrain modeling and visual simulation.

The paper addresses the above problems by introducing an efficient and easy to implement algorithm for visualization of large scale terrain models in real-time. The quality and smoothness of terrain data visualization within a 3D interactive environment is preserved, while the complexity of the algorithm is kept on the reasonable level. The Real-Time Optimally Adapting Mesh (ROAM) approach is used as an underlying model. We introduce a number of efficient techniques such as *implicit coordinates method* within the patch array representing ROAM, *the viewpoint dependent triangle rendering* method for dynamic Level of Detail (LOD) updates and

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 751-761, 2003.
© Springer-Verlag Berlin Heidelberg 2003

some culling techniques. The efficiency is confirmed by experiments conducted on greyscale Digital Elevation Maps.

2 The Motivation

The two traditionally used techniques for surface representation and visualization of terrain models are the TIN (triangulated irregular network) [14], and the uniform grid [5]. They are typically used to address problem of mesh simplification, although having some significant drawbacks preventing on-the-fly generation of multiple levels of detail.TIN models, for example, require a highly extensive computational effort for their generation. Because TINs are non-uniform in nature, surface following (e.g. for animation of objects on the surface) and intersection (e.g. for collision detection, selection, and queries) are hard to handle efficiently. This factor is especially important in many applications, such as games and CAD, where dynamic deformations of the mesh may occur. The most common drawback of regular grid representations is that the polygonalization is never optimal [7]. Large, flat surfaces may require the same polygon density as small, rough areas do. This problem may be alleviated by reducing the overall complexity and applying temporal blending, or morphing, between different levels of detail [12]. Some visual simulation systems handle transitions between multiple levels of detail by "alpha blending" two models during the transition period. Ferguson [4] claims that such blending techniques between levels of detail may be visually distracting, and discusses a method of Delaunay triangulation which smoothly matches edges across areas of different resolution. However, this method is inherently difficult to implement.

Another approach based on maintaining of Real-time Optimally Adapting Meshes (ROAM) was introduced by Lindstrom [7], and utilized by DeBerg and Duchaineauy [2, 3]. The approach suggested in [7] is based on a hierarchical quadtree technique. In order to reduce the projected pixel error, the terrain is dynamically triangulated in a bottom up fashion according to the distance to the point of view. Since resolution is allowed to change smoothly, the result is a much better image quality. However, this algorithm still has a room for improvement. When the viewpoint is changing, the triangulation is continuously updating, resulting in a so-called "popping". As the observer approaches an area with detail information, this detail will suddenly appear at a certain distance. Another problem, crucial for ROAM algorithms, is controlling a smoothness and quality of the rendered image. While many simplification methods are mathematically viable, the level of detail generation and selection are often not directly coupled with the screen-space error resulting from the simplification.

To eradicate these problems, a number of approaches were suggested. A rapid geomorphing algorithm, performing top-down manipulations on the quadtree data structure, was presented in [11]. A recently proposed method of view-dependent refinement allows to build a mesh with a small number of triangles that for a given view is a good approximation of the original, dense mesh [8]. Novel methods for ROAM optimization, utilizing a variety of culling techniques, continue to evolve [1, 6, 8, 14]. As can be seen from the above discussion, efficient and easy algorithms for displaying complex geographical data are still in high demand, and this paper introduces one of them and discusses its performance on an example of real-time visualization of 3D Digital Elevation Models.

3 The Proposed Approach

Real-time Optimally Adapting Mesh (ROAM) method is selected as an extendable, efficient tool for internal data representation and dynamical updates of the terrain model. The method is extended with an original *implicit coordinates method* within the patch array a*nd the viewpoint dependent triangle rendering* method for dynamic level of detail (LOD) changes. The method is characterized by the following set of unique features:

- Smooth, continuous changes between different surface levels of detail
- Dynamic generation of levels of detail in real-time
- Introduction of implicit coordinates method within the patch array for more efficient ROAM representation
- Introduction of the viewpoint dependent triangle rendering method for dynamic level of detail (LOD) updates
- Implementation of culling techniques, including the original Relational Position culling for more efficient terrain rendering
- Flexibility in choosing/selecting various culling techniques
- Reduction in the amount of time required to achieve a given triangle count
- Application of error metrics for increased smoothness and continuity

The terrain data sets studied are the simple gradual contour changes and the complex steep contour changes, represented by greyscale Digital Elevation Maps of 1024 pixels by 1024 pixels. Each experimental set is internally represented by three quadrant detail levels, corresponding to 16, 64 and 128 nodes per side within the binary tree structure. The load time, the total number of triangles per path, the total number of culled triangles per path, the number of frames per second and the number of triangles per frame are examined for this structure. The occlusions culling techniques are individually and collectively combined with ROAM technique and examined with the different representations of detail levels to verify the algorithm feasibility and efficiency.

3.1 Digital Elevation Model (DEM)

Digital Elevation Model (DEM) can refer either to a specific elevation file format or to sources of elevation data in general. DEM data is usually stored as an array of regularly spaced elevation values, referenced horizontally either to a Universal Transverse Mercator (UTM) projection or to a geographic coordinate system. The grid cells are spaced at regular intervals along south to north profiles that are ordered from west to east. A standard grid posting is interpolated directly from the contour files to create DEMs with 10 - 90 meter (<1 - 3 arc second) resolution (depending on the source paper map scale or contour interval) [10]. Two greyscale Digital Elevation Model (DEM), representing progressive contours with gradual elevation changes (Fig. 1) and sudden steeper elevation changes (Fig. 2) were studied.

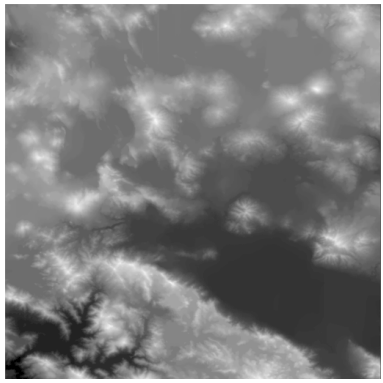


Figure 1. Gradual elevation changes changes

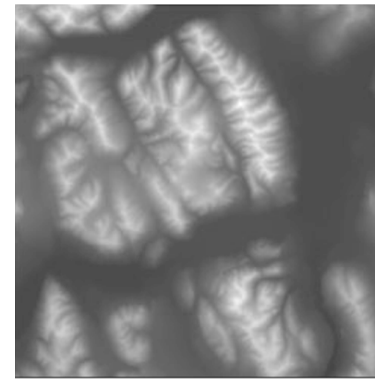


Figure 2. Steep elevation

4 Real-Time Terrain Rendering Algorithm

Our approach to terrain data rendering is based on the Real-time Optimally Adapting Mesh (ROAM) method (see [7]). The method presents a highly flexible and adaptable method for representing continuous mesh with controlled Level of Detail. It allows controlling the details of a mesh and to maximize the quality and minimize the number of triangle primitives used in the process. The ROAM-based system constructs a consistent and dynamic detail representations of terrain data by utilization of two main priority queues. They are driven by *split and merge* operations that adjust the terrain detail level dynamically. The split and merge functions are both built and changed from the data information held within a preprocessed Binary Triangle Tree data structure. We suggest to use the implicit coordinates method within an array of patch objects for memory conservation. In addition, we use a specific viewpoint dependent triangle procedure to reduce the total amount of computations needed to render the terrain data. The more detailed description of the algorithm is provided below.

The Real-time Terrain Rendering Algorithm

1. Preprocessing step.

1.1 Build a Binary Triangle Tree data structure to represent the geometric properties of the rendered terrain.

2. Dynamic rendering step.

2.1 Create two queues, the split queue and the merge queue, to keep the priorities for each individual triangle in the mesh triangulation;
2.2 Implement the split and merge operations for updating the triangulated mesh.
2.3 Perform recursive, incremental update to real-time optimally adapting mesh, using the implicit coordinates method within an array of patch objects;
2.4 Optimize processing using the viewpoint dependent triangle rendering method;
2.5 Perform as-needed updates for triangle strips affected by the culling changes;
2.6 Implement error metric control.

4.1 Split and Merge Function

As it was mentioned above, the ROAM algorithm is built around a Binary Tree structure that supplies triangle information for the split and merge functions [7]. Obtaining any level of triangulation from a sequence of splits or merges can be done from any other triangulation level. Two triangles that share the same base and are on the same detail level are referred to as a *diamond*. The split operation adds a new vertex at the diamond center resulting in the creation of four new right-isosceles triangles, which will increase the number of triangles representing a terrain area. As the number of triangles increase, the detail level that can be represented will also increase. The merge operation works inversely to the split operation.

In our implementation, the split and merge operations provide a flexible framework for making detailed updates to the triangulated mesh. The basic idea of each queue is to keep the priorities for each individual triangle in the mesh triangulation. The split operations would then start with the base triangulation level in the queue and then repeatedly split the triangle until the highest priority triangle is reached. The only requirement for the split priority queue is that the child's priority level must not be more than its parent's. The merge priority queue allows the merge operation to start from the previously rendered mesh triangulation. This allows a more consistent and quicker frame-to-frame coherence.

4.2 Implicit Coordinates Method

In our implementation, patches of triangles are used to create and manage the mesh approximation within the terrain's landscape. We introduce the *implicit coordinates method* within an array of patch objects to ensure more efficient memory usage. The method is based on the following idea. Instead of storing explicitly X, Y and Z coordinates for each vertex of a triangle, implicit coordinates, within the landscape, are stored for the isosceles right triangles that will be rendered onscreen. The advantage of this approach is that implicitly defining coordinates saves 36 bytes of RAM per triangle. An index within the patch array references an individual Binary Triangle Tree that in turn stores the references to each triangle level of detail for that patch. The size of the patch determines the relative size of each patch within the landscape. The patch objects are held within the Landscape object. The landscape object is built by combining each patch section until the entire terrain is rendered.

4.3 Viewpoint Dependent Triangle Rendering

A conventional method to reduce the amount of computations needed to render a complex scene is to apply Level of Detail (LOD) techniques. LOD method determine which sections of rendered mesh require less detail based on any number of criteria. In this paper, we extend the LOD technique with the *viewpoint dependent triangle rendering* method. The method allows more flexible information storage for dynamic and interactive first person view rendering. We allow portions of the terrain that are currently too far away to be rendered with few triangles, and the same sections of

terrain to be rendered with more triangles if the viewpoint moves closer. This is done by examining the field of view with the view frustum to determine which patch sections need more detail due to their proximity to the user.

4.4 Recursive Bintree Data Structure

To satisfy the LOD requirement a binary triangle tree structure will be used to hold the various levels of detail that is needed by the graphics-rendering engine. In the case of ROAM, a binary triangle tree structure, or a *bintree*, is a recursive structure where, at its lowest level, represents a right-isosceles triangle (see Fig. 3). In our implementation, each patch of terrain will have an individual bintree to define the triangle detail levels. The triangle bintree structure starts with the base terrain, either the least detail representation or the detail level from the previously rendered image, in the leaf components of the structure. The need for a change in detail level is determined by examining the corresponding error metrics.

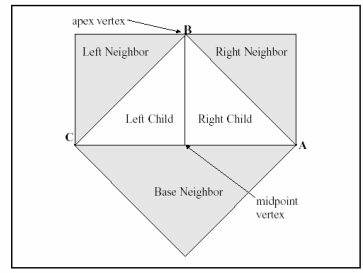


Figure 3. Binary Triangle Tree Structure

5 Improving Rendering Quality

Data culling is a process of selecting, from the whole scene, particular information that needs to be rendered. Culling at this level is often achieved by using geometry-based methods to determine which scene information needs to be rendered. We implement three types of geometric culling algorithms to improve smoothness and reduce "popping": View Frustum Culling, Backface Culling and an original Relational Culling technique (see Fig. 4). As far as we know, this is the first study that performs detailed comparison analysis of a variety of culling methods in application to different GIS data.

The *view frustum* is the volume of space that includes everything that is currently visible from a given viewpoint. Six planes arranged in the shape of a pyramid with the top removed, define the view frustum area. If a point or object is inside this volume then it is within the frustum area and is potentially visible. If a point is outside of the frustum then it is not visible to the user, it needs not be rendered. To determine the position of the points and object, their bounding volumes are computed. If the

bounding volume lies on one of the frustum edges then that bounding volume is further subdivided into smaller bounding volumes until each object is either determined to be inside or outside the frustum area. If at the lowest detail level an object still lies on a frustum edge, the portion of the object inside the frustum area is rendered while the rest is culled.

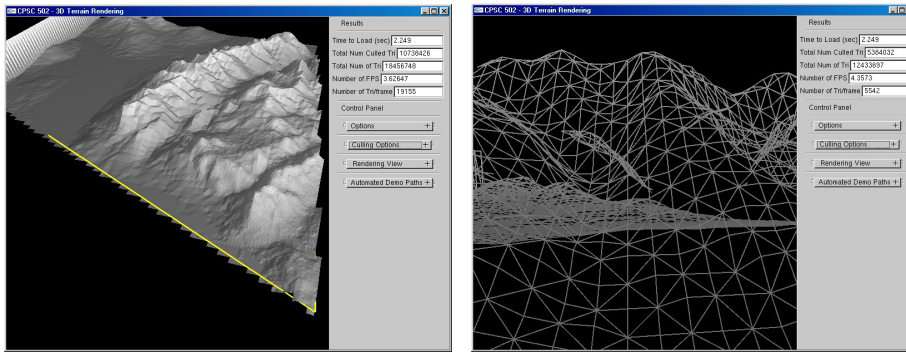


Figure 4. Sample Culling on Rendered Terrain (left), meshing (right).

Historically, the geometric shapes used as bounding volumes are boxes, or spheres, that are quicker to test and require less memory to store. They also require only four floating-point numbers for representation, while a bounding box requires eight. The disadvantage of a bounding sphere is sometimes it needs to be very large to surround an object that could be easy encompassed by a bounding box. To rectify this disadvantage, we link multiple bounding spheres in a chain to better simulate the object shapes.

We store the essential information described by the bounding spheres in the hierarchies of bounding volumes as a Direct Acyclic Graph (DAG). This structure will allow for quick and easy access of object information based on their relative positions. Modeling is done using OpenGL environment to extract the six planes of the current view frustum by retrieving the current PROJECTION and MODELVIEW matrices, combining the two, and then extracting the frustum values from the resulting matrix. Our algorithm results in four numbers that can represent the six planes. A point is within the view frustum if it is in front of all six planes simultaneously. To determine if a patch is within the view frustum a bounding box approach is utilized. The eight corners of the patch box are used to determine if the patch should be rendered or not rendered based on the same procedure as identifying if a point is with the view frustum.

The second method that we implement in this project is the *backface culling*. Based on a user's eye-space, back-facing polygons are located on the far side of an opaque object. These polygons, although part of the viewer's scene, are not visible to the viewer and do not need to be rendered. Once the polygons are determined to be back-facing, they can be culled before the scene is rendered. We calculate the normal of the projected polygon to determine if it is back-facing. This test involves calculating the polygon's normal and the vector formed from the viewing point to any point on the polygon.

The third technique that we introduce is the original *relational position culling* technique, based on pre-processing the terrain landscape into patches. Each patch would contain the Binary Triangle Tree structure of its terrain data and store each triangle's detail information within its node. Additionally, a visibility flag is stored to determine which patch is seen within the view frustum. This approach is developed to quickly cut the generalized unnecessary terrain data from the terrain data set. Initially, the algorithm determines the frustum triangle corners from a two-dimensional (2D) view frustum, which gives the algorithm the user and user's viewpoint's positions. These three points are used to determine the minimal rectangle that encompasses the 2D view frustum. Any points not within this rectangle are immediately culled. Advantages of this method are its simplicity and performance, that are discussed in the previous work [6] by the authors.

6 Algorithm Performance Analysis

The main contribution of this work is in the development of an efficient and adaptive real-time rendering algorithm based on ROAM technique, combined with a number of methods for increased rendering speed, smoothness and realism. The algorithms were implemented in Open GL.

When examining the results from the experiments, several relationships were observed. One of the most notable observations was the change of the number of frames per second (frame rate) during each of the paths corresponding step. Fig. 5 demonstrates the change in frame rate with all three culling techniques (View Frustum, Position based and Backface culling) enabled with three distinct patch sizes represented by their corresponding array size. The size-16 frame rate performs as expected with the graph trend line remaining rather flat and consistent throughout the entire experimental path, except for its initial load up stage, which is completed by the 25th frame. The size-64 frame rate demonstrates some interesting qualities. The first 100 frames emulates a similar pattern as the size-16 trend line with the exception of a consistently lower frame rate due to the increase of the number of patches that need to be rendered. The size-64 trend illustrates a significant increase in frame rate from frame 100 to 170. The increase corresponds to the first set of constant right and left hand turns in the experimental path that lasts until frame 161. Note that there are no significant differences in the number of frames rendered per second for different terrain models.

This set of turns can also be seen when examining Fig. 6 for both terrain models. The decrease in triangles per frame corresponds to the first left hand turn during a specific path over the terrain, most likely due to the quick pace of the turn and the difficultly of the rendering engine to propagate the necessary triangle detail levels before the next turn begins. As each turn is performed, the viewed landscape's true detail level is reduced. The number of triangles per frame continues to reduce as the turn progresses. This is the result of the view frustum largely shifting out of the frame of view. The trend line flattens as the forward movement allows the rendering engine time to increase the detail level of each frame, due to the limited changes in the view frustum.

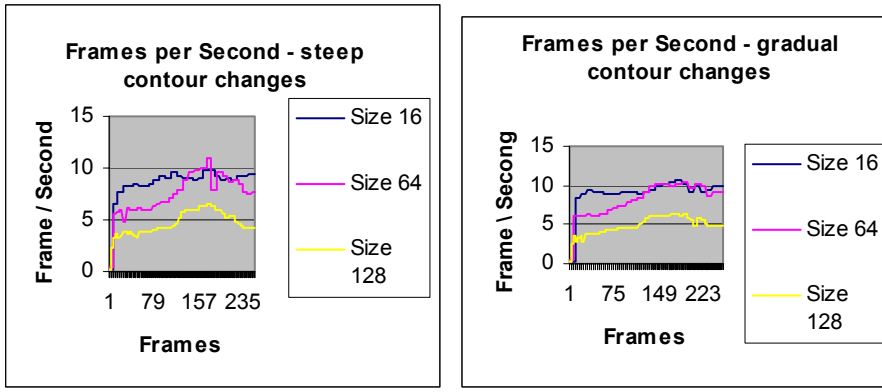


Figure 5. Number of frames per seconds for two models.

The rise of the size-64 trend line frame 180 is due to quick left and right turns which would leave the middle section of the view frustum untouched with only the frustum edges needing to be recomputed. The largest increase in the size – 64's trend line is due to the experimental path moving directly backwards (back stepping), which significantly decreases the amount of the view frustum that must be recomputed.

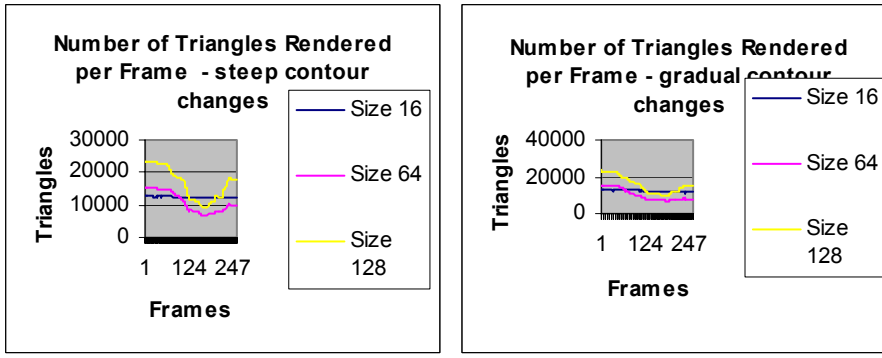


Figure 6. Number of triangles rendered per frame for two models.

Now, consider the time complexity of the algorithm. Analysis of time required for split and merge functions is quite interesting. It was anticipated that there would be an equal sharing of time between both the split and merge functions. That is, as one set of triangles needs to be split it would cause another set of triangles to congruently be merged adjusting to meet the overall set triangle detail level. When the application is examined in further detail, this is not always the case. Initially, as the triangles propagate through the landscape to their set detail level there is a relatively equal amount of time being spent in both algorithms. Interestingly, this is not the case when the culling techniques are active. In this case, a higher proportion of application time is spent in the split function and is directly related to the number of triangles being rendered per frame, Fig. 6, and the movement of the view frustum. When a

patch is culled from a the frustum, it has some detail level of rendered triangles. These triangles are eliminated without the use of the merge algorithm. This elimination of triangles allows the application to split new triangles on the new patches entering the frustum without increasing the overall triangle detail level. As the movement of the frustum increases with consecutive turns, this relationship becomes even more pronounced.

This observation helps to clarify the relationship between the number of triangles per frames and the movement of the frustum seen is Fig. 6. As the movement of the frustum becomes more continuous there is a reduction in the number of triangles being rendered per frame. This would be directly related to the disappearance of the rendered patches, with some triangle detail level, being eliminated the frustum view and the appearance of new patches, with lowest detail level, being brought into view. As the turn becomes more continuous, the greater the number of patches that is eliminated and the higher the demand on the split algorithm.

The time efficiency of the algorithm is also related to the size of the patch set. It also explains the algorithm behaviour for Size-16 and Size-64 patch sizes in Fig. 6. Since the Size-16 patches are larger, there is a greater area outside the view frustum being rendered. This wasted rendering area would act as a frustum buffer. Even though the patches would be eliminated at the same pace as a smaller patch size frustum, Size-64, these larger patches would have the smallest amount of detail on the portions of the patch that are outside the frustum site lines. Therefore the elimination of these patches will have less effect on the number of triangles being rendered per frame. This relationship also adjusts the time usage relationship between the split and merge algorithms. Due to the frustum buffer the merge function has time to adjust the triangle detail level of the patch that will be eliminated. This triangle merge reduction allows the split algorithm more time to readjust triangle detail level before the jolt of having the entire patch eliminated. As the patch size becomes smaller the adjustment time for the merge algorithm becomes less and the split algorithm more dominant.

7 Conclusions

The main contribution of this work is in the development of an efficient and adaptive real-time rendering algorithm based on ROAM technique, combined with a number of methods for increased rendering speed, smoothness and realism. Examining both the number of frames per second and the number of triangles per frame suggests a number of conclusions to be drawn. When investigating the experimental path's frame rates with all of the culling techniques active, we show that the patch sizes within the landscapes are significantly related to the change of frustum position. This correlation is confirmed by examining the number of triangle rendered per frame.

This paper also explored the time efficiency of the split and merge algorithms. The experimentation demonstrated that patch size has a correlation between the times being spent in each algorithm. Larger patch sizes effectively provide frustum buffer. As the patch size becomes smaller, the merge algorithm becomes less effective but the split function become more in demand due the rapid removal of patch with large detail levels. These results provide a unique inside view on the correlation between the different mechanisms, incorporated together in the presented algorithm for

realistic and efficient terrain representation and the real-time level of detail reduction. Further investigation of culling techniques and error metrics for improved visualization results for different types of rendered terrain models is planned.

The work was partially supported by GEOIDE and URGC Start-Up Grants.

8 References

[1] Blow, J. (2000) Terrain Rendering at High Levels of Detail, in Proceedings of the Game Developers' Conference, San Jose, California, USA.

[2] De Berg, M. and Dobrint, K. (1995) On levels of detail in terrains, Proc. 11th ACM Symposium on Computational Geometry.

[3] Duchaineauy, M, Wollinshy, M. (1997) ROAMing Terrain: Real-Time Optimally Adapting Meshes, IEEE Visualization '97 Proceeding.

[4] Ferguson, R., Economy, R., Kelly,W, and Ramos P. (1990) Continuous Terrain Level of Detail for Visual Simulation. IMAGE V Conference, pp. 144-151.

[5] Gross, M. H., Gatti, R., and Staadt, O. (1995) Fast Multiresolution Surface, Meshing. In Proceedings of Visualization ' 95, pp. 135-142.

[6] Hesse, M and Gavrilova, M. (2003) Quantitative Analysis of Culling Techniques for Real-time Rendering of Digital Elevation Models, WSCG 2003, in print.

[7] Lindstrom, P, and Koller, D. (1996) Real-time continuous level of detail rendering of height fields, Computer Graphics, SIGGRAPH 1996 Proc., pp. 109-118.

[8] Lindstrom, P and Pascucci, V. (2002) Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization, Proc. of IEEE Visualization and Computer Graphics, 8(3), pp. 239-254.

[9] Lloyd, B. and Egbert,P. (2002) Horizon Occlusion Culling for Real-time Rendering of Hierarchical Terrains, IEEE Visualization 2002, Boston, Massachusetts.

[10] National Mapping Division, U.S.Geological Survey, US GeoData Dig. Elev. Models.

[11] Rottger, S., Heidrich, W, Slusallek, P, Seidel H. (1998) Real-Time Generation of Continuous Levels of Detail for Height Fields, Graphische Datenverarbeitung.

[12] Taylor,D. C. and Barret, W. A. (1994) An Algorithm for Continuous Resolution, Polygonalizations of a Discrete Surface. In Proc. of Graphics Interface, 94, pp.33-42.

[13] VTP Virtual Terrain Project http://www.vterrain.org/index.html.

[14] Zhao, Y, Zhou, Y., Shi, J., Pan, Z. (2001) A Fast Algorithm For Large Scale Terrain Walkthrough, CAD/Graphics '2001, International Academic Publishers.

Geometric Graphs for Improving Nearest Neighbor Decision Rules

Godfried Toussaint*

School of Computer Science, McGill University
Montréal, Québec, Canada

1 The Power of Nearest-Neighbor Decision Rules

Non-parametric decision rules are powerful because of their simplicity, good performance, and the fact that no assumptions are made about the underlying distributions of the data (see Aha [1], Devroye, Gyorfy and Lugosi [6], Duda and Hart [7], Duda, Hart and Stork [8], McLachlan [10], O'Rourke and Toussaint [11]). In this setting we have available a set of d measurements (also called a feature vector) taken from each member of a data set of n objects (patterns) denoted by $\{X, Y\} = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$, where X_i and Y_i denote, respectively, the feature vector on the ith object and the class label of that object. One of the most attractive decision procedures, conceived by Fix and Hodges in 1951, is the nearest-neighbor rule (1-*NN*-rule) [9]. Let Z be a new pattern (feature vector) to be classified and let X_j be the feature vector in $\{X, Y\} = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$ closest to Z. The nearest neighbor decision rule classifies the pattern Z into class Y_j, breaking ties randomly.

A key feature of this decision rule (also called *lazy learning* [1], *instance-based learning* [2], and *memory-based reasoning* [13]) is that it performs remarkably well considering that no explicit knowledge of the underlying distributions of the data is used. Consider for example the two class problem and denote the *a priori* probabilities of the two classes by $P(C_1)$ and $P(C_2)$, the *a posteriori* probabilities by $P(C_1|X)$ and $P(C_2|X)$, and the mixture probability density function by

$$p(X) = P(C_1)p(X|C_1) + P(C_2)p(X|C_2),$$

where $p(X|C_i)$ is the class-conditional probability density function given class $C_i, i = 1, 2$. In 1967 Cover and Hart [3] showed, under some continuity assumptions on the underlying distributions, that the asymptotic error rate of the 1-*NN* rule, denoted by P_e[1-*NN*] is given by

$$P_e[1 - NN] = 2\mathbf{E}_X[P(C_1|X)P(C_2|X)],$$

where $\mathbf{E}_X$ denotes the expected value with respect to the mixture density $p(X)$. They also showed that P_e[1-*NN*] is bounded from above by twice the Bayes error

* This research was supported by NSERC and FCAR.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 762–765, 2003.
© Springer-Verlag Berlin Heidelberg 2003

(the best possible rule). More precisely, for M pattern classes the bounds proved by Cover and Hart [3] are given by:

$$P_e \leq P_e[1 - NN] \leq P_e(2 - MP_e/(M-1)),$$

where P_e is the optimal Bayes probability of error given by:

$$P_e = 1 - \mathbf{E}_X[\max\{P(C_1|X), P(C_2|X), ..., P(C_M|X)\}]. \tag{1}$$

and

$$P_e[1 - NN] = 1 - \mathbf{E}_X[\sum_{i=1}^{M} P(C_i|X)^2]. \tag{2}$$

Devroye [5] generalized these results by proving the bounds for all distributions. In other words, the nearest neighbor of Z contains at least half of the total discrimination information contained in an infinite-size training set. Furthermore, a simple generalization of this rule called the *k-NN*-rule, in which a new pattern Z is classified into the class with the most members present among the k nearest neighbors of Z in $\{X, Y\}$, can be used to obtain good estimates of the Bayes error, and its probability of error asymptotically approaches the Bayes error (Devroye et al. [6]). Even more noteworthy is the fact that with a simple editing (preprocessing) procedure the asymptotic error of the 1-*NN* rule can be made to equal the optimal Bayes error. In 1972 Wilson [15] first conceived the idea of editing with this goal in mind, and proposed the following elegant and simple algorithm. (A gap in the proof of Wilson [15] was pointed out by Devijver and Kittler [4] but alternate proofs were provided by Wagner [14] and Penrod and Wagner [12].)

PREPROCESSING

A. For each i:

1. Find the k-nearest neighbors to X_i among $\{X, Y\}$ (not counting X_i).
2. Classify X_i to the class associated with the largest number of points among the k-nearest neighbors (breaking ties randomly).

B. Edit $\{X, Y\}$ by deleting all the points misclassified in the fore going.

DECISION RULE

Classify a new unknown pattern Z using the 1-*NN* rule with the *edited* subset of $\{X, Y\}$.

For infinite size training sets the *edited* 1-*NN* rule in principle provides everything we want. In practice, however, we have a finite size training set and finding a counterpart to the *edited* 1-*NN* rule is much more difficult. In addition, computational complexity and storage requirements as well as a host of related questions become relevant. Such questions include the following. How can the storage of the training set be reduced without degrading the performance of the decision rule? How should the reduced training set be selected to represent the

different classes? How large should k be? How should the value of k be chosen? Should all k neighbors be equally weighted when used to decide the class of an unknown pattern? If not, how should the weights be chosen? Should all the features (attributes) we weighted equally and if not how should the feature weights be chosen? What distance metric should be used? How can the rule be made robust to overlapping classes or noise present in the training data? How can the rule be made invariant to scaling of the measurements? How can the nearest neighbors of a new point be computed efficiently? What is the smallest neural network that can implement nearest neighbor decision rules? Geometric graphs such as Voronoi diagrams and their relatives provide elegant solutions to most of these problems.

In this talk I will discuss how several geometric graphs solve such problems in an elegant way, as well as the resulting computational geometry questions that they suggest.

References

[1] D. W. Aha, editor. *Lazy Learning*. Kluwer, Norwell, MA, 1997.

[2] D. W. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. In *Machine Learning, 6*, pages 37–66. Kluwer Academic Publishers, Boston, Mass., 1991.

[3] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[4] Pierre A. Devijver and Josef Kittler. On the edited nearest neighbor rule. In *Fifth International Conference on Pattern Recognition*, pages 72–80, Miami, December 1980.

[5] Luc Devroye. On the inequality of Cover and Hart. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:75–78, 1981.

[6] Luc Devroye, László Gyorfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag New York, Inc., 1996.

[7] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, 1973.

[8] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., New York, 2001.

[9] E. Fix and J. Hodges. Discriminatory analysis. Nonparametric discrimination: Consistency properties. Tech. Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[10] Geoffrey J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley and Sons, Inc., New York, 1992.

[11] Joseph O'Rourke and Godfried T. Toussaint. Pattern recognition. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 43, pages 797–814. CRC Press LLC, Boca Raton, FL, 1997.

[12] C. S. Penrod and T. J. Wagner. Another look at the edited nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 7:92–94, 1977.

[13] C. Stanfill and D. L. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, December 1986.

[14] Terry J. Wagner. Convergence of the edited nearest neighbor. *IEEE Transactions on Information Theory*, 19:696–697, September 1973.

[15] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2:408–420, 1972.

Red-Blue Separability Problems in 3D*

Ferran Hurtado[1], Carlos Seara[1], and Saurabh Sethia[2]

[1] Dpt. Matematica Aplicada 2, Univ. Politécnica de Catalunya,
08028 Barcelona, Spain.
hurtado|seara@ma2.upc.es. http://www-ma2.upc.es/∼hurtado
[2] Dpt. of Computer Science, Oregon State University,
Corvallis, OR 97331-3202, USA.
saurabh@cs.orst.edu. http://www.cs.orst.edu/∼saurabh

Abstract. In this paper we study the problems of separability of two disjoint point sets in 3D by multiple criteria extending some notions on separability of two disjoint point sets in the plane.

1 Introduction

Let B and R be two disjoint sets of points in 3D classified as *blue* and *red* points, respectively. Let n be the number of points in B and R. We consider the sets of points in general position, thus there are no four points in a plane and no three points on a line.

Let $\mathcal{C}$ be a family of surfaces in 3D. The sets B and R are $\mathcal{C}$ *separable* if there exists a surface $S \in \mathcal{C}$ such that every connected component of $\mathbb{R}^3 - S$ contains points only from B or from R. If S is a plane, we have *linear separability*. The decision problem of linear separability for two disjoint sets of objects (points, segments, polygons or circles) in 2D or (points, segments, polyhedra or spheres) in 3D can be solved in linear time [6,10].

In [1,7,8] the authors study the separability of two disjoint point sets in the plane by the following criteria: *wedge separability*, *strip separability* and *double wedge separability*. A *wedge* is the union of two rays with common origin (the *apex*). A *strip* is the union of two parallel lines; their common slope is the *slope of the strip*. A *double wedge* is the union of two crossing lines, their intersection point is the *apex*. Optimal $\Theta(n \log n)$ time algorithms for deciding wedge, strip, and double wedge separability, as well as for constructing the locus of feasible apices of wedges and double wedges, and the interval of feasible slopes of strips are described in [1,7,8]. The locus of feasible apices and the interval of feasible slopes are partial descriptions of all separating wedges, double wedges, and strips. They have also shown how to find wedges and double wedges with maximum and minimum aperture angle, and the narrowest and the widest strips.

* Partially supported by Proyectos DGES-MEC PB98-0933, UPC 1999SGR000356. Partially supported by Joint Commission USA-Spain for Scientific and Technological Cooperation Project 98191.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 766–775, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In this paper we study the separability of two disjoint point sets in 3D by extending the criteria above and giving solutions to various separability problems. While the separability criteria are extensions of their 2D counterpart, the techniques used in 3D do not extend from those used for 2D. For each separability criterion we consider the problem of deciding whether that particular separability is feasible, which is probably equivalent to finding one solution to the problem; in some cases, we also consider the problem of finding partial descriptions of all feasible solutions. For some separability criteria we consider, the convex hull of either the red or blue points needs to be monochromatic. If it is so, we perform this check as a first step in our algorithms. It can be done in $O(n \log n)$ time by computing the convex hulls of the point sets in 3D [11]. In all our algorithms this time is dominated by other computations.

The paper is organized as follows. We consider the problem of computing all the feasible solutions for linear separability in section 2. We study slice, wedge, and diwedge separability in section 3 as natural extensions in 3D for strip, wedge, and double wedge separability in 2D. We study the decision problem for prismatic, pyramidal, and dipyramidal separability, which also can be considered as extensions in 3D for strip, wedge, and double wedge separability criteria, but allowing a linear number of planes in section 4. Finally, we study some separability criteria defined by a constant number of planes in section 5.

2 Linear Separability

The decision problem of linear separability for two disjoint point sets in 3D can be solved in linear time [10]. We look at the problem of computing the locus of all the feasible solutions in some detail. We do the following.

1. In linear time compute a plane separating the two sets of points. Assume that this plane is the $y = 0$ plane such that the red points are above the plane and the blue points are below the plane.
2. In $O(n \log n)$ time compute the convex hulls of the two point sets, $CH(R)$ and $CH(B)$.
3. The boundary of the locus of all the separating planes of R and B is defined by two unbounded convex polygonal regions with $O(n)$ total complexity, which are formed by some unbounded faces (defined by the extreme separating planes between $CH(R)$ and $CH(B)$) and some faces of $CH(R)$ and $CH(B)$. This boundary can be computed in $O(n)$ time using a projective transformation given by Davis in [4]. We omit the details of this computation from this abstract.

Once we have computed the boundary of the locus then, we can decide in $O(\log n)$ time whether a given plane is a separating plane, by just checking whether the given plane intersects any of the two unbounded convex polygonal regions.

The problem of computing all the feasible solutions has an $\Omega(n \log n)$ time lower bound due to the time lower bound for the 2D version of this problem. Thus, we have the following theorem.

Theorem 1 *Let B and R be two disjoint point sets in $\mathbb{R}^3$. The locus of all the separating planes can be computed in $\Theta(n \log n)$ time. Once we have precomputed the locus, to decide whether a given plane separates the point sets can be done in $O(\log n)$ time.*

3 Separability by Two Planes

In this section we study natural extensions for strip, wedge, and double wedge separability criteria in 2D, which involves exactly two separating planes. More concretely, we study slice, wedge, and diwedge separability as extension for strip, wedge, and double wedge separability, respectively.

3.1 Slice Separability

A slice is defined as the space between two parallel planes (Figure 1). The normal vector to the planes of the slice is called the *slice direction*. Slice separability can be considered as a natural extension for strip separability in the plane. The slice separability problem asks: Is there a slice that contains all the red points but does not contain any blue point or vice versa?

Before giving a solution to the problem, we address the following question. Suppose that there exists a slice separating the red points from the blue points and this slice is given by two horizontal parallel planes. Let B_1 and B_2 be the sets of blue points above and below the slice, respectively. The sets B_1 and B_2 form a partition of B. The question now is, how many different partitions of B corresponding to slice separators can exists? In the full paper we show that the number of partitions is $\Theta(n^2)$ in the worst case.

Theorem 2 *Let B and R be two disjoint point sets in $\mathbb{R}^3$. Deciding whether the sets are slice separable can be done in $O(n^3)$ time. The locus of slice directions of all feasible solutions can be computed in $O(n^3 \log n)$ time. The locus can have $\Theta(n^2)$ connected components with total complexity $\Theta(n^2)$.*

Proof. Assume that there exists a separating slice which contains R. Let u be its *slice direction*. Let B_1 (B_2) be the set of blue points above (below) the slice. Then a plane π_1 with normal vector u passing through any red point r is a separating plane of B_1 and B_2. While keeping the point r on the plane, we can move u around with two degrees of freedom until it bumps into two blue points b_1 and b_2. There are three exclusive possibilities: both points belong to B_1, both belong to B_2, or one belongs to B_1 and the other one belongs to B_2. Therefore, if there exists a separating slice then, it can be found by the following algorithm.

1. Choose a red point $r \in R$.
2. For each pair of blue points b_1 and b_2, compute the plane π_1 passing through b_1, b_2, and r.
3. Considering the three possibilities above, compute B_1 (B_2), the set of blue points above (below) π_1.

4. By linear programming compute a separating slice (if it exists) defined by two parallel planes such that one separates B_1 from $R \cup B_2$ and, the other one separates $R \cup B_1$ from B_2.

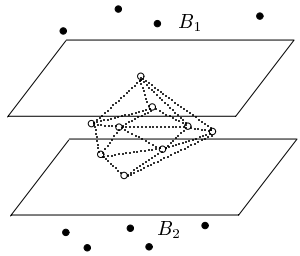


Fig. 1. Slice separability.

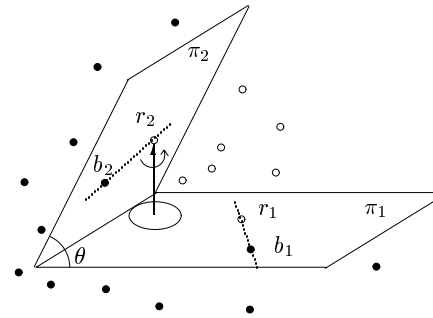


Fig. 2. Wedge separability.

This gives an $O(n^3)$ time algorithm for deciding whether there exists a separating slice, because the algorithm computes all the possible partitions for slice separability. In order to compute the set of slice directions of all the feasible solutions we proceed as follows. Once we have found a separating slice partitioning B into B_1, B_2; we do the following.

4.1. Compute $CH(B_1)$, $CH(B_2)$, $CH(R \cup B_1)$, and $CH(R \cup B_2)$.
4.2. Compute the region on the unit sphere formed by the set of directions of the separating planes between $CH(B_1)$ and $CH(R \cup B_2)$. Proceed analogously for computing the set of directions of the separating planes between $CH(R \cup B_1)$ and $CH(B_2)$. We obtain two regions with $O(n)$ complexity each bounded by at most two convex chains. The projection of each region on the plane $z = 1$ is either a convex polygon or two unbounded convex polygons. Thus there are at most four convex polygonal chains.
4.3. Compute the (non empty) intersection of the two regions in $O(n)$ time. Its boundary corresponds to parallel planes which are always touching $CH(R)$ and some blue point.

Thus, in additional $O(n \log n)$ time, we can compute the convex region on the unit sphere defined by the set of slice directions of all the feasible solutions for each good partition. The total time complexity of the algorithm is $O(n^3 \log n)$.

In the full paper we show that the locus of slice directions can have $\Theta(n^2)$ connected components with total complexity $\Theta(n^2)$.

The Narrowest and the Widest Separating Slices The width of a separating slice is defined by the distance between the two parallel planes. If the sets are slice separable, it is natural to ask about the narrowest and the widest separating slices. We have obtained the following results.

Theorem 3 *Let B and R be two disjoint point sets in $\mathbb{R}^3$. The widest separating slice can be computed in $O(n^3)$ time, if we have precomputed the set of slice directions of the separating slices.*

Theorem 4 *Let B and R be two disjoint point sets in $\mathbb{R}^3$. The narrowest separating slice can be computed in $O(n^2 \log n)$ time, if we have precomputed the set of slice directions of the separating slices.*

The running time of our algorithms to compute the narrowest and widest separating slice is $O(n^3 \log n)$ due to the precomputation of the set of slice directions of the separating slices. The difference in complexities after this precomputation is simply due to the use of different techniques. The optimal complexities may be much lower than our bounds and hence the difference in complexities of our algorithms need not reflect the comparative difficulty of the two problems.

3.2 Wedge Separability

A natural extension for wedge separability in the plane is separability by a 3D wedge which is defined as follows. Two intersecting planes divide the 3D space into four quadrants. Any one quadrant is called a wedge. The wedge separability problem is the following: Is there a wedge that contains R but does not contain any blue point or vice versa?

Theorem 5 *Let B and R be two point sets in $\mathbb{R}^3$. Deciding whether the sets are wedge separable can be done in $O(n^3)$ time.*

Proof. Consider a separating wedge with a fixed aperture angle θ. We can move the two planes parallel to themselves until they touch a red point of $CH(R)$. Let π_1, π_2 be the two planes and r_1, r_2 the respectively touched red points. Project r_2 on π_1 and construct the circle shown in Figure 2. Now rotate plane π_2 around the circle keeping the touching point r_2 on it (in this way we maintain the aperture angle θ) until π_2 bumps into a new (red or blue) point. Now do the same with the other plane. Thus we have two points on each of the planes of the wedge (Figure 2). The two planes can now be rotated simultaneously about the lines passing through their respective two points such that the aperture angle between them remains θ. We can do this until one of the planes bumps into a new red or blue point. Thus at least one of the planes will have three points on it of which at least one would be red.

Note that there is still one degree of freedom as we rotate the other plane around the line defined by the two points until it bumps into a new point; but doing this, the value θ will be increased or decreased and then, the two planes of the wedge will have three points on it of which at least one will be from $CH(R)$. This observation leads to the following algorithm.

1. For each pair of points compute the line l passing through them.
2. Assuming a hierarchical representation of $CH(R)$, compute the at most two supporting planes between l and $CH(R)$ in $O(\log n)$ time. For each of these planes do the following.

2.1. Let the plane under consideration be the plane π_1 of the wedge. Assume that π_1 is the plane $z = 0$ and such that $CH(R)$ is above π_1. In $O(n)$ time compute the set B_1 of blue points above plane π_1.
2.2. In $O(n)$ time compute (if it exists) a separating plane between B_1 and $CH(R)$. It will be the second plane of the wedge.

This gives an $O(n^3)$ time algorithm for deciding whether there exists a separating wedge.

Wedges with Maximum and Minimum Aperture Angle Suppose that the point sets are wedge separable but not line or slice separable. It is natural to ask about the separating wedges with maximum and minimum aperture angle. We have obtained the following result.

Theorem 6 *Let B and R be two point sets in $\mathbb{R}^3$. Computing the separating wedges with maximum and minimum aperture angle can be done in $O(n^3 \log n)$ time.*

Wedges with Fixed Aperture Angle A natural problem is to decide whether there exists a separating wedge with a fixed aperture angle θ, $0^\circ \leq \theta \leq 180^\circ$. Note that if we have precomputed the separating wedges with maximum and minimum aperture angle, the problem is not solved because it may be that B and R are not wedge separable for all possible values of aperture angle between the minimum and the maximum. We have obtained the following result.

Theorem 7 *Let B and R be two point sets in $\mathbb{R}^3$. Computing a separating wedge with fixed aperture angle can be done in $O(n^3 \log n)$ time.*

3.3 Diwedge Separability

A natural extension for double wedge separability in the plane is separability by a 3D *diwedge* which is defined as follows. Two intersecting planes divide the 3D space into four quadrants. The union of a pair of opposite quadrants is called a *diwedge*. The diwedge separability problem is the following: Is there a diwedge that contains R but does not contain any point from B or vice versa?

We define the *aperture angle* of a diwedge to be the bigger of the two aperture angles defined by the planes of the diwedge. Our results and proofs for diwedge separability are similar to wedge separability except that the asymptotic running times of all algorithms is higher by a factor of n. Hence to conserve space, we state our results on diwedge separability without proofs.

Theorem 8 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding whether the sets are diwedge separable can be done in $O(n^4)$ time.*

Theorem 9 *Let B and R be two point sets in $\mathbb{R}^3$. Computing a separating diwedge with maximum or minimum aperture angle can be done in $O(n^4 \log n)$ time.*

Theorem 10 *Let B and R be two point sets in $\mathbb{R}^3$. Computing a separating diwedge with fixed aperture angle can be done in $O(n^4 \log n)$ time.*

4 Separability by a Linear Number of Planes

In this section we study other separability criteria which can be consider also as extensions for the slice, wedge and double wedge separability in the plane, but allowing a linear number of planes. More precisely, we extend the concepts above to prismatic, pyramidal and dipyramidal separability, respectively.

4.1 Prismatic Separability

A prism is defined as the space swept by a convex polygon when it is moved along a line perpendicular to its plane; the direction of this line is called the *direction of the prism*. The prismatic separability problem asks: Is there an infinite prism which contains all red points but none of blue points or vice versa?

Theorem 11 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding prismatic separability and computing the locus of the set of directions of all the separating prisms can be done in $O(n^3)$ time. This locus is formed by at most $O(n^2)$ connected components with $O(n^2\alpha(n))$ total complexity.*

Proof. A solution for prismatic separability can be described by the direction normal to the base of the prism, the *direction of the prism*. The presence of a blue point rules out a certain set of direction vectors. This set is given by the dipyramid (two symmetrical pyramids having a common apex) whose apex is the blue point and whose planes are tangent to the red convex hull (Figure 3).

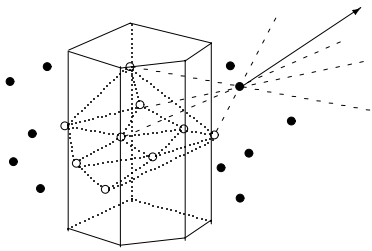


Fig. 3. Prismatic separability.

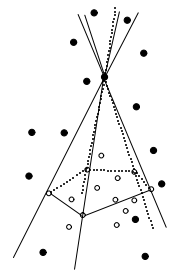


Fig. 4. Pyramidal separability.

Each pyramid of this dipyramid represents a set of directions which cannot be the solution, otherwise the point will be contained in the prism. On the unit sphere these directions occupy two convex regions. We get two such convex regions for every blue point. The complement of the union of all these regions is the set of direction vectors that correspond to a solution. To simplify computation of this set we can project the direction vectors on the unit sphere to two planes at $z = 1$ and $z = -1$. The convex regions are now (possibly unbounded) convex polygons. There are $O(n)$ (possibly unbounded) convex polygons, each of size $O(n)$.

These polygons can be easily computed in $O(n)$ time each, taking a total of $O(n^2)$ time. The complexity of this algorithm would be dominated by the time to compute the complement of the union of these polygons. One way to compute the complement of the union of these polygons is by computing their arrangement. One way to compute the arrangement is by using Chazelle and Edelsbrunner's algorithm [3] with $O(N \log N + K)$ running time, where N is the total number of segments and K is the size of the arrangement. In our case N is $O(n^2)$ and K is $O(n^3)$. Thus we can compute the arrangement and hence the union in $O(n^3)$ time. The sets B and R are prismatic separable if, and only if, the complement of the union is not empty.

In the full paper we prove that the locus of the set of directions of separating prisms is formed by at most $O(n^2)$ connected components with total complexity $\Theta(n^2\alpha(n))$.

4.2 Pyramidal Separability

Join all the vertices of a convex polygon to a point in space to get a pyramid. The convex polygon is called the base of the pyramid while the point in space is called its apex. An infinite pyramid is one whose base is at infinity. The pyramidal separability problem asks the question: Is there an infinite pyramid that contains all the red points but none of the blue points or vice versa?

Theorem 12 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding pyramidal separability and computing the locus of apices of all the separating pyramids can be done with a randomized algorithm whose expected running time is $O(n^3 \log^2 n)$. The locus of apices of separating pyramids is formed by $O(n^3)$ connected components with $O(n^3 \log n)$ total complexity.*

Proof. A solution to pyramidal separability can be stated by the apex point of the pyramid. The presence of a blue point rules out a certain region in 3D where the solution cannot be there. Consider the dipyramid subtended by the blue point on the red convex hull (Figure 4). The pyramid of this dipyramid that does not contain the red convex hull cannot contain the solution. Also if we consider the blue point as a light source, the shadow of the red convex hull cannot contain the solution. The complement of the union of these ruled out unbounded convex polyhedra for all the blue points gives the set of apices of all feasible solutions. There are $O(n)$ 3D unbounded convex polyhedra each of

complexity $O(n)$. The complexity of computing the union of these unbounded convex polyhedra would dominate.

One way to compute the union is by using the following results from Aronov, Sharir and Tagansky [2]. They show that the number of vertices, edges and faces of the union of K convex polyhedra in three dimensional space, having a total of N faces is $O(K^3 + NK \log K)$. This bound is almost tight in the worst case, as there exists collections of polyhedra with $\Omega(K^3 + KN\alpha(K))$ union complexity. They also describe a rather simple randomized incremental algorithm for computing the boundary of the union in $O(K^3 + KN \log K \log N)$ expected time. In our case, K is $O(n)$ and N is $O(n^2)$; therefore the complexity of the union of the unbounded convex polyhedra above is $O(n^3 + n^3 \log n) = O(n^3 \log n)$ and, the boundary of the union can be computed in $O(n^3 \log^2 n)$ expected time.

Now we consider the question of knowing how many different regions (connected components) corresponding to apices of separating pyramids can exist. A result by Katona [9] says that *the complement of the union of K convex polyhedra in d-space has at most $O(K^d)$ connected components*, which in our case say that the number of regions of apices of separating pyramids is at most $O(n^3)$.

We also have a deterministic result for pyramidal separability which we state without proof.

Theorem 13 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding whether the sets are pyramidal separable can be done in $O(n^7)$ time.*

4.3 Dipyramidal Separability

Dipyramidal separability has the same definition as pyramidal separability except that now we have two symmetrical pyramids having the same apex. We have obtained the following result.

Theorem 14 *Let B and R be two point sets in $\mathbb{R}^3$. Deciding whether the sets are dipyramidal separable can be done in $O(n^8 \log n)$ time.*

5 Separability by a Constant Number of Planes

In this section we study some particular separability criteria which involve three to six planes. More precisely, we consider triangular prism, tetrahedral, triplane, and box separability. Due to space constraints we state our results without proof.

The triangular prismatic separability problem asks the question: Is there an infinite triangular prism which contains all the red points but none of the blue or vice versa?

Theorem 15 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding whether the sets are triangular prismatic separable can be done in $O(n^5)$ time.*

The tetrahedral separability problem asks the question: Is there a tetrahedron that contains all the red points but none of the blue points or vice versa?

Theorem 16 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding whether the sets are tetrahedral separable can be done in $O(n^7)$ time.*

Three intersecting planes divide the 3D space into eight octants. The triplane separability problem asks the question: Are there three planes such that each of the octants they define has points of only one color?

Theorem 17 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding whether the sets are triplane separable can be done in $O(n^7)$ time.*

The box separability problem asks the question: Is there an orthogonal box (not necessarily axis aligned) which contains all the red points but none of the blue or vice versa?

Theorem 18 *Let B and R be two sets of points in $\mathbb{R}^3$. Deciding whether the sets are box separable can be done in $O(n^7)$ time.*

6 Conclusion

There are many more interesting questions on separability in 3D as compared to 2D. We have shown algorithms for some of them. Other possible extensions for slice, wedge and diwedge separability are the cylindrical, conical and diconical separability.

References

1. E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, S. S. Skiena, *Some Lower Bounds on Geometric Separability Problems*, 11th Fall Workshop on Computational Geometry, 2001.
2. B. Aronov, M. Sharir, B. Tagansky *The Union of Convex Polyhedral in Three Dimensions*, SIAM J. Comput., Vol. 26, No. 6, 1997, pp. 1670–1688.
3. B. Chazelle, H. Edelsbrunner, *An Optimal Algorithm for Intersecting Line Segments*, Journal of ACM, 39, 1992, pp. 1–54.
4. G. Davis, *Computing Separating Planes for a Pair of Disjoint Polytopes*, Proceedings of the first Annual Symposium on Computational Geometry, 1985, pp. 8–14.
5. H. Edelsbrunner, F. P. Preparata, *Minimum Polygonal Separation*, Information and Computation, 77, 1988, pp. 218–232.
6. M. E. Houle, *Algorithms for Weak and Wide Separation of Sets*, Discrete Applied Mathematics, Vol. 45, 1993, pp. 139–159.
7. F. Hurtado, M. Noy, P. A. Ramos, C. Seara, *Separating Objects in the Plane with Wedges and Strips*, Discrete Applied Mathematics, Vol. 109, 2001, pp. 109–138.
8. F. Hurtado, M. Mora, P. A. Ramos, C. Seara, *Two Problems on Separability with Lines and Polygonal*, 15th European Workshop on Computational Geometry, 1999, pp. 33–35.
9. G. O. Katona, *On a Problem of L. Fejes Tth*, Stud. Sci. Math. Hung., 12, 1977, pp. 77–80.
10. N. Megiddo, *Linear-time Algorithms for Linear Programming in $\mathbb{R}^3$ and Related Problems*, SIAM J. Computer, Vol. 12, No. 4, 1983, pp. 759–776.
11. F. P. Preparata, S. J. Hong, *Convex Hulls of Finite Sets of Points in Two and Three Dimensions*, Communications of the ACM, 20, 1977, pp. 87–93.

Two-Dimensional Range Search Based on the Voronoi Diagram

Takeshi Kanda and Kokichi Sugihara

Department of Mathematical Informatics,
Graduate School of Information Science and Technology, University of Tokyo,
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8685, Japan,
chp95260@par.odn.ne.jp, sugihara@mist.i.j-tokyo.ac.jp

Abstract. This paper studies the two-dimensional range search problem, and constructs a simple and efficient algorithm based on the Voronoi diagram. In this problem, a set of points and a query range are given, and we want to enumerate all the points which are inside the query range as quickly as possible. In most of the previous researches on this problem, the shape of the query range is restricted to particular ones such as circles, rectangles and triangles, and the improvement on the worst-case performance has been pursued. On the other hand, the algorithm proposed in this paper is designed for a general shape of the query range, and is intended to accomplish a good average-case performance. This performance is actually observed by numerical experiments. We can observe that our algorithm shows the better performance in almost all the cases.

1 Introduction

Geometric search is one of the fundamental and important problems in computational geometry [4] [7] [11]. The most basic type of the geometric search is the nearest point search, in which the point nearest to a query point is searched for in a given set of points [14]. The problem can be solved by the Voronoi diagram for a low dimensional space [10], and by various types of tree data structures for a higher dimensional space [13]. Another, a little more complicated type of geometric search is the range search. In this problem, a set S of points and a query range R are given, and we want to enumerate all the points in S that are inside R [8] [14]. This probram has many applications such as hidden surface removal and ray tracing in computer graphics, geographic information systems and data mining In many applications, the set S of points is fixed, and the different query ranges are given many times. Hence, we are interested in the preprocessing for S in order to decrease the search time for each of query range.

Most of the previous researches on this problem are restricted to the case where the shape of a query range R is orthogonal [1], half-planar [6], circular [3] [5] or spherical [2] [12]. In this paper, on the other hand, we propose an algorithm for solving the two-dimensional range search problem for general shapes of a query range R, by means of a simple technique using the Voronoi diagram.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 776–786, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Furthermore, a good average-case performance is observed by numerical experiments.

In Section 2, we define the range search problem. In Section 3, we give an efficient algorithm for this problem, which is based on the Voronoi diagram. In Section 4, we experimentally show the performance of this algorithm. Lastly, conclusions are given in Section 5.

2 Range Search Problem

The **range search problem** in the two-dimensional space is defined as follows (See Figure 1):

> Given a set S of N points $\{P_n \mid n = 1,\ 2,\ \ldots,\ N\}$, preprocess the point set S, so that we can answer the query "Enumerate all the points in the set S inside the query range R?" quickly.

In many practical cases, the point set S is fixed once it is given, and a huge number of different query ranges are given one by one. A naive method of solving this problem is to check whether each point in S is inside R or not; however, this method requires as much as $\mathrm{O}(N)$ time for each query. So, we are interested in preprocessing S which can decrease the search time.

3 Range Search Based on the Voronoi Diagram

In this section, we give an algorithm of the two-dimensional range search using the Voronoi diagram. In Subsection 3.1, we review the Voronoi diagram. In Subsection 3.2, we consider the side-trip facility search, the algorithm of which forms the first step of that of the range search described in the following subsection. In Subsection 3.3, we construct the algorithm of the range search using the Voronoi diagram, and lastly, we demonstrate the validity of this algorithm in Subsection 3.4.

3.1 Voronoi Diagram

Given a set of N distinct points $\{P_i(x_i, y_i) \mid i = 1,\ 2,\ \ldots,\ N\}$ in the plane, we associate all locations in the plane with the closest member of the point set. The result is a tessellation of the plane into a set of regions associated with the members of the point set. This tessellation is called the **planar ordinary Voronoi diagram** generated by the point set [10]. The members of the point set are called **generators**. The region associated with a member P_i is denoted by $\mathcal{V}(P_i)$ and is called the **Voronoi polygon associated with** P_i. Vertices and edges in the Voronoi diagram are called **Voronoi vertices** and **Voronoi edges** respectively.

3.2 Side-Trip Facility Search Problem

Before considering the algorithm solving the range search problem itself, let us start with considering the **side-trip facility search problem**. For points $Q_0, Q_1, \ldots, Q_M$ in the plane, let us denote by $\overline{Q_0Q_1 \ldots Q_M}$ the polygonal line consisting of M line segments $\overline{Q_0Q_1}$, $\overline{Q_1Q_2}$, ..., $\overline{Q_{M-1}Q_M}$. Here we consider the next problem (See Figure 2):

> Given a set S of N points $\{P_n \mid n = 1,\ 2,\ \ldots,\ N\}$, preprocess the point set S properly and answer one of the following queries quickly:
> **Type 1**: "Is there a point in the set S which is within the distance δ from the query polygonal line $\overline{Q_0Q_1 \ldots Q_M}$?"
> **Type 2**: "Which point in the set S is the closest to the query polygonal line $\overline{Q_0Q_1 \ldots Q_M}$?"
> **Type 3**: "Enumerate all the points in the set S that are within the distance δ from the query polygonal line $\overline{Q_0Q_1 \ldots Q_M}$."

This problem may have some applications such as (Type 1) checking whether a circular robot can move along the polygonal line without hitting fixed circular obstacles, (Type 2) finding the nearest facility from a car moving along polygonal road, and (Type 3) enumerating all houses at which the magnitude of noise coming from a road or a railroad is larger than the standard one. The algorithm for Type 1 and Type 2 is described in the following part of this subsection. Type 3 is not solved directly in this paper, but can be solved by regarding this problem as a special case of the range search problem. At first, we construct the Voronoi diagram of the input point set S in the preprocessing step. As shown in Subsection 3.3, the algorithm of the side-trip facility search problem will be used for each range search based on the Voronoi diagram as its first step.

Suppose that we are given the query polygonal line $\overline{Q_0Q_1 \ldots Q_M}$. Firstly, we determine which Voronoi polygon includes the starting point Q_0. This step is a special type of the **point location**, in that the given tessellation is restricted to be the Voronoi diagram. This can be done by tracing the Voronoi polygons from an arbitrary initial Voronoi polygon step by step. Furthermore, for many types of input points, the average-case time complexity can be constant by the joint use of a bucketing technique [9], which is used for finding a suitable starting point also in this paper.

In the main part of the side-trip facility search, we trace the Voronoi polygon along the given polygonal line $\overline{Q_0Q_1 \ldots Q_M}$ (See Figure 3). In the course of this step, whenever we enter a new Voronoi polygon, we calculate the distance between the given polygonal line and the input point corresponding to the Voronoi polygon we currently stop by. As for Type 1 of the side-trip facility search problem, if the distance is within δ, the search is over and the answer is "Yes". If the answer "Yes" is not obtained by the step in which we reach the Voronoi polygon including Q_M, the answer is "No". As for Type 2, on the contrary, when the distance is smaller than the minimum one we have ever calculated, we update the minimum distance and memorize the current nearest point. When we reach the Voronoi polygon including Q_M, we obtain the minimum distance and the point which gives it.

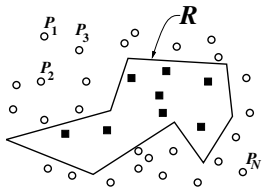


Fig. 1. Range search problem. (Solid lines represent the query range, filled squares represent points to be reported, and empty circles represent other points.)

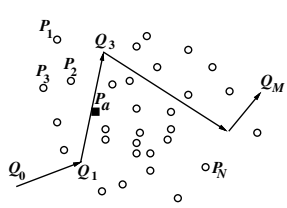


Fig. 2. Side-trip facility search problem. (Bold arrows represent the query polygonal line; square P_a is the answer point for Type 2.)

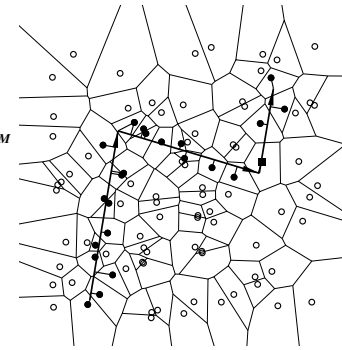


Fig. 3. Side-trip facility search based on the Voronoi diagram. (Solid circles represent the input points whose distances to the query polygonal line are calculated; short lines represent such distances.)

3.3 Procedure of the Range Search Based on the Voronoi Diagram

In this subsection, we construct the algorithm of the range search based on the Voronoi diagram. Similar to the side-trip facility search, we firstly construct the Voronoi diagram of the input point set S as the preprocess for this algorithm. we also prepare an array $\{\mathtt{A}[1],\ \mathtt{A}[2],\ \ldots, \mathtt{A}[N]\}$, each element has already been set `False`. It means that each input point is not checked yet. In case that an input point P_n is checked, we set $\mathtt{A}[n]$ `True`.

As the preparatory step of each query, We prepare a queue or a stack Q, which stores all of the input points that has already been checked, but whose neighbours have not yet been checked completely. At first, Q is empty. Then, there are following four steps, Step 1, Step 2, Step 3 and Step 4, in each query. Step 4 is the preparation for the succeeding query.

Step 1 (Point location on the Voronoi diagram)
Choose an arbitrary end point, say Q_0, of the boundary of the query range R, and find the input point P_s nearest to Q_0.

Step 2 (Remaining part of the side-trip facility search) (See Figure 4.)
Starting with Q_0, execute the side-trip facility search along the boundary of the query range R until we reach Q_0 again.

For each input point P_n traced in this search, **begin**
  If $A[n]$ is `False`, **begin**
    Insert P_n into Q. If P_n is inside R, report P_n. Set $A[n]$ `True`.
  end
end

Step 3 (Main part) (See Figure 5.)
While Q is not empty, **begin**

Delete one point P_n from Q.
For each point $P_{n'}$ neighbouring to P_n, **begin**
If $A[n']$ is `False`, **begin**
If $P_{n'}$ is inside R, report $P_{n'}$ and insert $P_{n'}$ into Q.
Set $A[n']$ `True`.
end
end
end

Step 4 Reset all of the elements `A`[1], `A`[2], ..., `A`[N] `False`.
(In order to avoid spending O(N) execution time at this step, we use another storage Q' which stores every input point checked so far. Thus, we can reset all the elements of A `False`, by just resetting the elements in Q'.)

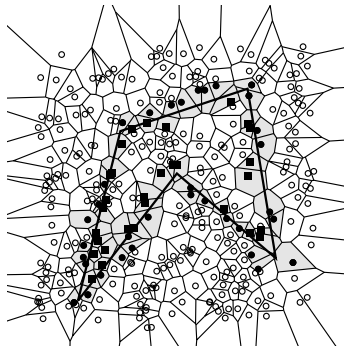


Fig. 4. Second step of the range search based on the Voronoi diagram. (Bold lines represent the boundary of the query range R; solid circles and squares represent the points traced at Step 2; light gray polygons represent the Voronoi polygon associated with such points; solid squares represent the points found inside R in this step.)

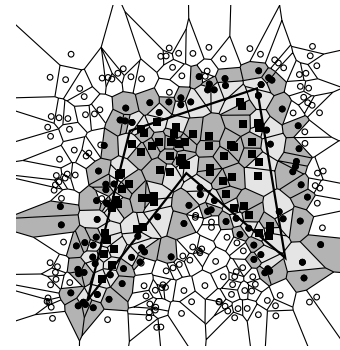


Fig. 5. Third step of the range search based on the Voronoi diagram. (Solid circles and squares represent the points traced at Step 2 and Step 3; dark gray polygons represent the Voronoi polygon traced at Step 3; solid squares represent the points found inside R at both steps.)

3.4 Validity of the Range Search Based on the Voronoi Diagram

In this subsection, we demonstrate the validity of the algorithm described above.

Firstly, the algorithm finds all the point inside the query range R. In order to prove this, we assume that there exists a point that is inside R but is not reported at either step. Then, we can derive a contradiction in the following way (See Figure 5).

Proof Let us start with classifying the input points into the following groups:
Type (I-1) (filled squares in Figure 5): points which are inside R and are reported,

Type (I-2): points which are inside R but are not reported,
Type (O-1) (filled circles in light gray polygons in Figure 5): points which are outside R and are traced at Step 2,
Type (O-2) (filled circles in dark gray polygons in Figure 5): points which are outside R, are not traced at Step 2, but are adjacent to at least one point which is traced at Step 2,
Type (O-3) (empty circles in Figure 5): points which are outside R, are not traced at Step 2, and are not adjacent to any point which is traced at Step 2.

Assume that there exists a point P_m in Type (I-2). Then, P_m is adjacent only to points in Type (I-2) or Type (O-2) or Type (O-3); because otherwise, P_m would be checked and reported. Therefore, there exists at least one point in Type (I-2) which is adjacent to a point in Type (O-2) or Type (O-3) (note that it cannot be true that all the input points belong to Type (I-2)). Consequently, there exists at least one point which is inside R and is not traced at Step 2, but is adjacent to a point which is outside R and is not traced at Step 2. This is a contradiction (indeed, the Voronoi polygons associated with points traced at Step 2 cannot have a gap, because the polygons include the boundary of the query range R). We can conclude that any point in Type (I-2) never exists. Q. E. D.

Secondly, we estimate the execution time of the range search based on the Voronoi diagram. As far as the number of Voronoi polygons which are adjacent to each Voronoi polygon can be regarded as less than a certain constant, the execution time is proportional to the number of points traced at Step 2 and Step 3. These points are classified into three types: (I-1), which just represents the group of points inside R, and Type (O-1) and Type (O-2). Suppose that we are given the query range R, and that the density of the input points becomes larger and larger. Then, we can expect that the number of points contained in R grows much faster that the number of input points which belong to Type (O-1) or Type (O-2). Therefore, the total number of input points traced in the algorithm is $O(K)$, where K denotes the number of points inside R. On the other hand, we can expect that the execution time is constant with respect to the number of input point N, exclusive of that of Step 1. The time complexity evaluated as above is the best possible average-case performance.

4 Numerical Experiments

In this section, we show the merits of the algorithm proposed in this paper from a practical point of view. We compared the execution time of our algorithm (inclusive of the point location) with those of the following three other representative algorithms.

Bucket: We use a simple bucketing technique. In this data structure, the whole square region where the input points are distributed is divided into small square cells called buckets. In the following experiments, the region $[0,\ 1] \times [0,\ 1]$ is divided into N/k buckets, where parameter k has been adjusted to 1.0. Note that k means the average number of input points in a bucket. Then we associate each of the input points one by one to the bucket which includes the input point.

At the first step of a query, we check whether each of the buckets intersects the query range or not. Note that this check is omitted for buckets which do not intersect even the circumscribing rectangle of the query range. Then, only when the bucket intersects the query range, we check whether each of the input points associated to the bucket is inside the query range.

The suitable value of k has been obtained by the following experiment shown in Figure 6. In this experiment, the execution times are measured for uniform input points and square query ranges mentioned below, where $N = 10000$, 20000, 50000, 100000, ..., 1000000, and $k = 0.01$, 0.02, 0.05, ..., 100. For all N's, the value 1.0 of k produces the minimum execution times.

k-d Tree (Direct Use): We use the k-d tree [1], one of the major tree structures used for geometrical search. Figure 7 is an example of the k-d tree. In this tree structure, the root node is associated with the whole region considered and one of the input points. All of the 2 child nodes, if any, are associated with smaller rectangular regions, both of which cover the region of the parent node without overlap. Each node is also associated with one of the input points within the corresponding region. The subdivision is done alternatingly by either the horizontal line or the vertical one passing through the input point corresponding to the current node. The direction of the division is decided by whether the depth of the current node is even or odd. The position of the division is selected so that the remaining input points are divided into 2 subsets with almost equal sizes.

In each query, we recursively traverse the tree to all of the leaves corresponding to the rectangular regions intersecting the query range, checking whether the input point associated to the rectangular region is inside the query range.

k-d Tree (After Triangulation): We use the k-d tree after triangulating the query range. In other word, for each of the triangles generated by the triangulation, we adopt the same method mentioned above. Finally, we report all of the input points reported for either triangle so far.

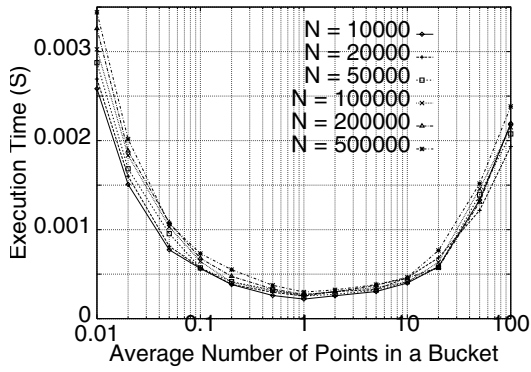


Fig. 6. Adjustment of the parameter k.

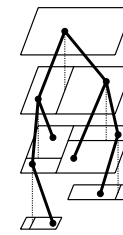


Fig. 7. Examples of the k-d tree.

These experiments were done for two types of input points such as

Uniform Input Points (Left of Figure 8): They are 10000 input points distributed uniformly in $[0,\ 1] \times [0,\ 1]$.

Clustered Input Points (Right of Figure 8): They are 10000 input points with 100 clusters distributed uniformly in $[0,\ 1] \times [0,\ 1]$. Each of these clusters consists of 100 points distributed according to the normal distribution with standard deviation 1×10^{-2}.

However, the performances for clustered input points are similar to those for uniform input points. Therefore, this paper shows only the performances for the uniform input points. The only exception is the performance of the k-d Tree (direct use) in case that the query range is an acute starred 100-gon mentioned below where the area is large. In this case, the execution times for the clustered input points are 1.5 times as long as those for the uniform input points.



Fig. 8. Two types of input points tested in this paper, 100 input points distributed uniformly (left), 100 input points with 10 clusters distributed uniformly, each of which consists of 10 points distributed according to the normal distribution with standard deviation 3×10^{-2} (right).

In these experiments, the shapes of query ranges are also restricted to the three types of shapes such as

Square (Figure 9, square, whose edges are parallel to the x-axis or the y-axis),
Round Starred 100-gon (Figure 10, comparatively round starred 100-gons),
Acute Starred 100-gon (Figure 11, comparatively acute starred 100-gons).

Through the following experiments, all the data of execution times are averages of 100 or more executions.

Firstly, we fixed the number of input points N to 10000, and changed the area of the query range from 0.01, 0.02, ... to 0.10, corresponding to the expected number of reported points K being 100, 200, ... to 1000. The execution times are shown in Figure 12, Figure 13 and Figure 14. In these figures, the horizontal axes mean the areas of query squares and the vertical ones mean the execution times.

We can see the execution times of all the algorithms are linear in the area of query ranges, that is, the expected number of reported points K. Especially, as shown in Figure 12, in case that the number of angle of the query ranges is small,

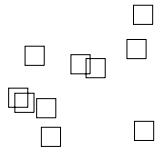


Fig. 9. Ten examples of squares used as query ranges. The areas of these squares are constantly 0.01.

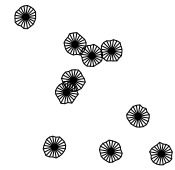


Fig. 10. Ten examples of round starred polygons used as query ranges. These are comparatively round starred 20-gons whose area is constantly 0.01. Each of them is triangulated by the triangles, one of the vertices of which is the core of the starred 20-gon. In the experiments, 100-gons of the same type are used as examples of query ranges.

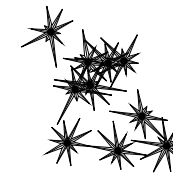


Fig. 11. Ten examples of acute starred polygons used as query ranges. These are comparatively acute starred 20-gons whose area is constantly 0.01. Each of them is triangulated in the same manner. In the experiments, 100-gons of the same type are used as examples of query ranges.

the algorithm based on the Voronoi diagram shows the best performance. Even in case that the number of angle of the query ranges is large, as shown in Figure 13 and Figure 14, as far as the area of the query ranges is small, the algorithm based on the Voronoi diagram still shows the best performance. If the query range is acute, that is, the length of the boundary is large, the algorithm using the k-d tree after triangulation is superior to that based on the Voronoi diagram. For the input points with clusters, the same tendency has been observed.

Secondly, we let the area of query ranges be so small that the expected number of reported points is 1, and let the number of input points N vary from 10000, 20000, ... to 100000. The execution times are shown in Figure 15, Figure 16 and Figure 17. In these figures, the horizontal axes mean the numbers of input points; the vertical ones mean the execution times.

For all algorithms, we can see the execution times are nearly constant, not depending on the number of input points N, as shown in 15, 16 and Figure 17. We can conclude that, if the number of reported points is small enough, the algorithm based on the Voronoi diagram shows the best performance, for any types of query ranges tested in these experiments.

5 Conclusions

We have constructed a simple and efficient algorithm for solving the two-dimensional range search problem with respect to general shapes of query ranges. In this method, the Voronoi diagram for the input points is constructed in the preprocessing step, and is used to restrict the points to visit in the main step

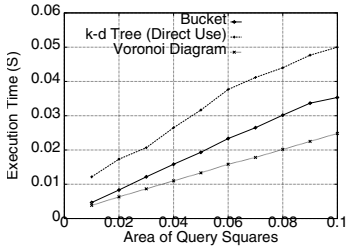


Fig. 12. Relation between K and execution times of the range search for **uniform input points** and **squares**.

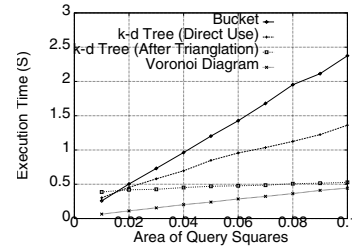


Fig. 13. Relation between K and execution times of the range search for **uniform input points** and **round starred 100-gons**.

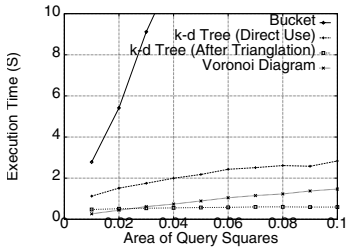


Fig. 14. Relation between K and execution times of the range search for **uniform input points** and **acute starred 100-gons**.

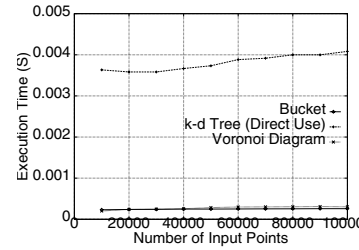


Fig. 15. Relation between N and execution times of the range search for **uniform input points** and **squares**.

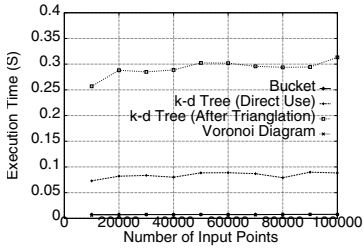


Fig. 16. Relation between N and execution times of the range search for **uniform input points** and **round starred 100-gons**.

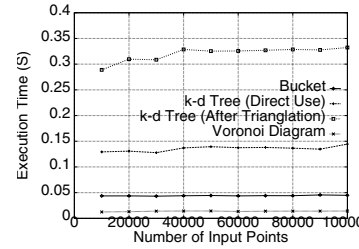


Fig. 17. Relation between N and execution times of the range search for **uniform input points** and **acute starred 100-gons**.

of the search. Then, for uniformly distributed input points and clustered input points, a good average-case performance is observed by numerical experiments; the execution time of this search is proportional to the area of query ranges, and little depends on the number of input points. Furthermore, we observed that the algorithm based on the Voronoi diagram shows the best performance in almost all the cases. It is only when the number of reported points is large and the number of angles of query ranges is also large, that the algorithm based on the Voronoi diagram is inferior to another algorithm, the one using the k-d tree after triangulation. ¿From the practical point of view, the algorithm introduced in this paper can replace many of the other algorithms, as far as the dimension is two.

Acknowledgements

This paper is partially supported by the Grant-in-Aid for Scientific Research of the Japanese Ministry of Education, Culture, Sports Science and Technology.

References

1. J. L. Bentley: Multidimensional binary search trees used for associative searching, *Communications of the ACM*, vol. 18 (1975) 509–517.
2. J. L. Bentley, D. F. Stanat, E. H. Williams, Jr.: The complexity of finding fixed-radius near neighbors, *Information processing letters*, vol. 6, no. 6 (1977) 209–212.
3. J. L. Bentley, H. A. Maurer: A note on Euclidean near neighbor searching in the plane, *Information processing letters*, vol. 8, no. 3 (1979) 133–136.
4. M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf: Computational geometry — algorithms and applications —, Springer-Verlag, Berlin, Heidelberg (1997).
5. B. Chazelle, R. Cole, F. P. Prepareta, C. Yap: New upper bounds for neighbor searching, *Information and control*, vol. 68 (1986) 105–124.
6. H. Edelsbrunner, E. Welzl: Halfplanar range search in linear space and $O(n^{0.695})$ query time, *Information processing letters*, vol. 23 (1986) 289–293.
7. J. E. Goodman, J. O'Rourke: *Handbook of discrete and computational geometry*, CRC press, Boca Raton, New York (1997).
8. J. Matoušek: Geometric range searching, ACM computing survey, vol. 26, no. 4 (1994).
9. T. Ohya, M. Iri, K. Murota: Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms, *Journal of operations research society of Japan*, vol. 27, no. 4 (1984) 69–97.
10. A. Okabe, B. Boots, K. Sugihara, S.-N. Chiu: *Spatial tessellations — concepts and applications of Voronoi diagrams*, second edition, John Wiley & Sons (2000).
11. F. P. Preparata, M. I. Shamos: *Computational geometry — an introduction*, Springer-Verlag, New York (1985).
12. G. Yuval: Finding near neighbours in k-dimensional space, *Information processing letters*, vol. 3, no. 4, pp. 113–114, March 1975.
13. C. Böhm, S. Berchtold and D. A. Keim: Searching in high-dimensional spaces — Index structures for improving the performance of multimedia databases, *ACM Computing Surveys*, vol. 33, No. 3 (2001) 322–373.
14. E. Chávez, G. Navarro, R. Baeza-Yates and J. L. Marroquín: Searching in metric spaces, *ACM Computing Surveys*, vol. 33, No. 3 (2001) 273–321.

Computing a Closest Point to a Query Hyperplane in Three and Higher Dimensions

Pinaki Mitra[1] and Asish Mukhopadhyay[2]

[1] NIMC
Alipore, Calcutta
India
[2] School of Computer Science
University of Windsor
Windsor, Canada

Abstract. Given a set S of n points in E^d ($d \geq 3$), we consider the problem of computing a closest point to a query hyperplane Q. For $d = 3$, we report an algorithm whose preprocessing time is in $O(n^{1+\epsilon})$, space complexity is in $O(n \log n)$ and query time is in $O(n^{2/3+\epsilon})$.
For $d > 3$, we adopt a different approach and propose an algorithm which has a query time in $O(d \log n)$, in an amortized sense, under a rather strong assumption that we explain in the paper, with $O(n^{d+\kappa})$ preprocessing space and, $O(n^{d+1+\kappa})$ preprocessing time, both in an expected sense, for some $\kappa > 0$.

1 Introduction

The problem of preprocessing a set S of n points in the plane to determine the closest point to a query line was initially addressed by Cole and Yap [1], who obtained a solution with preprocessing time and space in $O(n^2)$ and query time in $O(\log n)$. Lee and Ching [4] obtained the same result, using geometric duality. In [7], an algorithm was presented with preprocessing time and space in $O(n \log n)$ and query time in $O(n^{0.695})$. The space complexity was improved to $O(n)$ in a subsequent result by Mitra and Chaudhuri [8]. In [5], the simplicial partition technique of [9] was used to improve the query time to $O(n^{1/2+\epsilon})$ for arbitrary $\epsilon > 0$. In this paper, we consider the problem in three and higher dimensions. For the three dimensional version of the problem, we extend the technique of [5] to obtain an algorithm whose preprocessing time and space complexities are in $O(n^{1+\epsilon})$ and $O(n \log n)$ respectively, while the query time complexity is in $O(n^{2/3+\epsilon})$.

For $d > 3$, we generalize the approach of [4]. The main idea is to locate the query point (corresponding to the query hyperplane) in a d-dimensional arrangement of hyperplanes. The hyperplanes immediately above and below this point that bound this cell correspond to candidate closest points in the primal plane. This is achieved by using a general point location algorithm due to Meiser [6] in the preprocessing stage. This leads to an algorithm which has a query time in $O(d \log n)$ query time, in an amortized sense, under a rather strong assumption

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 787–796, 2003.
© Springer-Verlag Berlin Heidelberg 2003

that we explain in the paper, with $O(n^{d+\kappa})$ preprocessing space and $O(n^{d+1+\kappa})$ preprocessing time, both in an expected sense, for some $\kappa > 0$.

The practical applications that motivate these problems come from the areas of pattern recognition and image processing. For example, if the query hyperplane separates two classes of pattern, then an answer to the query determines the pattern closest to the query line and provides an idea about the stability of classification.

2 Geometric Insights for the 3-dimensional Version

The most trivial solution to the problem will check each point in turn and compute the distance from the query hyperplane and report the point having the least distance. This will take $O(n)$ query time without any preprocessing. We improve this naive solution by preprocessing the point set so that it can be queried more efficiently.

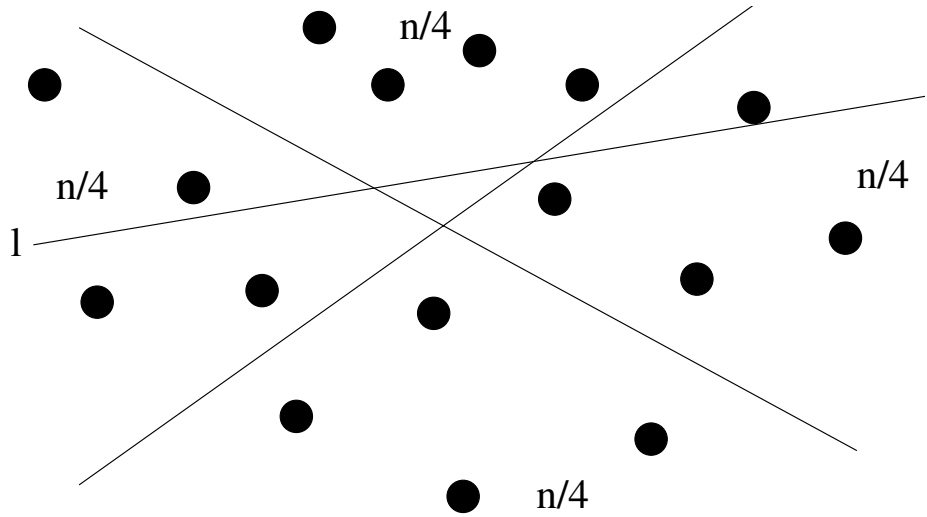


Fig. 1. Line l cuts a hamsandwich partition

In [8] the idea of hamsandwich cut was used. An arbitrary query line can intersect at most three of the four regions into which a plane is partitioned by two intersecting lines. This observation was used to partition a point set into 4 groups each of size $n/4$ using hamsandwich cuts such that the query line is disjoint from at least one of the four groups (see Fig. 1). The recursive hamsandwich partitions of the groups intersected by the query line allows one to identify smaller subgroups of points disjoint from the query line at each level of the recursion. Each group of points disjoint from the query line is queried with respect to its convex hull since a closest point is on the hull boundary.

If $Q(n)$ is the time to resolve a query on n points then it satisfies the recurrence

$$Q(n) \leq Q(n/2) + Q(n/4) + O(\log n) \tag{1}$$

where $O(\log n)$ is the time required to resolve the closest point query with respect to $n/4$ points that are provably disjoint from the query line.

In this paper, we use the simplicial partition by Matousek [9] to improve the query time. Here we also have to solve the subproblem of computing the closest point to a query plane from a non-intersecting convex polyhedron.

3 Simplicial Partitions

Simplicial partitions have been used in the context of range-searching [10]. In this paper we explore its applicability to the problem at hand.

A *simplicial partition* of S is a collection of pairs

$$\Psi(S) = \{(S_1, t_1), (S_2, t_2), \ldots, (S_r, t_r)\} \tag{2}$$

where the S_i's are mutually disjoint subsets of S whose union is S, and each t_i is a tetrahedron that contains S_i (see Fig. 2).

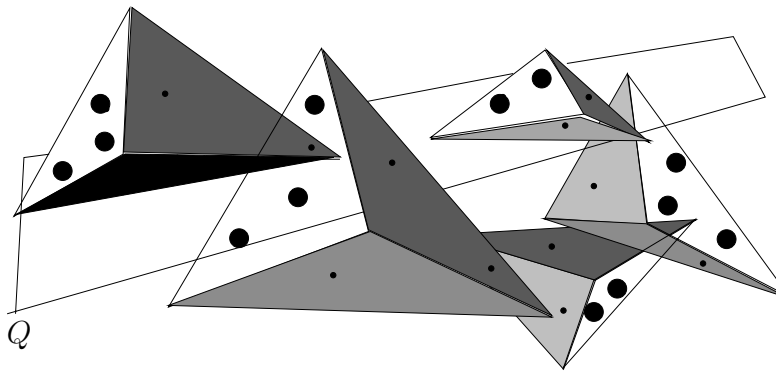


Fig. 2. Plane q cuts a simplicial partition

A simplicial partition is *fine* if $|S_i| \leq 2 * n/r$ for each i. For a given simplicial partition $\Psi(S)$, the *crossing number* of a plane q is the number of tetrahedrons of $\Psi(S)$ that the plane properly intersects. The crossing number of $\Psi(S)$ is the maximum crossing number over all planes that intersect the simplicial partition.

The following important result by Matousek [9] shows how to construct a fine simplicial partition with a low crossing number.

Theorem 1. *For any given set S of n d-dimensional points and a parameter r, $1 \leq r \leq n$, a fine simplicial partition of size r and crossing number $O(r^{(1-1/d)})$ exists. Further, for any given $\epsilon > 0$, such a simplicial partition can be constructed in time $O(n^{1+\epsilon})$.*

A simplicial partition is stored in an r-way tree, called a *partition tree*. At each of the r children of the root of a partition tree we store the simplex that encloses the corresponding point set, as well as any secondary information related to the point set. Each child-node is in turn the root of a partition tree on the point set that belongs to this node.

4 Algorithmic Details

The constructibility of a fine simplicial partition with a low crossing number implies that the point sets enclosed by $r - O(r^{2/3})$ simplices are not intersected by a query plane q and the convex hulls of these can be queried for the closest point to the query plane. For the remaining $O(r^{2/3})$ simplices that are intersected by the query plane, we proceed recursively on their respective simplicial partitions.

To determine the closest vertex of a convex polyhedron to a non-intersecting query plane, we compute the ray diagram of the convex polyhedron. This means constructing the dual subdivision of the face stucture of the convex polyhedron on the surface of a unit sphere by means of the normals to the facets of the polyhedron.

We consider the stereographic projection of this planar subdivision on the sphere to a plane. We preprocess this stereographic projection for point location using Kirkpatrick's algorithm [3]. For a given query plane, we consider a direction parallel to a normal to it, passing through the center of the unit sphere.

We locate the intesections of this direction with the unit sphere in the planar subdivision in $O(\log n)$ time. One of the two vertices corresponding to the resulting regions of the planar point location query will be the closest vertex of the convex polyhedron to the query plane.

Thus we have established the following theorem.

Theorem 2. *Given a n vertex convex polyhedra in three dimension the closest vertex from a non-intersecting query plane q can be reported in $O(\log n)$ query time with $O(n)$ preprocessing.*

In the analysis below we appeal to the following theorem :

Theorem 3. *The convex hull of a set of n three dimensional points can be computed in $O(n \log n)$ time. [11]*

4.1 Preprocessing Time

The processing time required to build the partition tree is $O(n^{1+\epsilon})$ by Theorem 1 above. After building the partition tree, we compute the convex hull of the

points associated with each node v. If $T(n)$ be the time required to compute this information on a partition tree for n nodes then it satisfies the recurrence inequality:

$$T(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ O(n \log n) + \sum_1^r T(n_v) & \text{if } n > 1 \end{cases}$$

Since $n_v \leq 2n/r$, for $r > 2$, $n_v < n$. Then as $\sum n_v = n$ and r is a constant, we conclude that $T(n) = O(n \log^2 n)$.

Since $\log^2 n = o(n^\epsilon)$ for arbitrary $\epsilon > 0$, the time needed to construct the partition tree dominates the preprocessing time.

4.2 Preprocessing Space

If $S(n)$ be the space required in a partition tree on n nodes, then it satisfies the recurrence inequality:

$$S(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ n + r + \sum_1^r S(n_v) & \text{if } n > 1 \end{cases}$$

where n_v is the number of points at node v. Since $n_v \leq 2n/r$, for $r > 2$, $n_v < n$. Then as $\sum n_v = n$ and r is a constant, we conclude that $S(n) = O(n \log n)$.

4.3 Query Time

If we let $Q(n)$ denote the query time on a point set of size n then it satisfies the following inequality:

$$Q(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ r + \sum O(\log(n_v)) + \sum Q(n_v) & \text{if } n > 1 \end{cases}$$

since the nearest point query is done on the convex hull of at least $r - cr^{2/3}$ sets S_i, and we recur on at most $cr^{2/3}$ subsets S_i that are intersected by the query plane q. The sum $\sum Q(n_v)$ is precisely taken over these latter subsets, while the sum $\sum O(\log(n_v)$ is taken over the remaining subsets whose convex hulls are queried for the closest point to the query plane q as in **Theorem 2**.

For a given $\epsilon > 0$ and $c > 0$, if we set $r = \lceil 2(2^{5/3} * c)^{\frac{1}{\epsilon}} \rceil$ then r becomes independent of n and we can use induction to show that

$$Q(n) = O(n^{2/3+\epsilon}) \tag{3}$$

5 Extension to Higher Dimensions

To extend the result to dimensions higher than 3, we consider the dual arrangement $\mathcal{A}$ of the set of hyperplanes corresponding to the d-dimensional point set

S in the primal plane. The idea here is to locate the dual point corresponding to a query hyperplane q in a cell $\mathcal{C}$ of the arrangement $\mathcal{A}$. Then the dual of the hyperplane corresponding to the top or bottom facet in the cell $\mathcal{C}$ along the d^{th} co-ordinate which is at minimum distance from q will be the answer to our query, since the vertical distance is preserved under dual transformation. So for this purpose we have to perform point location and subsequently a ray shooting query within the cell $\mathcal{C}$. We will use the following result by Meiser [6] to perform the point location in a d-dimensional arrangement of hyperplanes; however, we push its use to the preprocessing stage.

Theorem 4. *Point location in an arrangement of hyperplanes in E^d can be solved with query time $O(d^5 \log n)$, expected (in a randomized sense) space $O(n^{d+\kappa})$ and expected (again, in a randomized sense) time $O(n^{d+1+\kappa})$ for arbitrary $\kappa > 0$, where n is the number of hyperplanes.*

To resolve a query, we propose the following solution. We construct a grid partition of the bounding box of the $\binom{n}{d} = O(n^d)$ intersection points of the hyperplane arrangement $\mathcal{A}$. Let us denote the d coordinates in E^d by $x_1, x_2, ..., x_d$. The position vector of a point p, with respect to a hyperplane $h \in \mathcal{A}$, expressed by the equation $a_1x_1 + a_2x_2 + ... + a_dx_d + a_{d+1} = 0$, is denoted by:

$$pv(p) = \begin{cases} + \text{ if } p \in h^+ \\ 0 \text{ if } p \in h \\ - \text{ if } p \in h^- \end{cases}$$

We say a point $p = (p_1, p_2, ..., p_d) \in h^+$ where h is represented by $a_1x_1 + a_2x_2 + ... + r_dx_d + a_{d+1} = 0$ if $a_1p_1 + a_2p_2 + ... + a_dp_d + a_{d+1} > 0$. A similar definition holds when $p \in h^-$. Clearly, all the points in the interior of a d-dimensional cell of the arrangement $\mathcal{A}$ in E^d have the same position vector.

The d-dimensional grid is formed by equally dividing each of the intervals $(x_{1\,\min}, x_{1\,\max}), (x_{2\,\min}, x_{2\,\max}), ..., (x_{d\,\min}, x_{d\,\max})$ into $(n+1)$ parts. Thus we are superimposing a $(n \times n \times ... \times n)_d$ grid over the d-dimensional arrangement $\mathcal{A}$ (see Fig. 3). The basic intuition underlying our algorithm is that, in an amortized sense, only a few hyperplanes intersect each cell of the d-dimensional grid. The following theorem will be useful for analysing our point location algorithm.

Theorem 5. *Amortized over all the cells of the grid partition superimposed on the arrangement, only a constant, i.e., $O(1)$ hyperplanes, of the arrangement $\mathcal{A}$ pass through each cell of the grid.*

Proof: We take a d-dimensional hyperplane $h \in \mathcal{A}$. Consider the intersection of each of the dn hyperplanes of the $(n \times n \times ... \times n)_d$ grid with h. These intersections give rise to a $(d-1)$-dimensional arrangement of dn hyperplanes in h. The complexity of this hyperplane arrangement is $O((dn)^{(d-1)})$. So this must be the order of the number of cells through which h has passed in the

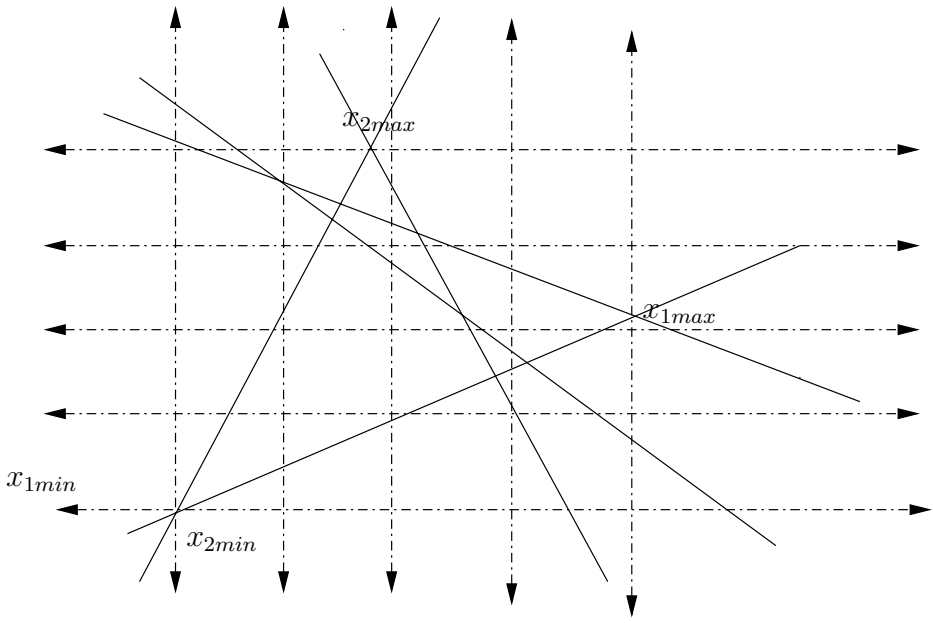


Fig. 3. 5×5 grid superimposed over an arrangement $\mathcal{A}$ of 5 lines

$(n \times n \times ... \times n)_d$ grid. Now if we consider all n hyperplanes of the arrangement $\mathcal{A}$, the order of the number of cells it intersects will be $O(n * (dn)^{(d-1)})$. So, amortized over the $O(n^d)$ cells in the grid, $O(n * (dn)^{(d-1)})/O(n^d)$, i.e., $O(1)$ (for fixed d) hyperplanes intersect each cell of the $(n \times n \times ... \times n)_d$ grid. **Q.E.D**

5.1 Preprocessing

Our preprocessing algorithm consists of the following steps.

Step I. Compute the arrangement $\mathcal{A}$ of n d-dimensional hyperplanes using the topological plane sweep algorithm of Edelsbrunner and Guibas [2].

Step II. Compute the arrangement of hyperplanes intersecting each cell of the $(n \times n \times ... \times n)_d$ grid. This partitions each grid cell into zero (if no hyperplane intersects the cell) or more sub-cells.

Step III. From each of the subcells we take an interior point and perform point location using **Theorem 4** to idenitify the d-dimensional cell of the arrangement $\mathcal{A}$ computed in step I and maintain a pointer to that cell.

We have to estimate the time and space complexity of this preprocessing algorithm. The topological plane sweep algorithm of Edelsbrunner and Guibas in **step I** runs in $O(n^d)$ time and space [2].

Since from **Theorem 5** only $O(1)$ hyperplanes, in an amortized sense, intersect each cell, the computation time for **step II**, amortized over all the grid cells, is $O(n^d)$.

The time required to create the pointers to the d-dimensional cells of $\mathcal{A}$ using the point location algorithm of Meiser [6] in **step III** will be $O(n^d * d^5 \log n)$, i.e., $O(n^d \log n)$, amortized over all the cells in the grid, while the expected space of $O(n^{d+\kappa})$, for some $\kappa > 0$ is that of Meiser's algorithm. The time complexity of this step is dominated by Meiser's algorithm which is $O(n^{d+1+\kappa})$ for some $\kappa > 0$, in an expected sense again.

5.2 Query Processing

Given a query hyperplane q we go through the following steps to answer the query :

Step I. Perform binary search in each dimension to locate the cell in the $(n \times n \times ... \times n)_d$ grid in which the dual point corresponding to the query hyperplane q lies.

Step II. Compute the position vector, i.e., $pv(dual(q))$ with respect to the hyperplanes intersecting the cell of the grid. If $pv(dual(q)) = 0$ for some hyperplane then the dual point corresponding to the hyperplane is answer to our query. Otherwise we proceed to the subsequent steps.

Step III. From the position vector determine the cell $\mathcal{C}$ of the arrangement $\mathcal{A}$ in which the dual of q lies.

Step IV. From the arrangement cell $\mathcal{C}$ identify the closest facet along d^{th} co-ordinate by intersecting every $(d-1)$-dimensional hyperplane corresponding to each facet of the cell $\mathcal{C}$, with a ray along d^{th} co-ordinate passing through q and finding the one whose intersection is at least distance from q. The dual point corresponding to this hyperplane is the answer to our query.

For the analysis below, we make the rather strong assumption that point dual of the query hyperplane is equally likey to be in any cell of the arrangement $\mathcal{A}$.

Analyzing the time complexity of the query algorithm, we note that **Step I** can be carried out in $d \log n$ time. This is the time complexity to carry out binary search in d different dimensions. From **Theorem 5**, the amortized complexity of the hyperplane arrangement, restricted to each cell of the grid is $O(1)$. Appealing to the assumption above, we can then claim that **Step II** and **Step III** can be carried out in $O(1)$ time, in an amortized sense, from the preprocessed information. It remains to bound the complexity of **Step IV**. For this step, we will examine each facet of the cell $\mathcal{C}$ of the arrangement $\mathcal{A}$. The following theorem will help us in bounding the time complexity of this step.

Theorem 6. *In a d-dimensional arrangement $\mathcal{A}$ of n hyperplanes the amortized count of the number of facets of an individual d-dimensional cell $\mathcal{C}$ is $O(1)$, for every fixed dimension d.*

Proof: We consider a d-dimensional hyperplane $h \in \mathcal{A}$. Consider the projection of the rest of the $(n-1)$ hyperplanes on h. It forms a $(d-1)$-dimensional arrangement of $(n-1)$ hyperplanes. The complexity of this arrangement is $O((n-1)^{(d-1)})$. So the hyperplane h contributes $O((n-1)^{(d-1)})$ facets in the d-dimensional arrangement $\mathcal{A}$. Now if we consider all n hyperplanes of the arrangement $\mathcal{A}$ then they in total contribute to $O(n*(n-1)^{(d-1)})$ facets. We know total number of d-dimensional cells in the arrangement $\mathcal{A}$ is $O(n^d)$. So average number of facets for a d-dimensional cell is $O(n*(n-1)^{(d-1)})/O(n^d)$, i.e., $O(1)$ for fixed d. **Q.E.D**

Using the last theorem and the assumption made above, we claim that the time complexity of **Step IV** of the query algorithm, in an amortized sense, is $O(1)$ for every fixed dimension d, since, on average, we will examine only a small number of facets that bound a d-dimensional cell $\mathcal{C}$ of the arrangement $\mathcal{A}$.

Thus we have established the following theorem :

Theorem 7. *Given a set S of n points in d-dimension, the closest point from a $(d-1)$-dimensional query hyperplane can be answered in $O(d \log n)$ query time, in an amortized sense, with $O(n^{d+\kappa})$ preprocessing space and, $O(n^{d+1+\kappa})$ preprocessing time, both in an expected sense, for some $\kappa > 0$.*

6 Conclusion and Open Problems

In this paper we have shown that the results of Mitra and Chaudhuri [8] can be generalized to higher dimensions. For the three dimensional version of the problem we have been able to improve the query time using the idea of simplicial partition at the expense of preprocessing time and storage. An open question concerns the computation of the closest of point in a set of n three dimensional points to a query plane q with $O(n \log n)$ preprocessing time $O(n)$ storage and $O(n^{2/3})$ query time. For the higher dimensional version of the open problem concerns the design of an algorithm which reports the closest point in a set of n d-dimensional points to a $(d-1)$ dimensional query hyperplane with worst case $O(n^d)$ preprocessing time and space and worst case $O(polylog(n))$ query time.

References

1. R. Cole and C. Yap, Geometric retrieval problems, *Proc. IEEE 24th Symp. on Foundations of Computer Science*, pp 112 - 121, 1983.
2. H. Edelsbrunner and L. J. Guibas, Topologically sweeping an arrangement, *J. Comput. Syst. Sci., Vol. 38,* pp 165 - 194, 1989.
3. D. G. Kirkpatrick, Optimal search in planar subdivisions, *SIAM J. Comput., Vol. 12(1)*, pp 28 - 35, 1983.
4. D. T. Lee and Y. T. Ching, The power of geometric duality revisited, *IPL, Vol. 21*, pp 117 - 122, 1985.
5. Asish Mukhopadhyay, Using simplicial partitions to determine the closest point to a query line, in *Pattern Recognition Letters*, to appear .

6. S. Meiser, Point location in arrangements of hyperplanes, *Information and Computation, Vol. 106*, pp 286 - 303, 1993.
7. P. Mitra, Finding the closest point to a query line, *Snapshots in Computational Geometry* (Editor G. Toussaint), Vol II, pp 53 - 63, 1992.
8. P. Mitra and B. B. Chaudhuri, Efficiently computing tthe closest point to a query line, *Pattern Recognition Letters, Vol 19*, pp 1027 - 1035, 1998.
9. J. Matoušek, Efficient partition trees, *Discrete and Computational Geometry, Vol. 8*, pp 315 - 334, 1992.
10. Mark de Berg, Marc van Kreveld, Mark Overmarks, Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications, Second edition, Springer Verlag*, 2001.
11. Franco P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction, Springer-Verlag, 2nd edition*, 1990.

Computing a Largest Empty Arbitrary Oriented Rectangle: Theory and Implementation

Asish Mukhopadhyay[1] and S.V. Rao[2]

[1] School of Computer Science
University of Windsor
Windsor, Canada
asishm@uwindsor.ca
[2] Department of Computer Sci. & Engg.
Indian Institute of Technology
Guwahati, India
svrao@iitg.ernet.in

Abstract. In this paper, we report on the theory and implementation of an $O(n^3)$ algorithm for computing the largest empty rectangle of *arbitrary orientation* for a given set of n points in the plane.

1 Introduction

Geometric optimization is a very active area of research in the fast-growing field of Computational Geometry [AS98]. The motivation for this study comes from a variety of practical problems such as stock-cutting, certain packing and optimal layout problems.

An optimization problem, known in the literature as the largest empty rectangle problem. that attracted a lot of attention is the following: **Given a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ inside a rectangular area R, the problem of finding the largest area (perimeter) rectangle with sides parallel to those of the original rectangle R.**

A practical application that motivates this problem is this: ¿From a rectangular piece of metal sheet R with a set of "flaw" points on it, cut out a largest area (perimeter) rectangle without any flaw points.

This problem was introduced into the literature by [NHL84], who proposed an algorithm with worst-case running time of $O(n^2)$ and expected running time of $O(n \log^2 n)$. The worst-case complexity was improved to $O(n \log^3 n)$ by [CDL86]. This was further improved to $O(n \log^2 n)$ in [AS87]. In [Orl90] an ouput-sensitive algorithm was proposed with worst-case complexity in $O(n \log n + s)$ and space complexity in $O(n)$.

2 Overview

Assume that the given set of n points are in the first quadrant of a rectangular system of coordinate axes.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 797–806, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Consider a grid diagram obtained by drawing a pair of orthogonal lines through each of the n points, parallel to two fixed directions. The orientation θ of our grid is the smaller of the two angles made by these two directions with the x-axis of our coordinate system.

The lines of the grid determine many different empty rectangles. A *maximal empty rectangle* (MER) is one that contains no points in its interior and is not properly contained in any other empty rectangle.

We consider MERs that are bounded by a point on each of its four sides. We therefore dispense with the axis-aligned rectangle R that encloses the point set and assume that it is large enough to enclose such an MER of arbitrary orientation. Our goal is to find such an MER that has maximum area over all possible grid orientations.

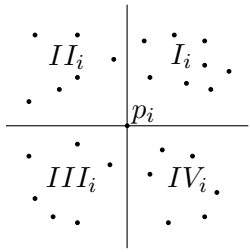


Fig. 1. Points are divided into four groups.

The problem has a discrete structure for as we change the grid orientation continuously, say counterclockwise, all MERs survive till some two points of our set become collinear along one of the two orthogonal directions. At this time, new MERs are also generated. Together, these constitute an event and we have $O(n^2)$ such events for each choice of a pair of points.

We denote a range of MERs bounded by the points p_t, p_b, p_l, and p_r by $< p_t, p_b, p_l, p_r, \alpha, \beta >$, where α is the grid orientation at which this range appears and β is the grid orientation at which it disappears. An MER attains maximum area, for some intermediate value of θ in this range, which is determined using calculus.

The essence of our solution, therefore, is to generate this discrete set of events and compute a maximum area MER over all rectangle ranges.

3 Algorithm

3.1 Staircases and Maximal Empty Rectangles

For a given point $p_i \in P$ and a grid-orientation, we characterize the MERs that are bounded by the point p_i on one of its sides as below.

The pair of orthogonal lines through $p_i \in P$, determined by the given grid-orientation, partitions the point set into four disjoint subsets, that we label I_i, II_i, III_i and IV_i as shown in Fig. 1.

The *maxima* [PS90] of each group has a staircase structure as shown in Fig. 2. We label by S_{X_i} the staircase of group X of point p_i. Each maximal empty rectangle that has the point p_i on one of its sides, has the points on the rest of the three sides on a pair of staircases in adjacent quadrants.

The points on S_{I_i} (and, similarly, those on S_{II_i}, S_{III_i} and S_{IV_i}) can be ordered counterclockwise with respect to p_i. We denote by $next(S_{I_i}, p_j)$ ($previous(S_{I_i}, p_j)$), the point next (previous) to p_j in this ordering, whenever such a point exists.

For a given grid orientation, we denote the orthogonal lines passing through a point p_i by h_i and v_i respectively, h_i making the smaller angle with the x-axis.

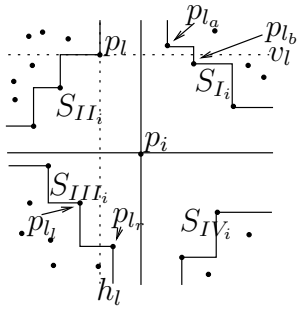


Fig. 2. Stair case of p_i.

Let p_l be a point on S_{II_i}. We denote by p_{l_b} (similarly, p_{l_a}) the point immediately below (above) the line h_l in S_{I_i}. Similarly, we denote by p_{l_l} (similarly, p_{l_r}) the point immediately to the left (right) of the line v_l in S_{III_i} (see Fig. 2). Likewise, we can define the same four pointers for each point on S_{I_i}, S_{III_i} and S_{IV_i}. We denote by S_i, these four staircases of p_i along with the pointers above, below, left and right for each point on them.

The complexity of computing S_i is given by the following lemma.

Lemma 1. *Given a set P of n points, sorted by their x-coordinates and a point $p_i \in P$, we can compute the staircase S_i in $O(n)$ time, using $O(n)$ space.*

The following observation characterizes the largest empty isothetic rectangle, which contains the point p_i on one of its edges.

Observation 3.1 *The largest empty isothetic rectangle whose right (similarly, left) edge contains the point p_i is bounded between S_{II_i} and S_{III_i} (S_{I_i} and S_{IV_i}) and the vertical line passing through p_i. Similarly, the largest empty isothetic rectangle whose top (similarly, bottom) edge contains the point p_i is bounded between S_{III_i} and S_{IV_i} (S_{I_i} and S_{III_i}) and the horizontal line passing through p_i.*

In other words, each corner of the stair-case S_i defines two MERs, and each corner is defined by two consecutive points on S_i. The following easy lemma states the time required for computing the largest area MER, bounded by p_i on one of its sides.

Lemma 2. *Given an S_i, the MERs bounded by p_i can be determined in $O(n)$ time.*

3.2 Basic Continuous Optimization Problem

In this subsection we discuss how to compute the maximum measure of an MER between the events, corresponding to $\theta = \alpha$ and $\theta = \beta$.

Let r be an MER bounded by the points be p_t, p_b, p_l, and p_r as shown in Fig. 3. As r always contains the quadrilateral, $< p_t, p_b, p_l$ and $p_r >$, its area measure is maximized when the sum of the areas of the triangles that make up the rest of r is maximized. This is easily reduced to a single parameter optimization problem.

The event that triggers the disappearance of the the above class of MERs can occur in two ways. One, a new point comes into it, making it non-empty; second,

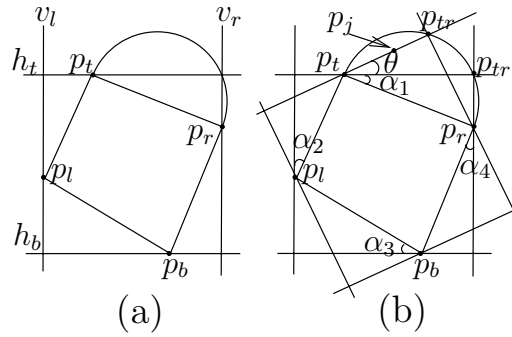


Fig. 3. Continuous optimization problem.

one of the sides of the MER coincides with a side of the inscribed quadrilateral. We discuss only the details of the first case below as the second case can be treated similarly.

Let this event be triggered by the appearance of the point p_j on the top edge of r.

If we let p_{tr} be the intersection of h_t and v_r then the following observations are straightforward.

Observation 3.2 *The area of the triangle* $< p_t, p_r, p_{tr} >$ *is* $\frac{1}{4}|\overline{p_t p_r}|^2 \sin 2(\alpha_1 + \theta)$*, where* $\theta \in [0, \beta - \alpha]$ *(see Fig. 3b).*

Observation 3.3 *The largest area rectangle in* $< p_t, p_b, p_l, p_r, \alpha, \beta >$ *is* $\frac{1}{4} Max_{\theta \in [0,\beta-\alpha]} \{|\overline{p_t p_r}|^2 \sin 2(\alpha_1 + \theta) + |\overline{p_l p_t}|^2 \sin 2(\alpha_2 + \theta) + |\overline{p_l p_b}|^2 \sin 2(\alpha_3 + \theta) + |\overline{p_r p_b}|^2 \sin 2(\alpha_4 + \theta)\}$.

We can solve the maximization problem in constant time, using calculus. Hence the following lemma.

Lemma 3. *The largest area/perimeter empty arbitrary oriented rectangle in a given range* $< p_t, p_b, p_l, p_r, \alpha, \beta >$*, can be found in* $O(1)$ *time.*

Please note that another range of empty rectangles bounded by the points p_t, p_b, p_l and p_r may be found, if we rotate further (see Fig. 4).

3.3 Event Queue and Staircase Updates

We first look at the following problem: For a given point $p_i \in P$, find the largest measure arbitrary oriented MER bounded by the point p_i on one of its sides.

A rectangle range begins and ends at event points. An event point in our scheme is the collinearity of two points p_j and p_k along one of the two orthogonal directions as the grid rotates in the range from $\theta = 0$ to $\theta = \pi/2$. Let $E_h(p_j, p_k)$ (respectively $E_v(p_j, p_k)$) denote the event triggered by the collinearity of p_j

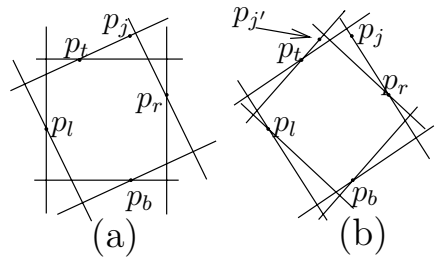


Fig. 4. Two different ranges of rectangles bounded by four points.

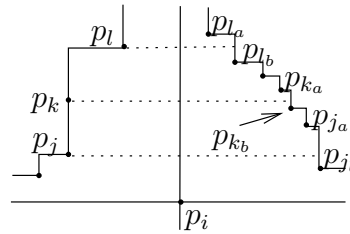


Fig. 5. New point on a staircase.

and p_k along a "horizontal" ("vertical") direction for which $\theta = angle(p_j, p_k)$ $(\theta + \pi/2 = angle(p_j, p_k))$.

The event points are maintained in an event queue, Q, which is a sorted list of the absolute values of the slopes of the supporting lines of line segments determined by points taken pairwise from P. It can be initialized in $O(n^2 \log n)$ time. We take absolute values, as an event can be a "horizontal" or a "vertical" one.

As each MER is implicitly maintained in the staircase structure S_i and can be initialized in $O(n \log n)$ time, we address the following questions:

- How to update an S_i at an event ?
- How to keep track of the ranges of the MERs ?

Each corner (defined by orthogonal lines through two consecutive points on the staircase) of S_i define two empty rectangles. We maintain the orientations, $O1$ and $O2$, at which these rectangles appear in S_i, triggered by an event as defined above, at the vertices in S_i adjacent to this corner according to some convention (we keep $O1$ with the rectangle that spans into the CCW group; $O2$ with the rectangle that spans into the CW group).

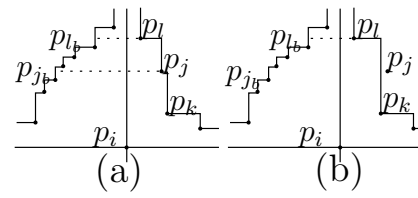


Fig. 6. Event: Disappearance of p_j.

Assume that the event is a "vertical" one, viz $E_v(p_j, p_k)$. We can classify this event according to one of the following cases:

New point on S_i: Let p_j be already on S_i, while the new point p_k comes on to S_i. This happens when p_k touches a line segment $e \in v_j$. Without loss of generality, we assume that $p_j \in S_{II_i}$ (see Fig. 5).
The rectangles $< p_i, next(p_m, S_{I_i}), p_m, p_j >$ become nonempty, where $p_m \in S_{I_i}$ and lie between p_{l_b} and p_{k_a} (see Fig. 5) on S_i. Therefore, for every point $p_m \in S_{I_i}$ that lie between p_{l_b} and p_{k_a}, the rectangle range is $< p_i, next(p_m, S_{I_i}), p_m, p_j, O1_{next(p_m, S_{I_i})}, \theta >$, where O_1 is the orientation at which the rectangle range appeared in S_i. The range of the rectangle $< p_i, p_{l_b}, p_l, p_j >$ is $< p_i, p_{l_b}, p_l, p_j, O2_{p_l}, \theta >$.
Similarly, we identify the ranges of the rectangles bounded by the points in S_{II_i} and S_{III_i} that are terminated by the event.
New rectangles $< p_i, next(p_m, S_{I_i}), p_m, p_k >$, where $p_m \in S_{I_i}$ lie between p_{l_b} and p_{k_a}, appear in S_i. So we set $O1_{next(p_m, S_{I_i})}$ to θ, for each p_m between p_{l_b} and p_{k_a}. Similarly, we set $O2_{p_l}$ to θ. We do the same for the new rectangles appearing between the points in S_{II_i} and S_{III_i}.

Disappearance of a point from S_i: Let p_j and p_k be adjacent points on S_{I_i} (see Fig. 6). Immediately, after the event, p_k dominates p_j. As in the previous case, the rectangles bounded by the point p_j are terminated. The new rectangles that appear are obtained by appropriately changing the bounding point p_j of these rectangles.

Point Migration: Let $i = k$ and $p_j \in S_{II_i}$. This means that this event has been triggered by the collinearity of the points p_i and p_j. Suppose that the point p_j migrates to S_{I_i} (see Fig. 7), appearing on the staircase immediately before and after the migration. In fact, the point migrates from the group it belongs to before the event to the group clockwise to it after the event, depending on whether the event is $E_v(p_j, p_i)$ or $E_h(p_j, p_i)$. As in the the previous case, again we can identify the ranges of rectangles terminated by this event. The point p_j may dominate a part of the stair case in S_{I_i}, and a part of the next layer in S_{II_i} (shown in dotted lines in the Fig. 7), which respectively disappear and appear in S_i immediately after the event. The MERs whose boundary contains the disappearing point are terminated. We update S_i accordingly.

Fourth type: The points p_j and p_k belong to S_{II_i} and S_{III_i} respectively. In this case, the topology of the staircase S_i does not change, but the ranges of the rectangles $< p_k, next(S_{III_i}, p_k), p_i, prev(S_{II_i}, p_j) >$ and $< p_j, p_k, p_i, prev(S_{II_i}, p_j) >$ are terminated and ranges due to new rectangles $< p_j, next(S_{III_i}, p_k), p_i, prev(S_{II_i}, p_j) >$ and $< p_k, next(S_{III_i}, p_k), p_i, p_j >$ are initiated (see Fig. 8).

Complexity of Updating S_i

Lemma 4. *Given a set of n points inside a rectangle R, the largest empty arbitrary oriented rectangle bounded by the point p_i can be computed in $O(n^3)$ time, using $O(n)$ space.*

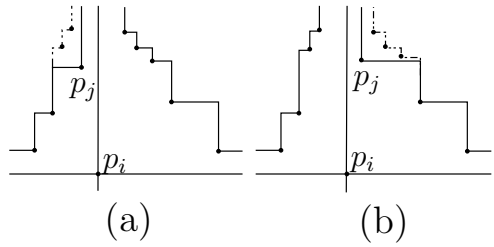


Fig. 7. Point p_j migrates from S_{II_i} to S_{I_i}.

Proof. We use a linked list to implement the staircases of S_i. To determine the event type and the location in S_i of the points involved in the event in constant time, we need an additional data structure: $Pointer_i[]$ to store cross pointers to S_i. If point p_j is in S_i, $Pointer_i[j]$ points to its location in S_i, otherwise its value is *nil*. The implementation of Q is standard.

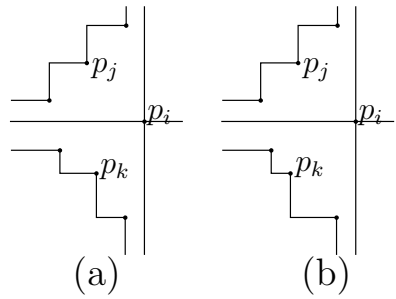


Fig. 8. Fourth type event.

The initialization of S_i and Q can be done respectively in time $O(n \log n)$ and $O(n^2 \log n)$. We analyse the complexity of each of the cases above.

Let the event be a "vertical" one: $E_v(p_j, p_k)$. We know that $j \neq k$. If $k \neq i$, $p_j \in S_i$, $p_k \notin S_i$, and p_k is on the part of v_j which is in S_i, then p_k is a new point on S_i. Using the $Pointer_i[j]$ and p_{j_b} the range of rectangles terminated by the event and updating of the orientations can be done in $O(k_1)$ time, where k_1 is number of rectangles terminated.

If p_j and p_k are two adjacent points in S_{I_i}, then one of them dominates the other after the event. Let p_k dominate p_j immediately after the event. As in the previous case, the range of rectangles terminated by the event and updating the orientations can be done in $O(k_2)$ time, where k_2 is number of rectangles terminated.

If $k \neq i$, $p_j, p_k \in S_i$, and p_j, p_k are in adjacent groups, we have the last case discussed above. The number of rectangles terminated by this event are two, and two new rectangles are started. Hence the update time is in $O(1)$.

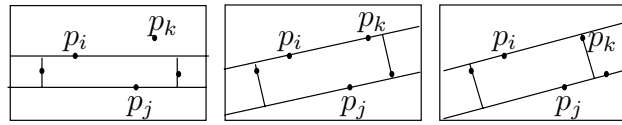


Fig. 9. First case: p_k is coming into the strip.

If $k = i$, p_j migrates to another group. Again, we can find the ranges of rectangles terminated by the event in time proportional to number of rectangles and update S_i in $O(n)$ time by recomputing S_i. For this, we need the list of points in sorted order (this is a function of the grid orientation θ). The sorted order can be maintained without any extra cost as follows. Initially, we sort the points along the x-coordinate. Whenever the event is a "vertical" one, we swap the positions of these points in the sorted list as these points are adjacent in the sorted list before and after the event.

Therefore, the time required to process an event is in $O(n)$. There are $O(n^2)$ events, and hence the lemma.

∎

3.4 Complexity of the Algorithm

A straightforward extension of the algorithm for all n stair-cases takes $O(n^4)$ time. The following observations, however, allow us to improve this bound.

Observation 3.4 *An MER appears in at most four staircases.*

Observation 3.5 *There are $n-1$ migrations for each staircase.*

Lemma 5. *Given a rectangle R and a set P of n points in it. The largest empty arbitrary oriented rectangle in side R can be computed in $O(n^3 + s)$ time using $O(n^2)$ space, where s is the total number of rectangle ranges.*

Proof. The time required to process all migrations are in $O(n^3)$ as there are $O(n^2)$ migrations. For all other events, the time taken is in $O(n^3 + s)$, since, for each event we are looking at every staircase for terminated rectangle ranges and each MER appears in at most four staircases, where s is the total number of rectangle ranges. Hence the lemma. ∎

Estimate of s The MER bounded by the horizontal (similarly, vertical) lines h_i (v_i) and h_j (v_j) passing through a pair (p_i, p_j) of points is uniquely defined, if it exists. We estimate the number of MERs bounded by p_i, p_j, h_i, and h_j. Similarly, we can estimate the number of MERs bounded in vertical strip formed by v_i and v_j and the points p_i and p_j.

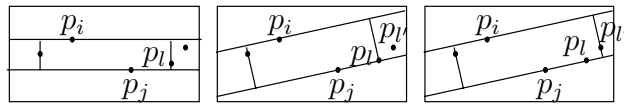


Fig. 10. Second case: p_l is going out of the strip.

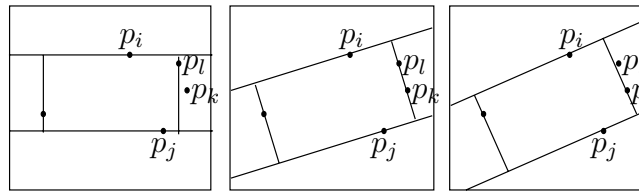


Fig. 11. Third case: p_k dominates p_l.

Let r be the MER defined by the horizontal strip formed by h_i and h_j and bounded by p_i and p_j. Let p_l and p_r be the points respectively on the left and right edges of r. Note that r also defined by the strip formed by v_l and v_r. Rotate the line h_i and h_j respectively about the points p_i and p_j. The rectangle-range bounded by p_i and p_j is terminated when one of the following happens:

1. Either the top edge or the bottom edge of r touches a new point (see Fig. 9).
2. Either p_l or p_r goes out of the strip (see Fig. 10).
3. Some other point in the strip dominates either p_l or p_r (see Fig. 11).

Whenever, the rectangle range is terminated, either a new rectangle range is started or the rectangle bounded by p_i and p_j is not empty. The first two cases arise when a horizontal line coincides either with h_i or h_j. The third case occurs when a vertical line coincides with either v_l or v_r. We charge the coincidence of a horizontal line (i.e. first two cases) to the pair (p_i, p_j) and their horizontal strip, and a vertical line (i.e. third case) to the pair (p_l, p_r) and their vertical strip. The number of times first two cases can happen is in $O(n)$. The total number of rectangle ranges are in $O(n^3)$, since there are $O(n^2)$ pair of points. This proves the following theorem.

Theorem 1. *Given a rectangle R and a set P of n points in it. The largest empty arbitrary oriented rectangle in side R can be computed in $O(n^3)$ time using $O(n^2)$ space.*

4 Implementation

The algorithm has been implemented in JAVA by Mr. Hong-tao Zhang, a graduate student of the first author. It works successfully on random sets of input points.

5 Conclusion

In this paper we have presented an $O(n^3)$ time algorithm for finding an arbitrary oriented largest empty rectangle. The algorithm is based on rotating the grid formed by the orthogonal lines passing through each point and enumerating all rectangle ranges. It would be interesting to design an algorithm without enumerating all rectangle ranges.

This method can be extended to the following restricted problem in three dimensional space: finding the largest empty volume box, which has a pair of faces perpendicular to one of the axes. This problem can be solved by reducing it three two-dimensional problems, one for each of the three axes. Consider the subproblem in which two faces are always orthogonal to the z-axis: in this problem we keep a pair of faces perpendicular to z-axis stationary and rotate the other two pair of faces about the z-axis. This latter problem is exactly what we ahve discussed in this paper.

This method does not directly generalize to the more difficult case of finding an arbitrary oriented box of largest measure, because there are an infinite ways of rotating these planes. So, it would be interesting to explore an alternative approach to this problem.

References

[AS87] A. Aggarwal and Subhash Suri. Fast algorithms for computing the largest empty rectangle. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 278–290, 1987.

[AS98] Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surveys*, 30:412–458, 1998.

[CDL86] Bernard Chazelle, R. L. Drysdale, III, and D. T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15:300–315, 1986.

[NHL84] A. Naamad, W.-L. Hsu, and D. T. Lee. On the maximum empty rectangle problem. *Discrete Appl. Math.*, 8:267–277, 1984.

[Orl90] M. Orlowski. A new algorithm for the largest empty rectangle problem. *Algorithmica*, 5:65–73, 1990.

[PS90] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, October 1990.

An Approximate Morphing between Polylines

Sergei Bespamyatnikh

Department of Computer Science, University of Texas at Dallas,
Box 830688, Richardson, TX 75083, USA
besp@utdallas.edu

Abstract. We address the problem of continuously transforming or morphing one simple polyline into another so that every point p of the initial polyline moves to a point q of the final polyline using the geodesic shortest path from p to q. We optimize the width of the morphing, that is, the longest geodesic path between p and q. We present a linear-time algorithm for finding a morphing with width guaranteed to be at most 1.618 times the minimum width of a morphing. This improves the previous algorithm [9] by a factor of $\log n$. We also develop a linear-time algorithm for computing a medial axis separator.

1 Introduction

The problems of *morphing* or continuously transforming one shape into another have been the subject of a recent investigation [1, 2, 6, 9, 12]. Morphing algorithms have numerous uses in shape interpolation, animation, video compression [1, 3, 9, 12, 17, 19]. In general there are various ways [2, 4, 9] to interpolate between two shapes and there are different criteria for the quality of a morph and notions of optimality. We consider the following morphing problem introduced in [9]. Let α and β be two shapes which are polylines. A polyline is a finite sequence of segments in the plane that are connected into a simple curve. We want to produce a continuous transformation of the polyline α to the polyline β. Rendering this transformation is an animation problem.

Guibas *et al.* [12] considered the problem when α and β are *parallel* polygons with the same number of vertices, i.e., polygons with the same sequence of angles. They showed that any two parallel polygons can be morphed into one another such that every interpolating polygon is also parallel to the initial and final polygons. They also showed that the morphing transformation can be computed in $O(n \log n)$ time.

Recently Efrat *et al.* [9] considered morphing between polylines such that the interpolating polyline is always between bounding polylines and the objective is to minimize the longest path made by a point moving from one polyline to another. The minimum value of the longest shortest path is called the *morphing width*. They obtained an algorithm with $O(n^2 \log^2 n)$ running time and $O(n^2)$ space. Bespamyatnikh [6] gave an algorithm with $O(n^2)$ running time and $O(n)$ space.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 807–816, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Efrat *et al.* [9] also presented an approximation algorithm for computing a morphing of width at most two times the morphing width, i.e. 2-approximation of the morphing width. The algorithm runs in $O(n \log n)$ time. We show that the factor 2 is the tight bound of their algorithm and present a linear time algorithm that guarantees a better approximation. We prove that the approximation factor of our algorithm is at most the golden ratio $(1+\sqrt{5})/2 \approx 1.61803$. The algorithm is based on a *medial axis separator* which can be extracted from the medial axis [7]. The algorithm for computing the medial axis [7] in a simple polygon is quite complicated. We show that the medial axis separator can be computed in linear time using a simple algorithm.

Another variant of the morphing width under a slightly different distance function, was previously investigated in [10]. Alt and Godau [2] considered the problem of computing the Fréchet distance between two polygonal curves. They show that the Fréchet distance can be computed in $O(mn)$ time. The Fréchet distance can be used to define a morph from one polyline to another. The Fréchet distance can be viewed as a simpler function than the width between polylines, which is minimized in our algorithm.

Approximate shape matching problems received attention since many algorithms for finding exact matches have high complexity. Heffernan and Schirra [14] gave $O(n^{2.5} polylogn)$ algorithm for approximate congruence problem. Recently, Efrat *et al.* [11] improved the running time to $O(n^{1.5} \log n)$ using approximate nearest neighbor queries. Indyk and Venkatasubramanian [16] considered a generalized bottleneck distance and obtained approximation algorithms.

Alt *et al.* [5] considered κ-*straight* curves such that the arclength between any two points on the curve is at most κ times their Euclidean distance. They show that the Fréchet distance between κ-straight curves is at most $1+\kappa$ times their Hausdorff distance. It implies that $(1+\kappa)$-approximation of the Fréchet distance between these curves can be computed in $O(m+n)\log(m+n))$ time.

The paper is organized as follows. In Section 2 we introduce notation and specify the problem. In Section 3 we give an approximation algorithm and prove the tight bound on approximation factor. In Section 4 we describe an algorithm for computing the medial axis separator.

2 Geometric Preliminaries

The initial and final polylines α and β can be represented as parameterized curves $f_\alpha, f_\beta : [0,1] \to \mathbb{R}^2$, i.e., the polyline α is the set of points $\{f_\alpha(u) \mid u \in [0,1]\}$ and the polyline β is the set of points $\{f_\beta(u) \mid u \in [0,1]\}$. We assume that α and β have common endpoints $s_0 = f_\alpha(0) = f_\beta(0)$ and $s_1 = f_\alpha(1) = f_\beta(1)$. The polylines α and β are simple (without self-intersections) and oriented, i.e., they are oriented from p_0 to p_1. We assume that the parameterizations f_α and f_β are *monotone*, i.e., a point $f_\alpha(u_1)$ lies in the subpolyline of α from p_0 to $f_\alpha(u_2)$ if $0 \le u_1 \le u_2 \le 1$ (similarly for β).

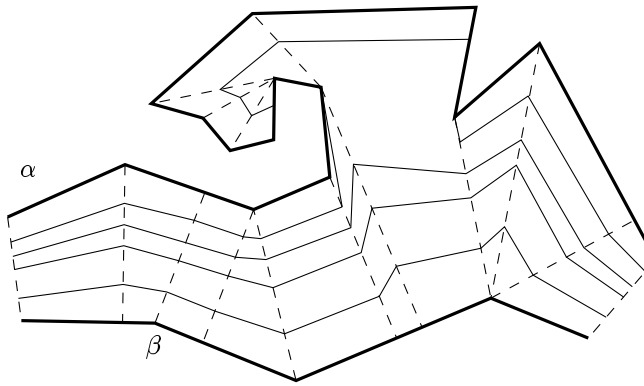


Fig. 1. Morphing between two polylines α and β.

A *morphing* is defined by a function $F : [0,1]^2 \to \mathbb{R}^2$ of two parameters $F(u,t)$ such that $F(u,0) = f_\alpha(u)$ and $F(u,1) = f_\beta(u)$ for any $u \in [0,1]$[1]. The parameter t corresponds to time and, at any moment $t \in [0,1]$, the morphing function F defines the *interpolating polyline* $\gamma(t)$ which is the set of points $\{F(u,t)|u \in [0,1]\}$. Fixing another parameter u we obtain the *trace* of a point that moves from $f_\alpha(u)$ to $f_\beta(u)$. The points $f_\alpha(u)$ and $f_\beta(u)$ are called *matching points*. Every morphing function F induces the *matching* $M(F) = \{(f_\alpha(u), f_\beta(u)) \mid u \in [0,1]\}$.

We consider the morphing problem which is defined as follows. We assume that the polylines α and β are disjoint and, furthermore, the segments of α and β and two segments $[f_\alpha(0), f_\beta(0)]$ and $[f_\alpha(1), f_\beta(1)]$ form a simple polygon P, see Fig. 1. The morphing function F should satisfy the following property.

Trace property. The trace of each point is the shortest path in the polygon P from $f_\alpha(u)$ to $f_\beta(u)$ for some $u \in [0,1]$.

A morphing function satisfying the trace property is determined by the matching $M(F)$. For two points $p, q \in P$, let $\pi(p,q)$ be the shortest path from p to q that lies inside the polygon P. Let $d_P(p,q)$ denote the length of the path $\pi(p,q)$. A *width* of a morph F is defined as

$$W(F) = W(f_\alpha, f_\beta) = \max_{u \in [0,1]} \{d_P(f_\alpha(u), f_\beta(u))\}.$$

The width can be expressed in terms of matching

$$W(M) = \max_{(a,b) \in M} \{d_P(a,b)\}.$$

We want to find a morphing function satisfying the trace property and minimizing the longest trace that is called the *morphing width* [9]

$$\mathcal{W}(\alpha, \beta) = \min_{f_\alpha, f_\beta} W(f_\alpha, f_\beta).$$

[1] Note that the parametrizations f_α and f_β are not included in the input of the problem since they are induced by the output morphing function F.

Note that the problem can be stated in terms of parametrizations f_α and f_β since the morphing function (and the matching) can be derived from them[2].

Bespamyatnikh [6] defined a *matching skeleton* of a morphing that characterizes an optimal morphing. Let F be a morphing function and let a be a vertex of the polyline α. We define a *match* of a as

$$M(a) = \{b \in \beta \mid (a, b) \in M(F)\}.$$

The match of a is a subpolyline of β. The match $M(b)$ of a vertex $b \in \beta$ is defined symmetrically. A match $(a, b) \in M(F)$ is *critical* if either a is a vertex of α and b is an endpoint of $M(a)$, or b is a vertex of β and a is an endpoint of $M(b)$. Let $M_c(F)$ be the set of critical matches of F. The *skeleton* $S(F)$ of a morphing function F is the union of the shortest paths of its critical matches

$$S(F) = \cup_{(a,b)\in M_c(F)} \pi(a, b).$$

The morphing F is determined by its skeleton. The skeleton has linear complexity because it is a planar graph whose vertices are the vertices of P. We assume that the output of a morphing algorithm is the skeleton $S(F)$. For example, the dashed segments in Fig. 1 form the skeleton of an optimal morph.

Theorem 1 ([6]). *The width of an optimal morphing F is the length of a shortest path in its skeleton.*

Let a be a point of α and b_1 and b_2 be two points of an edge of β. The shortest path in P from a to any point $b \in [b_1, b_2]$ has length at most $\max(d_P(a, b_1), d_P(a, b_2))$.

Lemma 1 ([6]). *There is a minimum width morph such that each critical match either*

(i) *is a pair of vertices from α and β, or*
(ii) *corresponds to the shortest distance from a vertex of a polyline to an edge of another polyline.*

Corollary 1 ([6]). *The minimum width $W(M(F))$ over all morphing functions F is a shortest distance in P either between two vertices of P (one from α and another from β) or between a vertex and an edge of P (vertex from α (resp. β) and edge from β (resp. α)).*

[2] We even can assume that one of the parametrizations is fixed. For example, f_β can be fixed to be the *arc-length parametrization*, i.e., $f_\beta(u)$ is the point of β so that the length of the subpolyline of β from $f_\beta(0)$ to $f_\beta(u)$ is $u|\beta|$ where $u \in [0, 1]$ and $|\beta|$ is the length of β. Then the problem is to find a parametrization f_α minimizing the width $W(f_\alpha, f_\beta)$.

3 Approximation Algorithm

First we briefly describe an approximation algorithm by Efrat *et al.* [9]. Consider the shortest path $\sigma = \pi_P(s_0, s_1)$ separating two polylines α and β in P. The key idea is to construct a morph by first transforming α into σ and then transforming σ into β. The first transformation is defined by matching a point a of α to its geodesic neighbor on σ, i.e. a point $c \in \sigma$ minimizing $d_P(a, c)$. Every point of σ matches at least one point of α. Similarly, the second transformation matches a point b of β to its geodesic neighbor on σ. Combining two transformations we obtain a morph between α and β. Efrat *et al.* [9] proved that the morph has a width at most twice the optimal width. We show that this bound is tight.

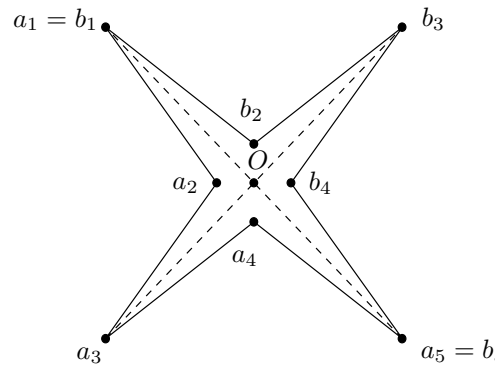


Fig. 2. $(2 - \varepsilon)$-approximation.

Observation 1 *For any $\varepsilon > 0$, there is an example of two polylines α and β so that the approximate morphing generated by the shortest path $\pi(a, b)$ has length at least $(2 - \varepsilon)\mathcal{W}(\alpha, \beta)$.*

Our main idea is to construct a morph using another polyline separating α and β. Let $\mathcal{M}$ be the medial axis. $\mathcal{M}$ is a tree embedded in the plane so that each edge is either a straight line segment or a parabolic edge. There is a unique path in $\mathcal{M}$ from s_0 to s_1. We call it a *medial axis separator* and denote by $\pi_{\mathcal{M}}$. Consider an edge e of $\pi_{\mathcal{M}}$. e separates two Voronoi cells. Suppose that e is a line segment, see Fig. 3 (a). Then two Voronoi cells correspond to two edges e_1 and e_2 of the polylines α and β, respectively. Let p be an interior point of e. There are unique points $p_1 \in e_1$ and $p_2 \in e_2$ such that the shortest path from p to $e_i, i = 1$ ends at p_i. We match the points p_1 and p_2, i.e., $(p_1, p_2) \in M(F)$.

If e is a the parabolic edge, then one of the Voronoi cells separated by e is a vertex of P, see Fig. 3 (b). Without loss of generality we assume that e separates a vertex p_1 of α and an edge e_2 of β. Let p be an interior point of e. Again, there is a unique point $p_2 \in e_2$ such that the shortest path from p to e_2 ends at p_2

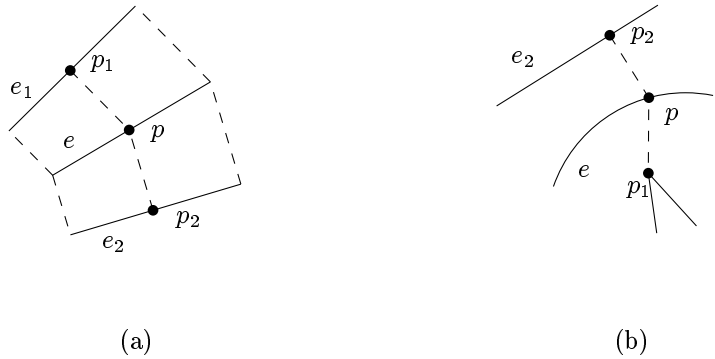


Fig. 3. Matches induced by interior point of (a) line segment and (b) parabolic edge.

(it is just the segment pp_2). We match the points p_1 and p_2, $(p_1, p_2) \in M(F)$. The matches induced by all interior points of edges of $\pi_{\mathcal{M}}$ do not determine a match of a morphing, in general, since some vertices and even edges of two polylines may match a vertex of the medial axis separator, see Fig. 4. However, the following lemma shows that the matches defined so far are better than the matches produced by the shortest path $\pi(a, b)$ [9] (the shortest path of such a match can have length $2\mathcal{W}(\alpha, \beta)$ by Observation 1).

Lemma 2. *Let (p_1, p_2) be the match produced by an interior point of an edge in π_M. The length of the shortest path $\pi(p_1, p_2)$ does not exceed the optimal width $\mathcal{W}(\alpha, \beta)$.*

We show how to complete the matching. Consider a vertex p of the medial axis separator. Let r be the distance from p and the boundary of P. Let A and B be the set of points of α and β, respectively, at distance r from p. Let $\alpha' = a_i a_{i'}$ and $\beta' = b_j b_{j'}$ be the shortest subpolylines of α and β containing A and B, respectively. A simple way of transforming α' to β' is by matching a_i to all the points of β' and matching all the points of α' to $b_{j'}$, see Fig. 5. We call the matching M induced by $\pi_{\mathcal{M}}$ a *medial axis matching* and we call the corresponding morph a *medial axis morphing*.

Theorem 2. *The medial axis morphing has width at most the golden ratio $(1 + \sqrt{5})/2 \approx 1.61803$ times the optimal width $\mathcal{W}(\alpha, \beta)$. The above algorithm computes the medial axis morphing in linear time.*

Proof. The shortest paths of the matches produced by the edges of the medial axis separator have lengths within $\mathcal{W}(\alpha, \beta)$ by Lemma 2. We focus on the matches produced by the vertices of $\pi_{\mathcal{M}}$. Let p, r, α', and β' be as defined above. Let $b_{\max}$ be the point of β' maximizing $d_P(a_1, b), b \in \beta'$. Similarly, let $a_{\max}$ be the point of α' maximizing $d_P(a, b_2), a \in \alpha'$. Without loss of generality $d_P(a_1, b_{\max}) \geq d_P(a_{\max}, b_2)$. It suffices to prove that $d_P(a_1, b_{\max}) \leq$

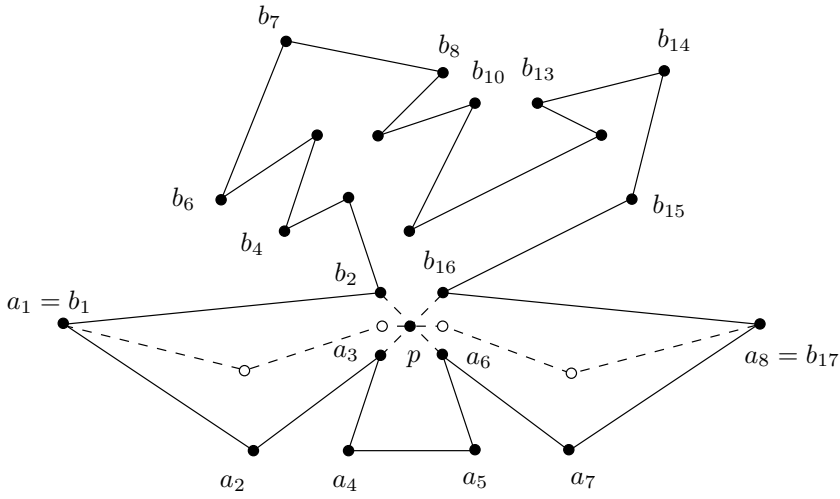


Fig. 4. Point p of medial axis matches vertices $a_3, \ldots, a_6$ of α and vertices $b_2, \ldots, b_{16}$ of β.

$c\mathcal{W}(\alpha, \beta)$ where $c = (1+\sqrt{5})/2$. We consider two cases depending on the distance $d_P(p, b_{\max})$.

Case 1. $d_P(p, b_{\max}) \geq \sqrt{5}r$.

In this case

$$d_P(a_1, b_{\max}) \leq d_P(a_1, p) + d_P(p, b_{\max}) \leq r + \mathcal{W}(\alpha, \beta) \leq \frac{\mathcal{W}(\alpha, \beta)}{\sqrt{5}} + \mathcal{W}(\alpha, \beta) = c\mathcal{W}(\alpha, \beta).$$

The first inequality is the triangle inequality for the metric $d_P()$. The second inequality follows since $d_P(b_{max}, p)$ is at least the geodesic distance from $b_{\max}$ to a point of α that is at least $\mathcal{W}(\alpha, \beta)$. The third inequality is the case assumption.

Case 2. $d_P(p, b_{\max}) < \sqrt{5}r$.

In this case

$$d_P(a_1, b_{\max}) \leq d_P(a_1, p) + d_P(p, b_{\max}) \leq r + \sqrt{5}r \leq \frac{1+\sqrt{5}}{2}\mathcal{W}(\alpha, \beta) = c\mathcal{W}(\alpha, \beta).$$

The last inequality follows since $d_P(a_1, p)$ is at least the geodesic distance from a_1 to a point of β that is at least $\mathcal{W}(\alpha, \beta)$. The approximation bound follows.

We analyze the running time. The medial axis can be constructed in linear time using an algorithm of Chin *et al.* [7]. The medial axis separator and all the matches can be computed in $O(n)$ time where n is the total number of vertices of α and β. The subpolylines α' and β' can be found in time proportional to their complexity. Note that the shortest paths $d_P(a_i, b_j), d_P(a_i, b_{j'})$, and $d_P(a_{i'}, b_{j'})$ are just segments. The shortest paths matching a_i and β' form the *shortest path tree* [13, 15] in the polygon $a_i b_j b_{j+1} \ldots b_{j'}$. It can be computed in linear time. Similarly, we compute the shortest tree in the polygon $b_{j'} a_i a_{i+1} \ldots a_{i'}$. These paths form the skeleton of the matching.

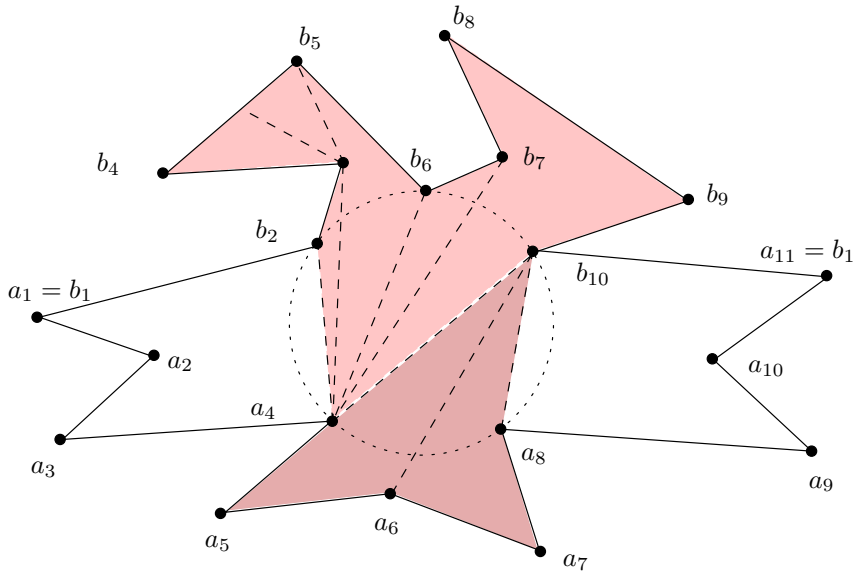


Fig. 5. Completing the match. $\alpha' = a_4 \dots a_8$ and $\beta' = b_2 \dots b_{10}$.

The upper bound in Theorem 2 is actually tight.

Lemma 3. *For any $\varepsilon > 0$, there is an example of two polylines α and β such that the medial axis morphing has width at least $((1+\sqrt{5})/2 - \varepsilon)\mathcal{W}(\alpha, \beta)$.*

4 Medial Axis Separator

The only part of our approximation algorithm that is difficult to implement is the computation of the medial axis [7]. Alternatively the medial axis can be constructed using either an algorithm of Lee [18] with $O(n \log n)$ running time or a randomized algorithm of Devillers [8] with $O(n \log^* n)$ expected running time. We show that the medial axis separator can be computed in linear time using a simple algorithm avoiding the medial axis computation.

Let $a_1, a_2, \dots, a_k$ be the vertices of α and $b_1, \dots, b_m$ be the vertices of β. Clearly, the medial axis separator $\pi_{\mathcal{M}}$ starts at a_1 and the first edge is a part of the bisector of a_1a_2 and b_1b_2. The idea is that we can construct $\pi_{\mathcal{M}}$ edge-by-edge by tracing the polylines α and β until we reach $a_k = b_m$. Let e be the current edge of π_M (more precisely, e is an edge of the Voronoi diagram for vertices/edges of α and β already visited.) Let p be the point traversing e (p starts at a_1 and ends at a_k). There are three cases depending on the type of the current edge of $\pi_{\mathcal{M}}$.

Edge-edge. e is the bisector of two edges $a_{i-1}a_i \in \alpha$ and $b_{j-1}b_j \in \beta$, see Fig. 6(a). The point p is equidistant from two points $a \in a_{i-1}a_i$ and $b \in b_{j-1}b_j$. p can move along e until either $a = a_i$ or $b = b_j$ (this may happen simultaneously).

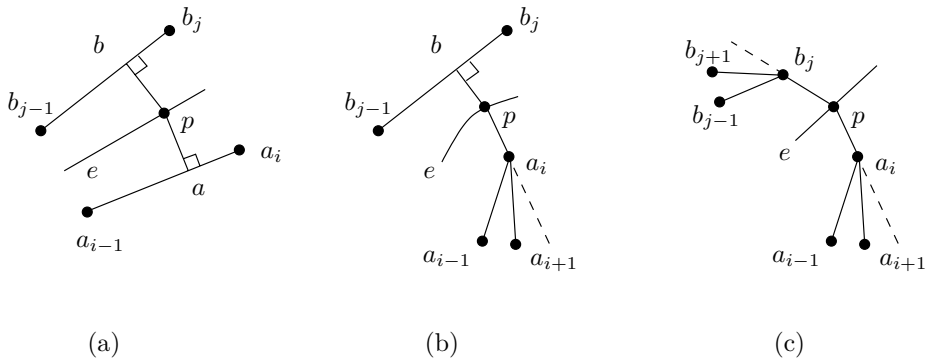


Fig. 6. Three cases of traversing $\pi_{\mathcal{M}}$.

Edge-Vertex. e is the bisector of a vertex $a_i \in \alpha$ and an edge $b_{j-1}b_j \in \beta$, see Fig. 6(b). The edge e is a parabolic piece. We assume that both points a_{i_1} and a_{i+1} (note that $i \notin \{1, k\}$) lie on the same side of the line pa_i. The point p is equidistant from a point $a \in a_{i-1}a_i$ and b_j. p can move along e until either $b = b_j$ or p, a_i, and a_{i+1} are collinear. In the first case we follow *vertex-vertex* case described below. In the second case, the point p is no longer on the medial axis separator since the edge a_ia_{i+1} is added. We apply the following *backtracking* technique. In $O(1)$ time we check if there is a point q in e that is equidistant from $b_{j-1}b_j \in \beta$ and a_ia_{i+1}. The point q is a "witness" that the part qp of e must be discarded. We put the point p at the position of q and continue. If q does not exist then whole edge e must be discarded and the same procedure can be applied to the previous edge. This process will stop since the point a_1 survives.

Vertex-Vertex. e is the bisector of two vertices $a_i \in \alpha$ and $b_j \in \beta$, see Fig. 6(c). We assume that all the four points $a_{i-1}, a_{i+1}, b_{j-1}$, and b_{j+1} are in the wedge a_ipb_j (the same wedge where the part of e before p is located). The point p travels along e until either p, a_i, a_{i+1} or p, b_j, b_{j+1} are collinear. In either case we insert a new edge of α or β to a list of traversed edges. Note that adding new edge may need to apply the backtracking.

Theorem 3. *The above algorithm computes the medial axis separator in a simple polygon in linear time.*

5 Conclusion

In this paper, we study the problem of approximate morphing from one polyline to another minimizing the morphing width. We prove that the medial axis separator provides both better approximation and faster algorithm than the previous result. The algorithm for computing the medial axis separator is of independent interest. It would be interesting to find a fast (with $o(mn)$ runtime) algorithm for computing an approximate Fréchet distance between arbitrary

polylines. The other future directions for study are (i) extensions to higher dimensions, (ii) various shapes (points, segments, polygons, etc.), and (iii) other morphing objectives.

References

[1] O. Aichholzer, H. Alt, and G. Rote. Matching shapes with a reference point. *Int. J. Comput. Geom. Appl.*, 7:349–363, 1997.

[2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[3] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, 3:237–256, 1988.

[4] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pp. 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[5] H. Alt, C. Knauer, and C. Wenk. Bounding the Fréchet distance by the Hausdorff distance. In *Proc. 17th European Workshop Comput. Geom.*, pp. 166–169, 2001.

[6] S. Bespamyatnikh. An optimal morphing between polylines. *Int. J. Comput. Geom. Appl.*, 12(3):217–228, 2002.

[7] F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete Comput. Geom.*, 21(3):405–420, 1999.

[8] O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Int. J. Comput. Geom. Appl.*, 2(1):97–111, 1992.

[9] A. Efrat, L. J. Guibas, S. Har-Peled, and T. Murali. Morphing between polylines. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pp. 680–689, 2001.

[10] A. Efrat, L. Guibas, S. Har-Peled, , D. Lin, J. Mitchell, and T. Murali. Sweeping simple polygons with a chain of guards. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pp. 927–936, 2000.

[11] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 2001.

[12] L. Guibas, J. Hershberger, and S. Suri. Morphing simple polygons. *Discrete Comput. Geom.*, 24:1–34, 2000.

[13] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[14] P. J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. *Comput. Geom. Theory Appl.*, 4:137–156, 1994.

[15] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.*, 4:63–98, 1994.

[16] P. Indyk and S. Venkatasubramanian. Approximate congruence in nearly-linear time. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pp. 354–360, 2000.

[17] A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Computer Graphics Proceedings*, pp. 343–350. ACM Siggraph, 1999.

[18] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-4(4):363–369, 1982.

[19] S. Venkatasubramanian. *Geometric Shape Matching and Drug Design*. Ph.D. thesis, Stanford University, 1999.

Efficient Proximity Search for 3-D Cuboids

Jie Gao[1]* and Rakesh Gupta[2]

[1] Department of Computer Science, Stanford University,
Stanford, CA 94305, USA
jgao@cs.stanford.edu
[2] Honda Research Institute USA, Inc.
800 California Street, Suite 300,
Mountain View, CA 94041, USA
rgupta@honda-ri.com

Abstract. *In this paper, we give the definition for the voronoi diagram and its dual graph – Delaunay triangulation for 3D cuboids. We prove properties of the 3D Delaunay triangulation, and provide algorithms to construct and update the Delaunay triangulation. The Delaunay triangulation data structure is used to perform proximity searches for both static and kinetic cases. We describe experimental results that show how the Delaunay triangulation is used on a mobile robot to model, understand and reason about the spatial information of the environment.*

1 Introduction

In this paper, we describe the Delaunay triangulation spatial data structure for 3D cuboids for a mobile robot, that allows it to do simple reasoning such as spatial relationship between two objects, or finding objects in the neighborhood of a given object. Specifically, given a query volume of an arbitrary shape, the goal is to find all the cuboids that intersect with the query volume. When the query volume varies with time, we want to keep track of all the intersected cuboids. We address the static and kinetic proximity search problem among a set of n cuboids in 3D.

Information about the environment is obtained and accumulated as the robot explores the world. Our mobile robot needs to build a 3D spatial model according to the partial information obtained in sequential frames, and update the model as more information comes in. Moreover, we need to support efficient queries in the egocentic frame of reference, where the environment is modeled relative to the robot. Typical approaches for proximity search group objects by proximity into hierarchies, and store them in another data structure such as a *kd*-tree or R-tree [13]. These conventional data structures do not work well for our purpose. First, *kd*-tree or quad-tree, do not support efficient sequential insertion and deletion. R-tree may generate very unbalanced tree structure when the data comes in sequentially. Second, *kd*-tree and R-tree support only standard query

* This work was done when the author was working at Honda Research Institute USA, Inc.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 817–826, 2003.
© Springer-Verlag Berlin Heidelberg 2003

volumes, like boxes in 3D and rectangles in 2D. For our purpose, the query volume can be what the robot is currently viewing or has seen before, which might be a cone or some other flexible volume. In addition, proximity search in *kd*-tree or R-tree is not fast enough when the number of objects is large [4,1].

The Voronoi diagram and its dual Delaunay triangulation [2,6,4] provide a natural framework for reasoning about objects in space. Voronoi model maps well to natural language, and how humans judge and express measurement, and is able to provide an answer without referring to metrical details. Delaunay triangulation representation can be built efficiently using a randomized incremental algorithm. This representation is easy to update as more information about objects comes in. Another nice property of Delaunay triangulation is that it allows efficient proximity searches, especially when the robot is moving around.

The Voronoi diagram [2] of 3D points have been studied extensively. Proximity search for points can be solved in $O(\log n + k)$ time, where k is the number of points inside the query volume and n is the number of points. For convex polygons in the plane, McAllister et. al. defined a compact piecewise-linear Voronoi diagram [12]. Govrilova studied the generalized Voronoi diagram on weighted sites under Euclidean, power, Manhattan and supremum metrics in 2 and higher dimensions [8]. She also applied the generalized Voronoi diagram on collision detection and proximity search.

In this paper we extend the Voronoi diagram and Delaunay triangulation to 3D cuboids. We gives definitions of Voronoi diagram and Delaunay triangulation for 3D convex objects in section 2. We also prove the local property of Delaunay triangulation. Section 3 shows how proximity queries can be performed using Delaunay triangulation, for both static and kinetic cases. We implemented the algorithm in the scene understanding for mobile robots. Details of the construction of the Delaunay triangulation is discussed in section 4. Section 5 uses experiments to show how the Delaunay triangulation helps the mobile robot in spatial reasoning about the scene.

2 Definition

We represent objects using polytopes. Suppose we have a set of disjoint polytopes $S = (P_1, P_2, ...P_n)$. A distance function δ between two points can be extended to a point a and a polytope P in R^3 by setting $\delta(a, P) = \min_{q \in P} \delta(a, q)$. For any point a outside a polytope P, there is a unique point p on P realizing the distance $\delta(a, P)$. We say line segment ap is a *spoke* and p is the *attachment point*. Obviously $\delta(a, P) = \delta(a, p)$. For two disjoint polytopes P_1 and P_2, the bisector $\pi(P_1, P_2)$ between them is defined to be the locus of points that are equidistant from them. Three polytopes are called collinear if there is a plane tangent to them simultaneously. Four polytopes are called cocircular if there is a ball tangent to them simultaneously.

2.1 Voronoi Diagram

The generalized Voronoi diagram $V(S)$ for polytopes is defined similarly as for points. We partition the free space into a number of *Voronoi cells* according to the nearest object. All the points in one Voronoi cell have the same nearest object. We denote the Voronoi cell for polytope P_i as V_i. A *Voronoi vertex* of four polytopes P_1, P_2, P_3, P_4, is a point v that is equidistant to all of them. There is a sphere centered at v that is tangent to all the four polytopes called the *Voronoi sphere*. The Voronoi sphere of a valid Voronoi vertex is free of any objects. In another point of view, v is also the intersection point of bisectors $\pi(P_i, P_j), i \neq j$, $1 \leq i, j \leq 4$. The Voronoi vertex defined by four points uniquely exists. The Voronoi cells for points are convex polytopes. However, for four polytopes, there could be zero or more than one Voronoi vertices, the Voronoi cells for polytopes may not be as regular as convex polytopes. We will talk about the Voronoi vertex computation in later sections.

We say that a set X is *star-shaped* with respect to a convex polytope P, if $P \subseteq X$ and every spoke $spoke(a, P)$ is contained in X, with $a \in X$.

Lemma 1. *The Voronoi cell V is star-shaped with respect to P.*

Proof. Firstly $P \subseteq V$. We only need to prove that if the nearest neighbor of a point a is P, then for any point q on the spoke $spoke(a, P)$, its nearest neighbor is P, too. We draw a sphere S_a centered at a, then S_a is free of any objects and S_a is tangent with the object P, with the tangent point p. Draw a sphere S_q centered at q with radius qp, see Figure 1. S_q is fully contained in S_p, so it is also outside any other objects. Then the nearest object of q must be P.

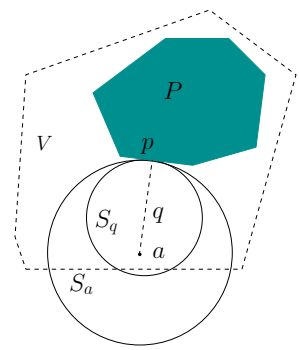


Fig. 1. The Voronoi cell is star-shaped

2.2 Delaunay Triangulation

The Delaunay triangulation $T(S)$ is the dual graph of the Voronoi diagram $V(S)$. For each pair of adjacent Voronoi cells, we connect the corresponding centroids of the cuboids and get the Delaunay triangulation.

For any four polytopes P_1, P_2, P_3, P_4, if there is a corresponding Voronoi vertex defined by them, then the Voronoi cells V_i and V_j $(i \neq j)$ must be adjacent. We define a *junction tetrahedron* corresponding to each Voronoi vertex. The vertices of the junction tetrahedron are the tangent points of each cuboid with the Voronoi sphere. It is easy to see that the circumsphere of the junction tetrahedron is the Voronoi sphere. So the junction tetrahedron is outside any of the objects.

Two Voronoi vertices are *adjacent* if they are incident to three common objects. We also define the corresponding junction tetrahedrons as adjacent. We define the neighboring polytopes for a junction tetrahedron to be the polytopes incident to it or to one of its adjacent junction tetrahedrons.

The Delaunay triangulation can be built from the Voronoi diagram, by connecting two objects if there exists a Voronoi vertex incident to both of them. The worse-case complexity of the Delaunay triangulation for 3D cuboids can be $\Theta(n^2)$. But for all practical purposes, three-dimensional Delaunay triangulation is observed to have linear complexity [7].

It is well known that the Delaunay triangulation for points has a local property. Specifically, if the circumcircle of a triangle doesn't contain any of its neighbors, we define this triangle to be *locally Delaunay.* The local property says that the triangulation is proved to be a Delaunay triangulation if all the triangles are locally Delaunay. It is also extendable to our case. We say that one junction tetrahedron is locally Delaunay if its circumsphere doesn't intersect with any of its neighboring polytopes. Clearly, if the triangulation is a Delaunay triangulation, all the junction tetrahedrons are Delaunay tetrahedrons. We will show that vice versa is also true. This property can be used in the incremental construction and dynamic update of the Delaunay triangulation.

To prove the local property, we first state some properties of the local Delaunay triangulation. Define $\widetilde{f}$ the sector of the sphere chopped off by the face f of a junction tetrahedron T.

Lemma 2. *If the junction tetrahedrons of the triangulation have the local Delaunay property, f_1 is a face of a tetrahedron T_1, f_2 is a face of a tetrahrdron T_2, T_1 and T_2 are adjacent, f_1 and f_2 are incident to the same three objects, then $\widetilde{f}_1 \bigcap \widetilde{f}_2$ are free of any objects.*

Proof. By contradiction, if there is a point p inside some object P, and p is inside $\widetilde{f}_1 \bigcap \widetilde{f}_2$. So there must exist a tetrahedron of P, P_1, P_2, P_3, with P_1, P_2, P_3 as the tangent objects of both f_1 and f_2. This contradicts with the local Delaunay property.

Theorem 1. *If all the junction tetrahedrons are locally Delaunay, then the triangulation $T(S)$ is a Delaunay triangulation.*

Proof. We prove by contradiction. The idea in this proof comes from the local property of the Delaunay triangulation for $2D$ polygons [10]. Assume that $T(S)$ is not a Delaunay triangulation. Then one of the Voronoi vertices is not valid, which means there exists a point p interior to one of the polytopes in S such that p is inside a Voronoi sphere S_1. Suppose the tetrahedron corresponding to the Voronoi sphere is T_1. By definition of the junction tetrahedron, T_1 is outside of any polytopes. For a face f_1 of T_1, denote by $\widetilde{f_1}$ the sector of the sphere chopped off by the face f_1, denote by c_1 the intersection (a disk) of the sphere S_1 and the plane defined by f_1. Since $p \in S_1$ and p is not in T_1, then $p \in \widetilde{f_1}$ for some face f_1 of T_1. We define the solid angle from p to the disk c_1 to be $\theta(p, c_1)$. Among all the junction tetrahedrons whose circumspheres contain p, let T_1 be the one with the maximum $\theta(p, c_1)$ and let f_1 be the face of T_1 so that $p \in \widetilde{f_1}$, see Figure 2 (P_3 appears twice in the figure because we can not draw it since this situation won't happen.).

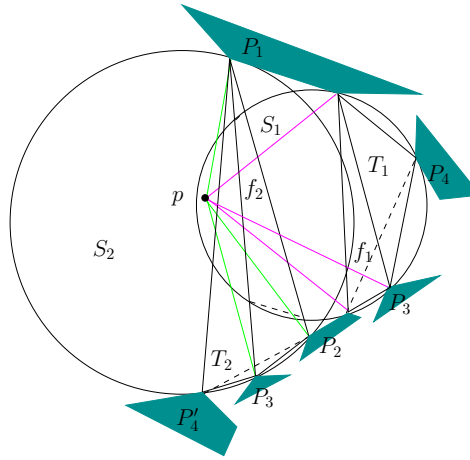


Fig. 2. Proof of the local property

Suppose the objects that are tangent with the face f_1 are P_1, P_2, P_3. Consider the tetrahedron T_2 that shares the same three objects with face f_1. Denote the corresponding face incident to P_1, P_2, P_3 as f_2. Claim that there exists a T_2 such that p is inside its circumsphere S_2. This comes from the local Delaunay property, i.e., p is not inside any of the neighboring polytopes of T_1. If all the circumspheres of T_1's neighboring junction tetrahedrons don't contain p, then either $\widetilde{f_1}$ is free of any objects, which contradicts with the assumption that p is inside $\widetilde{f_1}$, or there exists a junction tetrahedron defined for P, P_1, P_2, P_3 which contradicts the local Delaunay property.

From lemma 2, we know that p is not inside $\widetilde{f_2}$. Suppose the intersection of S_2 (S_1) with the plane defined by f_2 is c_2 (c_2'). Clearly c_2' is inside c_2. So the

solid angle $\theta(p, c_2) > \theta(p, c_2') > \theta(p, c_1)$, which contradicts with the maximality of $\theta(p, c_1)$.

3 Proximity Query in Delaunay Triangulation

The Delaunay triangulation can be used to perform efficient proximity queries. Specifically, given a query volume, we want to find all the objects intersecting with the query volume. Furthermore, we can keep track of the objects inside the query volume as the query volume is moving continuously. We explain the two cases in the following part.

3.1 Static Proximity Query

Delaunay triangulation can be used in the proximity search. Intuitively, in the Delaunay triangulation, two objects are connected if they are near each other in some direction. To find all the objects intersecting a query volume, we start from a initial object, which is known to be inside the query volume. Then, we check its neighbors in the Delaunay triangulation. Specifically, we keep a suspect list, which contains the objects that might be inside the query volume. The list is initialized to contain the initial seed. Each time we pop out one object, if it is inside the area, we put all its unchecked neighbors into the suspect list and continue the process until the list is empty. It is easy to prove that the method gives the correct result. If one object P intersecting the query volume isn't in the output, then P is not put in the suspect list, which implies that all P's neighbors are not in the list. So P is in a different connected component with the initial object, which contradicts with the connectivity of the Delaunay triangulation.

In the algorithm we assume that an object inside the query volume is known. In practice, the initial value can be obtained efficiently. For example, a moving robot subtracts information from the environment and build up the Delaunay triangulation incrementally. A typical task for the robot is to track a set of objects visible to the robot. So we can easily make use of the locality information and the initial value can always be obtained from the result in the last time step. The running time of the algorithm is $O(f(n)+k)$, where $f(n)$ is the time used to find the initial seed, and k is the number of objects intersecting with the query volume.

3.2 Kinetic Proximity Query

In the example above, the proximity query is done around a moving object, say, a robot. Changes to the proximity search result only happen in discrete time, when an object enters or leaves the query volume. This fits perfectly into the Kinetic Data Structure(KDS for short) framework [3] proposed by Bash et. al. In a KDS, we assume the robot follows a posted flight plan, though it can change it at any moment. The data structure will detect the change and update accordingly. The correctness of the structure is certified by a set of conditions, i.e., *certificates*.

Each certificate has a failure time, which is inserted as *events* in a event queue. Only when a certificate fails, the KDS certification repair mechanism is invoked to repair the certificate set and the result as well. For a survey of KDS on various problems, see [9].

The kinetic proximity search maintains a set of certificates. The query volume divides R^3 into 2 point sets, the inside and the outside. An object is called inside (outside) object if it is completely inside (outside) the query volume. An object is called an overlapping object if it intersects both inside and outside sets. We say an object is on the boundary, if it is an overlapping object, or has a neighbor in the Delaunay triangulation which is not of the same type with itself. Intuitively the set of objects inside the query volume will not change until one boundary object moves in or out of the query volume. Instead of computing the visible objects of the robot from scratch for every time step, we can instead maintain the set of visible objects as well as the boundary objects and update them whenever a boundary object moves in or out. Specifically, we have:

- An overlapping object P moves completely inside or outside. Its neighbors need to be updated. Some of them might be changed from boundary objects to non-boundary objects. When P moves totally outside, the query result needs to be changed as well.
- A boundary and non-overlapping object P becomes an overlapping object. Similarly, its neighbors need to be changed to boundary objects. P is inserted into the query result.

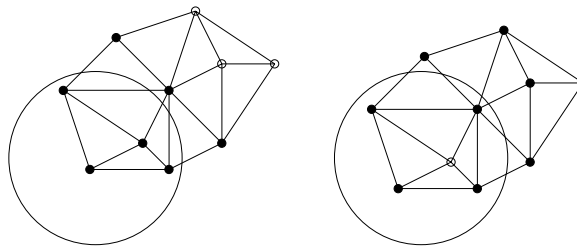


Fig. 3. Illustration of the maintenance of the objects inside the query volume. Boundary objects are shown as solid dots.

This method can be extended to the moving cuboids as well. Basically in addition to maintaining the boundary objects, we also need to maintain the Delaunay triangulation when the cuboids are moving around. We add more events to the system that certify the validity of a Voronoi vertex, in a standard KDS approach [9]. When an event fails, i.e., a Voronoi vertex becomes invalid, the Delaunay triangulation need to be updated. The update to the Delaunay triangulation involves only local flipping operation. So the update cost is $O(1)$ for

each event. One problem here is that computing when there exists a ball that touches five cuboids doesn't have closed form solution. So finding the failure time of a Voronoi vertex event can only be done numerically. We will talk more about this in the next section.

4 Computing the Delaunay Triangulation

Construction of the Delaunay triangulation for cuboids is much harder than the case of points, especially because the Voronoi sphere has no closed-form solution. Finding the Voronoi vertex is equivalent to finding a sphere that is tangent to four objects at the same time. We reduce this problem to an optimization problem and use the FSQP optimization algorithm [11]. For details, suppose the four objects are P_1, P_2, P_3, P_4 respectively, the center of the sphere is $X = (x_1, x_2, x_3)$, the radius of the sphere is r. Then the objective function we want to minimize is $\sum_{i=1}^{4}(\delta(X, P_i) - r)^2$. The optimization problem needs an initial value, which is obtained by computing the circumsphere defined by four points on each object. For the cube case, we choose the centers of the four objects. Considering that there might exist more than 1 Voronoi vertices for 4 objects and the FSQP optimization algorithm depends on the intial value, we have an alternate strategy if the optimation fails to find a Voronoi vertex. We retry with a different initial value, say the circumsphere defined by four points, with the four points being any of the vertices of the four cuboids. Experiments show that the method is very fast so that we can assume the computation takes $O(1)$ time. This method is extendable to convex objects, except that the computation of the distance from a point to a convex polytopes with m faces, costs $O(\log m)$ [5]. To compute the shortest distance from a point p to an arbitrary convex polytope P, we can compute the distance from p to a simplex whose vertices are vertices of the convex polytope. This value is a upper bound on the shortest distance since the simplex is entirely inside the polytope. The value decreases monotonically to the minimum when we refine the simplex. Combined with a hill climbing method, we can get the shortest distance in $O(\log m)$, where m is the number of faces of the convex polytope.

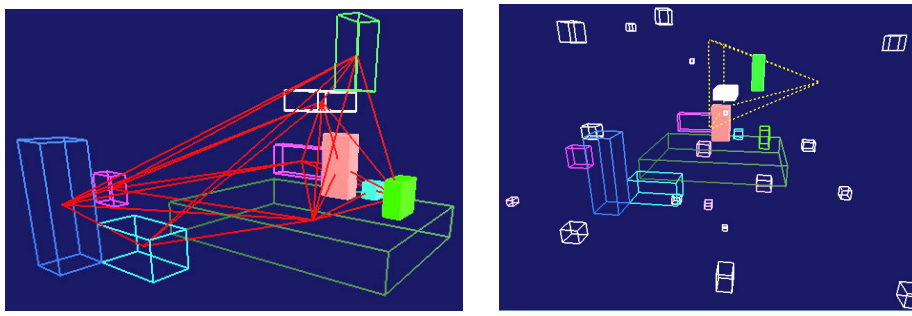
A well-known algorithm to construct the Delaunay triangulation for points is the randomized incremental algorithm. This algorithm can also be extended to 3D polytopes. The objects are inserted randomly and the Delaunay triangulation is updated upon insertion of new objects. By similar computation as in [4], the expected running time of this algorithm is $O(n \log n)$, where n is the number of objects, by assuming that finding the Voronoi vertex of four cuboids takes $O(1)$.

Upon deletion of one object P, we first delete all the Voronoi vertices incident to the object. The original Delaunay triangulation is left with a hole, with P's neighbors as the boundary. The hole is then filled by re-triangulating the boundary objects. Similarly if the position or the shape of a object P is changed. We only need to check the Voronoi vertices incident to P. If any of the Voronoi vertices are not valid, then they are deleted and the hole is filled by re-triangulating the boundary objects.

5 Experiments

We build a spatial data structure following the top-down approach. Object information is extracted from a stereo camera when the robot is exploring the environment. Information about objects and humans extracted from the scene is put in the spatial data structure. We extract object location and size information and organize them in 3D Delaunay triangulation that allows efficient proximity queries. We do not compute the Voronoi cells but only the Delaunay edges which are sufficient to determine proximity relationships. We build a 3D spatial model according to the partial information obtained in sequential frames, and update the Delaunay triangulation as more information comes in.

The Delaunay triangulation is used to compute a number of linguistic concepts of space such as *near, behind, between, inside,* and *among.* Figure 4 (i) shows a query for objects on the table. To determine objects that satisfy the query, we'll only need to search in the local neighborhood of the object using the Delaunay triangulation data structure. This kind of query will be very efficient even if there are hundreds of objects in the scene. An example with more objects is shown in figure 4 (ii). Here the objects intersecting the view frustum of the robot are shown as solid. The time taken for both queries is essentially same and independent of total number of objects in the environment.



(i) (ii)

Fig. 4. (i) The objects returned by the query *on the table* are shown solid. Delaunay edges are shown going from centroid of the cuboids. (ii) The objects inside the view frustum of the robot are shown solid.

6 Summary

In this paper we describe work towards spatial scene understanding for a mobile robot. We have provided proofs for the local property of Delaunay triangulation. We have shown how proximity queries can be performed using Delaunay triangulation. We propose a method for building a world model using sequential data

from a mobile stereo camera. Objects are modeled as 3D cuboids and organized in a Voronoi data structure, which supports efficient proximity searches. There are multiple advantages of this representation. First, the data can be incremental and we do not need to see full object in a view to model its spatial properties. Second, using this data structure allows efficient spatial queries in the robot egocentric frame of reference. We can efficiently answer queries about relationship among objects in the scene or objects in the proximity of a given object.

Such a world model with a shared spatial context allows natural interaction between the human and mobile robot. The human can then effectively instruct the robot on where to go and what to look for.

References

1. P. Agarwal, M. de Berg, J. Gudmundsson, M. Hammar, and H. Haverkort. Box-trees and R-trees with near-optimal query time. In *Proceedings of the 17th ACM Symposium on Computational Geometry (SoCG 01')*, pages 124–133, June 2001.
2. F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, Sept. 1991.
3. J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Alg.*, 31(1):1–28, 1999.
4. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
5. D. W. J. E. G. Gilbert and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Trans. Robotics and Automation*, 4(2):193–203, April 1988.
6. G. Edwards and B. Moulin. Toward the simulation of spatial mental images using the voronoi model. *Representation and Processing of Spatial Expressions*, pages 163–184, 1997.
7. J. Erickson. Dense point sets have sparse delaunay triangulations. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 125–134, 2002.
8. M. Gavrilova. *Proximity and Applications in General Metrics*. PhD thesis, Dept. of Computer Science, The University of Calgary, 1998.
9. L. J. Guibas. Kinetic data structures — a state of the art report. In P. K. Agarwal, L. E. Kavraki, and M. Mason, editors, *Proc. Workshop Algorithmic Found. Robot.*, pages 191–209. A. K. Peters, Wellesley, MA, 1998.
10. L. J. Guibas, J. Snoeyink, and L. Zhang. Compact voronoi diagrams for moving convex polygons. In *7th Scandinavian Worshop on Algorithm Theory*, 2000.
11. C. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for cfsqp version 2.5: A c code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report Institute for Systems Research TR-94-16R1, University of Maryland, College Park, 1997.
12. M. McAllister, D. G. Kirkpatrick, and J. Snoeyink. A compact piecewise-linear voronoi diagram for convex sites in the plane. In *IEEE Symposium on Foundations of Computer Science*, pages 573–582, 1993.
13. O. Procopiuc. Data structures for spatial database systems.

An Explicit Solution for Computing the Euclidean *d*-dimensional Voronoi Diagram of Spheres in a Floating-Point Arithmetic

M.L. Gavrilova

Dept of Comp. Science, University of Calgary
Calgary, AB, Canada, T2N1N4
marina@cpsc.ucalgary.ca

Abstract. The problem of computing a *d*-dimensional Euclidean Voronoi diagram of spheres is relevant to many areas, including computer simulation, motion planning, CAD, and computer graphics. This paper presents a new algorithm based on the explicit computation of the coordinates and radii of Euclidean Voronoi diagram vertices for a set of spheres. The algorithm is further applied to compute the Voronoi diagram with a specified precision in a fixed length floating-point arithmetic. The algorithm is implemented using the ECLibrary (Exact Computation Library) and tested on the example of a 3-dimensional Voronoi diagram of a set of spheres.

1 Introduction

In areas such as motion planning, computer simulation of physical systems, mechanical engineering, robotics and computer graphics, the focus of study is often on the geometric objects, approximated by spheres. In these applications, the issues concerning robustness and numerical stability of algorithms, as well as the actual running times of their implementations, are crucial [1,3,4,12].

Numerical degeneracy, resulting from performing calculations on the real numbers in a computer floating-point arithmetic, is often handled by approximation [11]. Over the last 25 years, an alternative approach to real computation has been developed, called the *exact real computation*, in which results are generated exactly [2,5,7,8,12]. As identified by L. Blum [1], the state of the art in development of numerical and simulation software, as well as computer algebra systems, is moving towards exact systems which could be used to replace the traditional approximate systems.

The main contribution of this paper is in a new algorithm to compute the generalized Voronoi diagram vertex in R^d using *explicit formulas* for coordinates and radii of Voronoi vertices. The algorithm is further extended to compute the values with the precision to the smallest representable floating-point number in a given computer architecture. Note, that the computation of the exact Delaunay triangulation [2,6] is a much simpler task, which does not involve computing the *d*-dimensional coordinates of the Voronoi vertices. It is also not feasible to use a

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 827–835, 2003.
© Springer-Verlag Berlin Heidelberg 2003

traditional approach of computing the Voronoi vertices using libraries for exact computation, such as PRECISE [7] or LEDA [8], due to the loss of efficiency.

To tackle the problem, we developed an original approach, which combines an interval analysis techniques, the Exact Value of Algebraic Expression method, and an iterative approximation method to compute the intersection of d-dimensional quasi-halfspaces with hyperbolic boundaries. The algorithm possesses the following characteristics: (1) it obtains explicit formulas for d-dimensional vertices, which allows to design a fairly simple algorithm, (2) it is based on Exact Value of Algebraic Expression method, which is the next step over the ESSA method [6], (3) it uses the advanced iterative method for finding solution with the given precision based on computation of Jacobian for a system of quadratic equations. As a result, much more efficient implementation and simpler method for computing vertices of d-dimensional Voronoi diagram of a set of spheres is obtained, than, for instance, method described in [14]. The method is implemented using the ECLibrary [13]. Experimental results for computing a Voronoi vertex of a set of spheres in 3D are also obtained.

2 The Problem

Given $(d+1)$ spheres in R^d: $P_i = \{\mathbf{p}_i = (x_{i1}, x_{i2}, ..., x_{id}), r_i\}, i = 1..d+1$, where x_{ij} and r_i are represented by floating-point numbers. The Voronoi diagram is used to store the topological information about the system of spheres as well as to answer various queries.

Definition 1. A generalized Euclidean Voronoi diagram for a set of spheres S in R^d is the set of Voronoi regions $\left\{ \mathbf{x} \in R^d \,\middle|\, d(\mathbf{x}, P) \leq d(\mathbf{x}, Q), \forall Q \in S - \{P\} \right\}$, where $d(\mathbf{x}, P)$ is the Euclidean distance function between a point $\mathbf{x}$ and a sphere $P \in S$.

The distance between a point $\mathbf{x}$ and a sphere P with center at $\mathbf{p}$ and radius r_p is defined as

$$d(\mathbf{x}, P) = d(\mathbf{x}, \mathbf{p}) - r_p, \tag{1}$$

where

$$d(\mathbf{x}, \mathbf{p}) = \sqrt{\sum_{i=1}^{d} (x_i - p_i)^2}. \tag{2}$$

According to the definition, the generalized Voronoi vertex is obtained as the intersection of d quasi-halfspaces with hyperbolic boundaries (see Fig. 1). The task is to find the best floating-point approximation of the Voronoi vertex, i.e. the point in the space equidistant from these spheres.

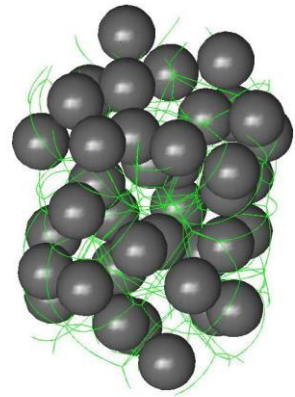


Fig. 1. The Euclidean Voronoi diagram of a set of spheres in 3D

3 The Explicit Solution for Finding Coordinates and Radii of Voronoi Diagram Vertices

3.1 Algebraic Representation of a Solution

First, let us obtain the algebraic representation of the solution for the coordinates and the radius of the inscribed sphere (i.e. the Voronoi vertex) in R^d. Then, we use a specific technique to derive the explicit solution for the coordinates and the radius, using a substitution technique. We use estimates on the number of inscribed spheres in the Euclidean metric, obtained in [14], required to derive the algorithm for computation of the best floating point approximation of a Euclidean Voronoi vertex in d dimensions.

Definition 2. A sphere $C = \{\xi, \rho\}$ inscribed among $d+1$ spheres $P_1, P_2, ..., P_{d+1}$ is a sphere with center $\xi = (\xi_1, \xi_2, ..., \xi_d)$ and radius ρ, such that $\rho = d(\xi, P_1) = d(\xi, P_2) = ... = d(\xi, P_{d+1})$.

The above system can be represented as:

$$\begin{cases} (x_{11} - \xi_1)^2 + (x_{12} - \xi_2)^2 + ... + (x_{1d} - \xi_d)^2 - (r_1 + \rho)^2 = 0 \\ (x_{21} - \xi_1)^2 + (x_{22} - \xi_2)^2 + ... + (x_{2d} - \xi_d)^2 - (r_2 + \rho)^2 = 0 \\ ... \\ (x_{d+1,1} - \xi_1)^2 + (x_{d+1,2} - \xi_2)^2 + ... + (x_{d+1,d} - \xi_d)^2 - (r_{d+1} + \rho)^2 = 0 \end{cases} \tag{3}$$

The center of the sphere represents the vertex of the Voronoi diagram. First, we apply the reduction technique to obtain a system of d linear equations with $d+1$ unknown variables $(\xi_1, \xi_2, ..., \xi_d, \rho)$. This system will be then transformed to obtain a quadratic equation of ρ. Quadratic roots will be found precisely using the iterative approximation method based on computation of Jacobian

of a quadratic system of equations, from which the coordinates of the Voronoi vertex will be derived.

In order to obtain the system of linear equations, let us reduce radii of the $d+1$ spheres by the radius of the smallest sphere (assume that it is r_{d+1}). Define a coordinate system with the center of coordinates at $\mathbf{p}_{d+1}$. Coordinates of the spheres after the transformation are:

$$P_i^* = \{\mathbf{p}_i^* = (x_{i1}^*, x_{i2}^*, ..., x_{id}^*), r_i^*\}, i = 1..d, \tag{4}$$

where

$$x_{ij}^* = x_{ij} - x_{d+1,j}, r_i^* = r_i - r_{d+1}, i, j = 1..d. \tag{5}$$

The last sphere is transformed into a point at the origin of coordinates. The coordinates of the center of the inscribed sphere C change to $\xi_j^* = \xi_j - x_{d+1,j}, j = 1..d$, and the new radius is $\rho^* = \rho + r_{d+1}$.
In Euclidean metric, coordinates of the inscribed sphere satisfy following system of equations:

$$d(\xi^*, \mathbf{p}_i^*) = \rho^* + r_i^*, i = 1..d+1. \tag{6}$$

Expanding the distance function, we obtain:

$$\begin{cases} (x_{i1}^* - \xi_1^*)^2 + ... + (x_{id}^* - \xi_d^*)^2 = (\rho^* + r_i^*)^2 \\ (\xi_1^*)^2 + (\xi_2^*)^2 + ... + (\xi_d^*)^2 = (\rho^*)^2 \end{cases} \tag{7}$$

where $i = 1..d$. The last equation can be subtracted from the remaining equations to cancel the quadratic terms. Thus, we arrive at:

$$\begin{cases} x_{i1}^*\xi_1^* + x_{i2}^*\xi_2^* + ... + x_{id}^*\xi_d^* + r_i^*\rho^* = w_i^* \\ (\xi_1^*)^2 + (\xi_2^*)^2 + ... + (\xi_d^*)^2 = (\rho^*)^2 \end{cases} \tag{8}$$

where $w_i^* = \frac{1}{2}\left((x_{i1}^*)^2 + (x_{i2}^*)^2 + ... + (x_{id}^*)^2 - (r_i^*)^2\right), i = 1..d$.
Consider the first d equations of the above system. They are linear in $(\xi_1, \xi_2..., \xi_d, \rho)$. We can represent the first d linear equations in a form:

$\mathbf{A_r} * \mathbf{z} = \mathbf{b}$, where $\mathbf{A_r}$ is the matrix of the left hand side of the system, $\mathbf{z} = (\xi_1, \xi_2..., \xi_d, \rho)^T$ and $\mathbf{b} = (w_1^*, w_2^*..., w_d^*, \rho)^T$.

According to [14], the following Lemma is true:

Lemma 1 The linear system (8) has an infinite number of solutions when $rank(\mathbf{A_r}) = rank(\mathbf{A_r}|\mathbf{b}) < d$, has zero solutions when $rank(\mathbf{A_r}) < rank(\mathbf{A_r}|\mathbf{b}) \leq d$, can have up to two solutions when $rank(\mathbf{A_r}) = rank(\mathbf{A_r}|\mathbf{b}) = d$.

Thus, if we find d minor of matrix $\mathbf{A_r}$ that is not zero, we will proceed in finding the coordinates of the center of the inscribed sphere.

3.2 Explicit Algebraic Solution

We proceed to obtain the explicit formulas for the coordinates and the radii of inscribed sphere, by performing additional transformations on the above system. These formulas allow to simplify the process of finding Voronoi vertices in d-dimensions significantly.

Without the loss of generality, assume that the minor of first d columns of $\mathbf{A_r}$ is non-zero. Denote this minor by $\mathbf{A}$. Thus, we can select the radius of the inscribed sphere as the free variable. Moving it to the right-hand-side, we get a linear system for d variables, representing the coordinates of the inscribed sphere. Applying Cramer's rule, we get the formulas for the coordinates:

$$\xi_i^* = \frac{A_i}{A}, i = 1..d, \tag{9}$$

where

$$A = \begin{vmatrix} x_{11}^* & x_{12}^* & \dots & x_{1d}^* \\ x_{21}^* & x_{22}^* & \dots & x_{2d}^* \\ \dots & \dots & \dots & \dots \\ x_{d1}^* & x_{d2}^* & \dots & x_{dd}^* \end{vmatrix} \tag{10}$$

and A_i is obtained by replacing the i-th column of A by the column $\left[w_j^* - r_j^* \rho^*\right]_{j=1..d}$.

$$A_i = \begin{vmatrix} x_{11}^* & \dots & x_{1,i-1}^* & w_1^* - r_1^*\rho^* & x_{1,i+1}^* & \dots & x_{1d}^* \\ x_{21}^* & \dots & x_{2,i-1}^* & w_2^* - r_2^*\rho^* & x_{2,i+1}^* & \dots & x_{2d}^* \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_{d1}^* & \dots & x_{d,i-1}^* & w_d^* - r_d^*\rho^* & x_{d,i+1}^* & \dots & x_{dd}^* \end{vmatrix} \tag{11}$$

$A_i = A_{wi} - \rho^* A_{ri}$, where

$$A_{wi} = \begin{vmatrix} x_{11}^* & \dots & x_{1,i-1}^* & w_1^* & x_{1,i+1}^* & \dots & x_{1d}^* \\ x_{21}^* & \dots & x_{2,i-1}^* & w_2^* & x_{2,i+1}^* & \dots & x_{2d}^* \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_{d1}^* & \dots & x_{d,i-1}^* & w_d^* & x_{d,i+1}^* & \dots & x_{dd}^* \end{vmatrix} \tag{12}$$

$$A_{ri} = \begin{vmatrix} x_{11}^* & \dots & x_{1,i-1}^* & r_1^* & x_{1,i+1}^* & \dots & x_{1d}^* \\ x_{21}^* & \dots & x_{2,i-1}^* & r_2^* & x_{2,i+1}^* & \dots & x_{2d}^* \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_{d1}^* & \dots & x_{d,i-1}^* & r_d^* & x_{d,i+1}^* & \dots & x_{dd}^* \end{vmatrix} \tag{13}$$

Now we can substitute algebraic expressions for d unknowns ξ_i^* into the last quadratic equation of the system (8). Multiplying the result by A^2, we arrive to:

$$\sum_{i=1}^{d} (A_{wi} - \rho^* A_{ri})^2 - (\rho^*)^2 A^2 = 0. \tag{14}$$

This is a quadratic equation in ρ^*: $a(\rho^*)^2 - 2b\rho^* + c = 0$, where

$$a = \sum_{i=1}^{d} A_{ri}^2 - A^2, b = \sum_{i=1}^{d} A_{ri} A_{wi}, c = \sum_{i=1}^{d} A_{wi}^2. \tag{15}$$

The solution of this quadratic equation is

$$\rho^*_{1,2} = \frac{b \pm \sqrt{b^2 - ac}}{a}. \tag{16}$$

If $a = 0$ then the single solution is found as

$$\rho^* = \frac{c}{2b}. \tag{17}$$

After the solution ρ^* is obtained, we obtain the values for remaining variables substituting the solutin into the equation (9), and then return to the original coordinates remembering that the coordinates of the original center of the inscribed sphere C were changed to the new ones as $\xi^*_j = \xi_j - x_{d+1,j}, j = 1..d$, and the new radius is $\rho^* = \rho + r_{d+1}$.

4 Computing Voronoi Vertices with a Required Precision in a Floating-Point Arithmetic

This section presents the algorithm for the computation of the coordinates of the Voronoi diagram vertices, based on the obtained explicit formulas. The algorithm proceeds by computing the coordinates of the Voronoi vertices utilizing the exact floating-point arithmetic. It uses the advanced iterative method for finding solution with the given precision based on computation of Jacobian for a system of quadratic equations. The interval arithmetic is employed as needed for further optimization.

First, we apply the Exact Value of Algebraic Expression method algorithm to find a non-zero minor of $\mathbf{A_r}$ exactly.

We use an iterative Newton's method to obtain the exact solution. The numeric solution obtained in non-precise arithmetic by formulas (16) or (17) is taken as the initial approximation $\mathbf{x}_0$.
On each step, we compute the discrepancy $\mathbf{\Delta}_n = \mathbf{F}(\mathbf{x}_n)$, where $\mathbf{F}(\mathbf{x})$ is the system of equations (3).

The next iteration is performed as $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}_n^{-1}\mathbf{\Delta}_n$, where $\mathbf{J}_n$ is the Jacobian of the system (3):

$$\mathbf{J} = 2^{d+1} \begin{bmatrix} \xi_1 - x_{11} & \xi_2 - x_{12} & \dots & \xi_d - x_{1d} & \rho + r_1 \\ \xi_1 - x_{21} & \xi_2 - x_{22} & \dots & \xi_d - x_{2d} & \rho + r_2 \\ \dots & \dots & \dots & \dots & \dots \\ \xi_1 - x_{d+1,1} & \xi_2 - x_{d+1,2} & \dots & \xi_d - x_{d+1,d} & \rho + r_{d+1} \end{bmatrix} \tag{18}$$

The algorithm terminates when the value of the discrepancy computed at the next iteration step remains the same, i.e. it the solution was obtained with the specified precision in a number of bits. Following the above discussion, we obtain Lemma 2:

Lemma 2 The algorithm for reliable Euclidean Voronoi vertex computation will determine the radius of the inscribed sphere in d-dimensional space exactly with the precision to the last representable digit.

5 Experimental Results

The method feasibility was verified on an example of a 3D Euclidean Voronoi diagram of a set of spheres. Algorithms were implemented in object-oriented C++ environment on 1.4 MHz Intel processor. All operations on floating-point numbers were performed using the ECLibrary package, based on representation of a floating-point number as an array of bit strings of the specified length, and performing manipulations on these arrays to compute the results of arithmetic operations, such as addition, subtraction, multiplication, division and taking a square root.

The experiments were conducted on two configurations: one representing a random distribution of spheres confined inside a cube, another is a close to a degenerate case when four spheres in 3D are almost tangent to the same 3D planes (see Fig. 2). Note that in the second case the coordinates of the spheres are slightly disturbed in order to make it possible to find a solution (i.e. the inscribed sphere).

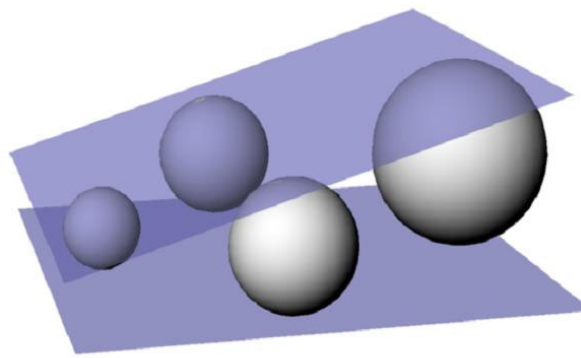


Fig. 2. An example of a close to degenerate case for a set of spheres in 3D

First, the number of iterations required to obtain the exact to a specified number of bits solution was measured. The computations were carried out to compute the 53 bits representing a mantissa of a floating-point number exactly. Experiments showed that on average only two iterations were required to obtain the desired precision.

Next, the time required for a single iteration was measured. The most time consuming step was computation of an inverse of a Jacobian (18), and the decrease in a speed vs. the direct non-exact implementation was somewhere between one to two orders of magnitude, depending on the number of iterations. However, precision increased up to 10 times for both configurations. This is illustrated in the Table 1:

Distribution	1st Approximation	Iteration
Random	10^{-13}	10^{-14}
Degenerate	10^{-6}	10^{-7}

Table 1. Discrepancy values for random and close to degenerate sphere distributions

In the above table, rows represent the type of the distribution, and the columns represent the values of discrepancy first computed using non-exact arithmetic, and then obtained after the first iteration using Newton's method. Note that values of discrepancy for degenerate case is much higher than for the random distribution of spheres, which is consistent with the fact that inexact method produces much more coarse results for close to degenerate cases.

Note that the method presented in [14], based on the division-in-half iterative technique, produces the worst results (up to 3 times slower) than the presented method based on ECLibrary. Note that the library outperforms LEDA library on most of arithmetic operations to the order of a magnitude.

6 Conclusions

The new algorithm based on the explicit computation of the coordinates and radii of Euclidean Voronoi diagram vertices for a set of spheres is presented. Results obtained confirm the algorithm correctness and demonstrate suitability of the method for reliable computation of the Euclidean Voronoi diagram. The method possesses a number of important characteristics, such as simplicity, easy of implementation and possibility of bringing to different platforms (such as ECLibrary, GMP, etc.). The algorithm is more efficient than the previously developed method using ECLibrary or LEDA implementation. It can be directly applied for reliable computation of other generalized Voronoi diagrams, including the Voronoi diagram in Laguerre geometry, and in L_1 and L_{inf} metrics. On the further quest toward the more efficient solution, we would like to improve the underlying algorithms in the ECLibrary. Bringing the algorithms to the GMP platform (a part of CORE library) is another task that we foresee in the near future.

7 Acknowledgements

The author would like to thank the University of Calgary Research Services (Start-Up grant) for partial support of the project.

References

1. Blum, L., Cucker, F., Shub, M. and Smale, S. "Complexity and Real Computation" (1997)
2. Dey, T.K., Sugihara K. and Bajaj, C. L. DT in three dimensions with finite precision arithmetic, Comp. Aid. Geom. Des **9**(1992) 457-470
3. Edelsbrunner, H. And Mcke, E. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, 4th Annual ACM Symposium on Computational Geometry (1988) 118-133.
4. Fortune, S. And Wyk, C. Efficient exact arithmetic for computational geometry, 4th Annual ACM Symp. Comput. Geometry, (1993) 163-172
5. Gavrilova, M. and Rokne, J. Reliable line segment intersection testing, Computer Aided Design, (2000) 32, 737-745.
6. Gavrilova, M., Ratschek, H. and Rokne, J. Exact computation of Voronoi diagram and Delaunay triangulation, Reliable Computing, (2000) 6(1) 39-60.
7. S. Krishnan, M. Foskey, T. Culver, J. Keyser, D. Manocha, PRECISE: Efficient Multiprecision Evaluation of Algebraic Roots and Predicates for Reliable Geometric Computations, Symp. on Computat. Geometry (2002)
8. Naher, S. The LEDA user manual, Version 3.1 (Jan. 16, 1995). Available from ftp.mpi-sb.mpg.de in directory /pub/LEDA.
9. Ratschek, H. And Rokne, J. Exact computation of the sign of a finite sum, Applied Mathematics and Computation, (1999) 99, 99-127.
10. Rokne, J.: Interval arithmetic. In: Graphics Gems III, Academic Press, pp. 61-66 and pp. 454-457 (1992).
11. Sugihara, K. And Iri, M. A robust topology-oriented incremental algorithm for Voronoi diagrams, IJCGA, (1994) 4 (2): 179-228.
12. Yap, C., Dub, T. The exact computation paradigm, In "Computing in Euclidean Geometry" (2nd Edition). Eds. D.-Z. Du and F.K. Hwang, World Scientific Press (1995)
13. Gavrilova, M. (2002) Algorithm library development for complex biological and mechanical systems: functionality, interoperability and numerical stability DIMACS Workshop on Implementation of Geometric Algorithms, December 2002 (abstract)
14. Gavrilova, M. (2002) A Reliable Algorithm for Computing the Generalized Voronoi Diagram for a Set of Spheres in the Euclidean d-dimensional Space, in the Proceedings of the 14th Canadian Conference on Computational Geometry, August, Lethbridge, Canada

Dynamically Maintaining a Hierarchical Planar Voronoi Diagram Approximation

Imma Boada, Narcís Coll, and J. Antoni Sellarès

Institut d'Informàtica i Aplicacions
Universitat de Girona, Spain
{imma,coll,sellares}@ima.udg.es

Abstract. An approach for computing a hierarchical approximation of planar Voronoi diagrams for different site shapes (points, line-segments, curve-arc segments, ...) and different distance functions (Euclidean metrics, convex distance functions, ...) was presented in [3]. The approach is based on the Voronoi-Quadtree, a quadtree data structure from which a polygonal approximation, represented by a DCEL structure, of the associated Voronoi region boundaries can be computed at different levels of detail. In this paper we describe efficient algorithms for dynamically maintaining, under the insertion or deletion of sites, the Voronoi-Quadtree and the corresponding DCEL structure representing an approximation of a Generalized Voronoi diagram.

1 Introduction

The generalized Voronoi diagram of a set of sites partitions the plane into regions, one per site, in a way that all points in a region have the same closest site according to some given distance function. Many alternatives have been considered, taking sites of different shape or nature, associating weights to the sites, changing the underlying metrics, or using individualized distance functions for the sites [1,2,11].

The algorithms designed for computing exact generalized planar Voronoi diagrams often have numerical robustness problems and are time-consuming due to the numerous high precision calculations that are required. However in some applications (like motion planning or geographic map simplification) the computation of an approximated Voronoi diagram within a predetermined precision is sufficient. Several algorithms have been proposed for the approximation of Voronoi diagrams [8,9,10,13,14]. In [3] we proposed an algorithm for approximating generalized planar Voronoi diagrams for different site shapes (points, line-segments, curve-arc segments, ...) and different distance functions (Euclidean metrics, convex distance functions, ...) in the case that all the Voronoi regions are connected. The algorithm is based on a quadtree structure, the Voronoi-Quadtree (VQ), from which a polygonal approximation of the associated Voronoi region boundaries can be computed at different levels of detail. This approximation is represented by a DCEL, an structure with three types of records, vertices,

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 836–846, 2003.
© Springer-Verlag Berlin Heidelberg 2003

halfedges and faces, that maintains adjacencies between regions of a Voronoi diagram [4]. The most relevant features of this approach are, its generality (not all the sites must be homogeneous in shape or have associated the same distance function), robustness, efficiency and easy implementation. Moreover, the VQ is a suitable data structure for approximate nearest neighbor queries.

Motivated by the attractiveness of the VQ features, our main interest has been centered on the dynamic maintenance (under insertion and deletion of sites) of a VQ and the corresponding DCEL structure representing an approximation of a Generalized Voronoi diagram. The construction and maintenance of a dynamic exact Voronoi diagram has been treated in [5,6]. In this paper we present an insertion and a deletion algorithm. Given a VQ data structure, both algorithms are capable of generating the updated VQ and the corresponding DCEL representation, not generating the solution from scratch.

For the sake of completeness, in the next section we summarize the fundamental concepts and algorithms related to the VQ described in [3].

2 The Voronoi-Quadtree

A quadtree encodes the recursive subdivision of a rectangular region in the plane. The root of the tree represents the region. This region is subdivided into four identical rectangular regions, called quadrants. Each quadrant is represented by one of the four nodes descendants of the root. If the information of a quadrant is not simple enough it is subdivided into another four identical rectangular regions which are considered as descendant nodes of the node representing the quadrant in question. This process is repeated recursively until quadrants contain simple enough data (*terminal node*s) or quadrants have a minimum edge length called resolution (*maximal subdivision nodes*). The key point on the quadtree construction is the definition of the criteria that determines when a node is considered terminal or not.

2.1 Basic Definitions

A site S is represented by $S =< G_S, D_S, P_S, L_S >$, where G_S defines the geometry of the site S, D_S is the function that gives the distance from any point p to S, P_S (the base point of S) is a point such that $D_S(P_S) = 0$ and L_S is a label used to identify the site S. For example, if S is a circle then G_S is composed of the center and radius of S, if S is a segment then G_S is composed by the endpoints of S. When S is the circle of center C and radius r, if we consider the Euclidean distance, then $D_S(p) =\parallel p - C \parallel - r$, but if we consider the convex distance function determined by S, then $D_S(p) =\parallel p - C \parallel / r$.

Let $\mathcal{S} = \{S_1, \cdots, S_n\}$ be a set of sites. Each site $S_i \in \mathcal{S}$ has associated a Voronoi region $VR(S_i) = \{p | D_{S_i}(p) \leq D_{S_j}(p) \text{ for all } j \neq i\}$. The generalized Voronoi diagram of $\mathcal{S}$, denoted $VD(\mathcal{S})$, is defined as the decomposition of the plane into Voronoi regions.

Let R be a rectangular region and S_i a site such that $R \cap VR(S_i) \neq \emptyset$. We say that the S_i is *inside* R when $P_{S_i} \in R \cap VR(S_i)$ and we say that S_i is *outside* R otherwise. A rectangular region R is called a *basic* region when R intersects less than four Voronoi regions. A terminal node is a node that corresponds to a basic region.

2.2 VQ Construction Process

Once terminal nodes are characterized, to obtain the VQ of the part of the Voronoi diagram $VD(\mathcal{S})$ contained in a rectangular region we proceed as follows. Starting with the root, at which it has been assigned the rectangular region R containing the sites of $\mathcal{S}$, a subdivision process that partitions the node in four new identical rectangular regions, denoted $R0$, $R1$, $R2$ and $R3$, is applied. This subdivision process is repeated until the actual region is a basic region (i.e. a terminal node) or it is of maximal subdivision. The subdivision process is composed of three steps. The *labeling step* that assigns at each vertex of $R0$, $R1$, $R2$ and $R3$ the label of the nearest site belonging to the set of sites classified as inside or outside with respect to its parent. The *inside sites identification step* that attaches at $R0$, $R1$, $R2$ and $R3$ the sites classified as inside with respect to them. Finally, the *propagation step* that assigns outside sites to the neighbors of the processed node if it is required. This step is the key point of the subdivision process. During the labeling step, only inside and outside sites with respect to R are considered. Therefore, it is possible that an incorrect vertex label assignation appears. The propagation step solves this incorrectness and guarantees that increasing the level of subdivision, we are able to obtain a solution that tends to the exact one. In this way during the VQ construction process, the actual VQ is used as an approximation of the Voronoi diagram. This is the main difference with respect to [13] and [14].

For each node N representing a region R, we store: (i) the set $V(N)$ composed of the labels assigned, by the labeling step, to the vertices of N; (ii) the set $INsites(N)$ composed of the labels of the sites of $\mathcal{S}$ classified as inside with respect to R and (iii) the set $OUTsites(N)$ composed of the labels of the sites of $\mathcal{S}$ that are classified as outside with respect to R. Being $L(N)$ the total number of different labels in $V(N)$, $INsites(N)$ and $OUTsites(N)$, N is classified as a terminal node if $L(N) \leq 3$. Characterized by this $L(N)$ value and the distribution of $V(N)$ labels, only five terminal node configurations are possible (see Figure 1).

2.3 DCEL Construction

At the end of the VQ construction process we have a set of terminal nodes distributed at possibly different levels of the VQ. For each terminal node we can determine the set of Voronoi region boundaries intersecting the rectangular region represented by the node. The relation between the terminal configuration and the Voronoi regions is (see Figure 1): $T0$, the node is contained in a unique Voronoi region; $T1$, the node is intersected by two Voronoi regions whose

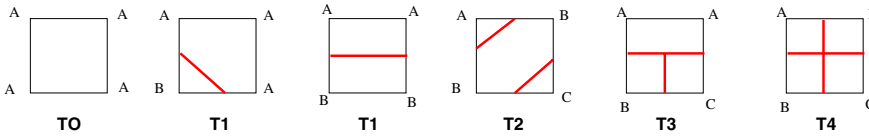


Fig. 1. *Configurations of VQ Terminal nodes:* $T0$ *when* $L(N) = 1$*;* $T1$ *when* $L(N) = 2$*;* $T2$ *when* $L(N) = 3$*;* $T3$ *when* $L(N) = 3$ *and one of the edges of the node has the same labels in its extremes;* $T4$ *when* $L(N) \geq 4$ *and the node* N *is of maximal subdivision.*

boundaries intersect two edges of the node; $T2$, the node is intersected by three Voronoi regions whose boundaries intersect four edges of the node; $T3$ the node is intersected by three Voronoi regions whose boundaries intersect three edges of the node; $T4$ the node is intersected by four or more Voronoi region boundaries that intersect all the edges of the node.

The above mentioned correspondence is the basis of the DCEL construction algorithm. The algorithm traverses the VQ and applies a function to each terminal node. The function approximates the intersection points between the Voronoi boundaries and the edges of the node, and determines how they have to be connected. As terminal nodes may be distributed at different levels of the VQ, to guarantee the continuity of the boundary of the Voronoi diagram, the intersection points are determined taking into account the level of the processed node and the level of its terminal neighbor nodes.

3 Dynamic Maintenance

In this section we present the insertion and deletion algorithms proposed for the dynamic maintenance of the VQ and its associated DCEL. To perform this maintenance we introduce a link, as a set of pointers, between the VQ terminal nodes and the edges of the DCEL. Both algorithms guarantee that only the nodes intersected by the boundary of the Voronoi region associated to the site to be inserted or deleted are processed.

3.1 Insertion Algorithm

Given a VQ codification and a site $T =< G_T, D_T, P_T, L_T >$ to be inserted the insertion algorithm applies a propagation process that goes from the leaf node containing T to all the nodes containing a piece of the boundary of its associated Voronoi region. The four steps that compose the insertion algorithm are described next.

Step 1. *Apply a top-down VQ traversal to identify the terminal node* (N) *that contains* P_T*. Initialize with null the queue* (Q) *used to maintain the nodes to be processed.* In Figure 2(a) and (b) an exact Voronoi diagram and its approximation have been represented. Taking this situation as an example and considering G as the label of T, the located node in step 1 is the one represented in Figure 2(c).

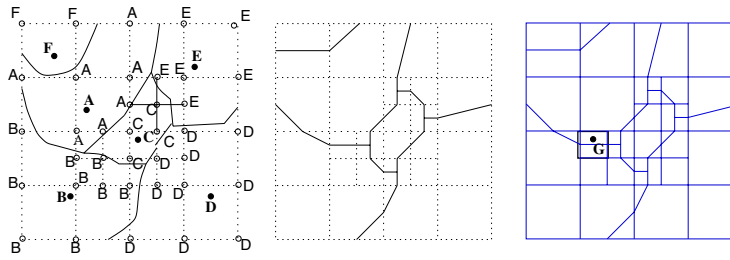


Fig. 2. *(a)(b)Voronoi Diagram and its approximation. Each rectangular region corresponds to one terminal VQ node. Each vertex of the node maintains its corresponding label (c) Identification of the terminal node containing the site G to be inserted.*

Step 2. *Apply the labeling process to the node N considering the label of the new site L_T. Modify the labels of the vertices and the pointers to the DCEL if it is required.* The assignment of the new labels has two major consequences. (i) The configuration of the new labeled node may change, from a terminal node to a non terminal node or from one terminal configuration to another one. If the node becomes non terminal it has to be subdivided and each one of its sons has to be processed. If a new terminal configuration is generated the new Voronoi boundaries intersected by the node have to be computed and the DCEL entries have to be updated. The application of the labeling process to the diagram of Figure 3(a) is represented in Figure 3(b). The node of Figure 3(b) becomes a $T0$ and thus the pointers to the DCEL are set to null. (ii) The modification of one vertex label leads a modification on the configuration of the neighbor nodes with this vertex in common and also the modification of their DCEL entries. Compare labels of nodes of Figure 3(a) with the labels of the nodes of Figure 3(c). The assignment of the new labels generates the Voronoi diagram approximation represented in Figure 3(c), where new boundaries correspond to thick lines.

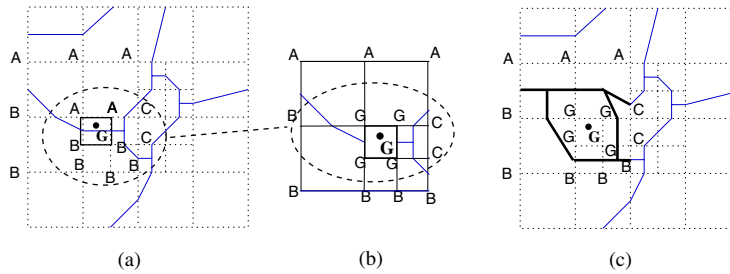


Fig. 3. *(a)Initial Voronoi diagram approximation and the initial assigned labels. (b) Labels assigned to the node with the new site as inside. (c) New codification of neighbor nodes with the modified vertex in common.*

Step 3. *For each one of the nodes detected in the previous step with the common modified vertex of N, check if the modification of any other of its vertices labels is required. If it is required send the node to the queue Q, otherwise go to Step 4.* In Figure 4(a) the nodes to be processed are represented by a continuous line. The new assignment of labels for this set of nodes is represented in Figure 4(b). Five labels have been modified and in this case the five nodes are sent to the queue (see Figure 4(c)).

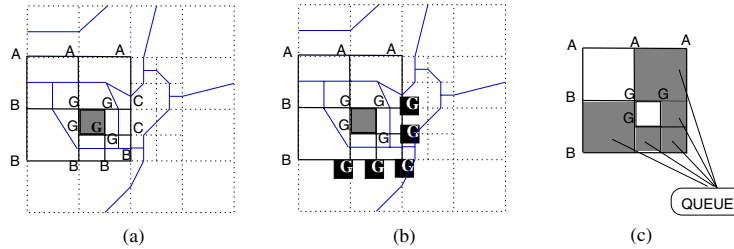


Fig. 4. *(a)(b)Initial Voronoi diagram approximation. Labels assigned to the node with the new site as inside. New assigned labels.(c) Nodes sent to the queue.*

Step 4. *Take a node of the queue Q and go to Step 2.*

The algorithm ends when the queue is empty. The Voronoi diagram approximation generated by the algorithm applied to the VQ of Figure 5(a) is represented in Figure 5(b). Figure 5(c) corresponds to the exact approximation of the output.

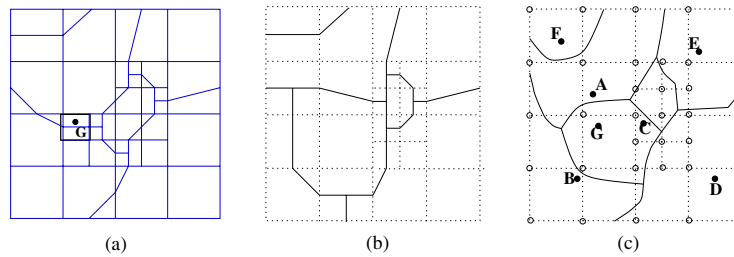


Fig. 5. *(a) Input of the insertion algorithm (b) Voronoi diagram approximation generated by the algorithm. (c) The corresponding exact Voronoi diagram*

3.2 Deletion Algorithm

Given a VQ codification and a site $T =< G_T, D_T, P_T, L_T >$ to be deleted, the deletion algorithm applies a contraction process. The process starts with a node

containing part of the boundary of the Voronoi region of T. Then traverses all the nodes containing part of this boundary. It ends when the node containing T is reached and all the traversed nodes have been processed. Next, we describe the four steps of the algorithm.

Step 1. *Traverse the VQ in a top-down order and stop when a terminal node N containing T is reached. Initialize with null the queue Q used to maintain the nodes to be processed.* In Figure 6(a) and (b) an exact Voronoi diagram and its approximation have been represented. Taking this situation as example and considering G as the label of T, the located node in step 1 is the one represented in Figure 6(c).

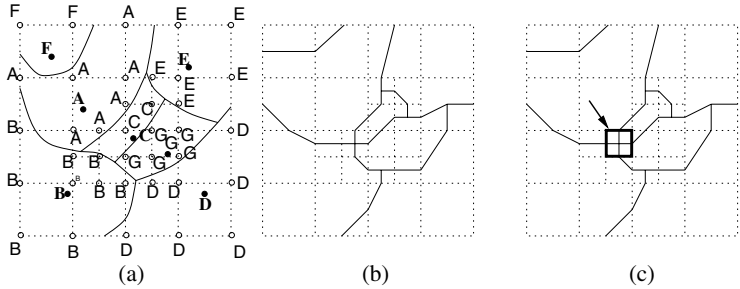


Fig. 6. *(a)(b) Voronoi diagram and its approximation. (c) Identification of the first terminal node that contains the site G.*

Step 2. *Identify the vertices of N with the label L_T. For each one of these vertices determine the new label that has to be assigned.* To carry out this process we take into account the labels of the neighbor nodes with the vertex to be modified in common (see Figure 7(a)). The new label is the label of the nearest site to the vertex (selected from the set of neighbor labels) (see Figure 7(b)).

The assignment of the new label, as in the insertion algorithm, has two major consequences. (i) The configuration of the new labeled node can be modified, from a terminal to a non terminal configuration or from one terminal configuration to another. In both situations the DCEL entries have to be updated. If a non terminal configuration is obtained, a subdivision process has to be applied and the new nodes have to be processed. The case in which a compression can be applied (i.e. when the node and its neighbors can be codified in the parent node with no error) it has also to be considered. (ii) The modification of one vertex label leads to a modification on the configuration of the neighbor nodes with this vertex in common and also the modification of their DCEL entries. The new labels generate the Voronoi diagram approximation represented in Figure 7(c), where the new boundaries correspond to thick lines.

Step 3. *Examine the neighbor nodes of N and send to the queue Q: the nodes with the label L_T and the nodes that have received a new outside as a consequence of the new label assigned to N.* In the Voronoi diagram approximation of Figure

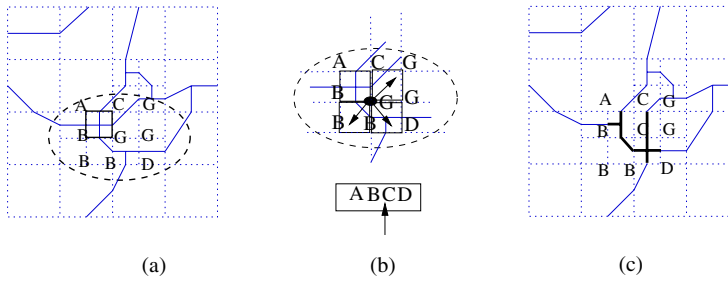


Fig. 7. *(a) Nodes sharing the vertex with label G.(b) The set of possible new labels is composed of* $\{A, B, C, D\}$*. The new selected label is C (c) The introduction of C as the new vertex label, produces new boundaries represented as thick lines.*

8(a) only two neighbor nodes have the label G (see Figure 8(b)). The nodes represented in Figure 8(c) are sent to the queue Q.

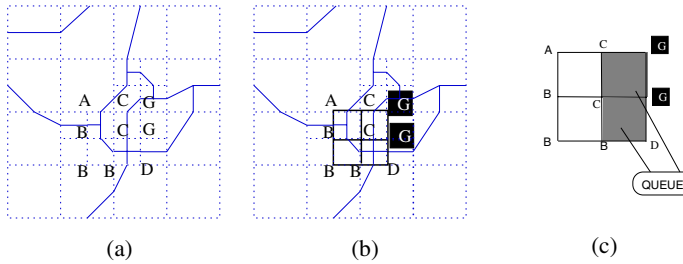


Fig. 8. *(a) Voronoi diagram approximation with the new label assignment. (b) Identification of neighbor nodes with the label of the site to be deleted. (c) Nodes sent to the queue*

Step 4. *Take a node of the queue and go to step 2.*

The algorithm ends when the queue is empty. In Figure 9(b) the approximation generated by the algorithm applied to the VQ of Figure 9(a) is represented. Figure 9(c) corresponds to the exact Voronoi diagram of the output.

4 Computational Cost

The VQ construction algorithm generates $o(\overline{V}2^L)$ expected number of nodes, and runs in $o(Ln + \overline{V}L2^L)$, where n is the number of sites, $\overline{V}$ the length of the boundary of the Voronoi diagram $VD(\mathcal{S})$ and L the level of the maximal subdivision [3]. The term Ln corresponds to the distribution the n sites over the L levels and the term $\overline{V}L2^L$ corresponds to the location in worst time L the

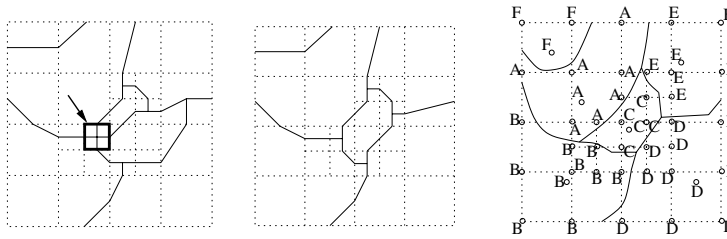


Fig. 9. *(a) Input of the deletion algorithm (b) Output of the deletion algorithm once G has been eliminated. (c) The exact Voronoi diagram of the output*

neighbors of each node. In fact, since the expected time of locating neighbors is approximately $O(4)$ [12], we can assume that the algorithm runs in $o(Ln+\overline{V}2^L)$ expected time.

Let T be a site to be inserted, $\mathcal{S}' = \mathcal{S} \cup \{T\}$ be the new set of sites and $VR(T)$ be the Voronoi region of T in the new Voronoi diagram $VD(\mathcal{S}')$. As the insertion algorithm only processes the terminal nodes that intersect the boundary of $VR(T)$ or the boundary of $VR(T) \cap VD(\mathcal{S})$ (see Figure 10), its computational cost only depends on the sum of the lengths of these boundaries $\overline{V'}$. Then, considering that the number of nodes to be processed is $\overline{V'}2^L$, the algorithm runs in $o(\overline{V'}L2^L)$ expected time. The algorithm does not process all the VQ nodes, therefore the $O(4)$ expected time of locating neighbors is never reached. Moreover, the time for locating neighbors can be improved using pointers between terminal nodes and its neighbors of the same level [12]. In this case the algorithm runs in $o(\overline{V'}H2^L)$ expected time, where H is the maximal difference between the levels of adjacent nodes.

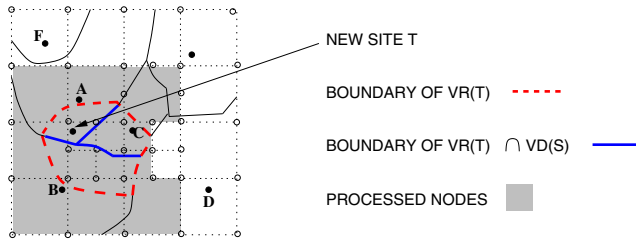


Fig. 10. *Nodes processed by the insertion algorithm.*

Applying a similar strategy, it can be proven that the expected time of the deletion algorithm is $o(\overline{V'}L2^L)$, where T is the site to be deleted, $\overline{V'}$ is the sum of the lengths of the boundaries of $VR(T)$ and $VR(T) \cap VD(\mathcal{S}')$, with $\mathcal{S}' = \mathcal{S} - \{T\}$.

5 Concluding Remarks

The VQ is a hierarchical representation of a Voronoi diagram from which a polygonal approximation, represented by a DCEL structure, of the region boundaries can be computed at different levels of detail. In this paper, we have presented the algorithms for dynamically maintaining, under the insertion or deletion of sites, the VQ and the corresponding DCEL.

The insertion algorithm applies a propagation strategy that starts at the leaf node containing the site to be inserted and then processes all the nodes containing a piece of the boundary of the Voronoi region of this site. The deletion algorithm applies a contraction technique that starts at a node containing part of the boundary of the Voronoi region of the site to be deleted and then processes all the nodes containing part of this boundary. Both algorithms guarantee that only the nodes intersected by the Voronoi region of the sites to be modified are processed. The algorithms have been implemented and tested for different site shapes and distance functions.

Future work will include an extension of our approach for approximating generalized Voronoi diagrams in three-dimensional space.

6 Acknowledgments

This work was supported by DURSI 2001SGR-00296. The first author was supported in part by grants TIC2000-1009 and TIC2001-2226-C02-02. The second and third authors were supported in part by grant TIC2001-2392-C03-01.

References

1. Aurenhammer, F.: Voronoi diagrams: A survey of a fundamental geometric data structure. ACM Computing Surveys, 23(3) (1991) 686-695
2. Aurenhammer, F., Klein, R.: Voronoi diagrams. In: Sack, J. R., Urrutia, J. (eds): Handbook of Computational Geometry, Chapter 5. Elsevier Science Publishers (2000) 201-290
3. Boada, I., Coll, N., Sellarès, J.A.: Hierarchical Planar Voronoi Diagram Approximations. In: Wismath, S. (ed.): Proceedings of the 14th Canadian Conference on Computational Geometry (2002) 40-45
4. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry. Algorithms and applications. Springer-Verlag (2000)
5. Fu, J., Lee, R.: Voronoi diagrams of moving points in the plane. International Journal on Computational Geometry and Applications, 1(1) (1991) 23-32.
6. Gavrilova, M., Rokne, J.: On Dynamic Generalized Voronoi Diagrams in the Euclidean Metric. Lecture Notes in Computer Science 2073, Vol. 1. Springer-Verlag (2001) 673-682.
7. Gold C.: Voronoi Diagrams page on the Web: Applications. http://www.voronoi.com/section_1.htm
8. Hoff, K., Culver, T., Keyser, J., Lin, M., Manocha, D.: Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. Proceedings of SIGGRAPH'99, ACM Press/Addison-Wesley (1999) 277-286

9. Kobayashi, K., Sugihara, K.; Crystal Growth Voronoi Diagram and its Applications to Collision-Free Paths. Lecture Notes in Computer Science 2073, Vol. 1. Springer-Verlag (2001) 738-747
10. Lavender, D., Bowyer, A., Davenport, J., Wallis, A., Woodwark, J.: Voronoi diagrams of set-theoretic solid models. Computer Graphics and Applications, 12(5) (1992) 69-77
11. Okabe, A., Boots, B., Sugihara, K., Chiu, S. N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons (2000)
12. Samet, H.: Applications of Spatial Data Structures: computer graphics, image processing, and GIS. Addison-Wesley (1990)
13. Teichmann, M., Teller, S.: Polygonal approximation of Voronoi diagrams of a set of triangles in three dimensions. Technical Report 766, Laboratory of Computer science, MIT (1997)
14. Vleugels, J., Overmars, M.: Approximating Generalized Voronoi Diagrams in Any Dimension. International Journal on Computational Geometry and Applications, 8 (1998) 201-221

Voronoi Diagram of Circles in a Large Circle

Deok-Soo Kim[1], Donguk Kim[1], and Kokichi Sugihara[2]

[1] Department of Industrial Engineering, Hanyang University,
17 Haengdang-Dong, Seongdong-Ku, Seoul 133-791, Korea
dskim@hanyang.ac.kr,
donguk@ihanyang.ac.kr
[2] Department of Mathematical Informatics, Graduate School of
Information Science and Technology, University of Tokyo,
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. Presented in this paper is an algorithm to compute a Voronoi diagram of circles contained in a sufficiently large circle, where the interior circles are not necessarily disjoint. Given circles in a large circle, the region in the large circle is divided into regions associated with the circles. The algorithm first constructs the ordinary point Voronoi diagram of centers of inner circles. Then, it modifies the point Voronoi diagram considering the large enclosing circle so that the seed topology for the desired Voronoi diagram is obtained by a number of edge-flip operations. After getting the correct topological information, the equations of edges are computed in a rational quadratic Bézier curve form.

1 Introduction

Suppose that circles contained in a larger circle are given on a plane, where the radii of the circles are not necessarily equal. One of the circles containing the smaller circles is called an *encloser* meaning that it encloses the smaller ones which are called *inner circles*. Given this circle set, we assign every location on a plane to the closest member in the set of circles. This forms a tessellation of the encloser and called it *a circle Voronoi diagram in a circle*.

The distance metric employed in this paper is an ordinary Euclidean L_2-distance. In addition, the distance between a point and a circle is the minimum distance from a point to the boundary of the circle. For the convenience of the presentation, we assume that the degree of a vertex is always three since the other degenerate cases can be transformed into such cases using Voronoi edges of length zero [11]. We also assume that there are at least two circles are contained in the encloser. The Voronoi diagram of circles in another circle can be very useful when the minimum circle enclosing a set of circles is to be efficiently computed. For example, we may want to compute the radius of a tube containing a set of wire bundles.

As is frequently recommended [8][9], the underlying data structure to represent the topology of Voronoi diagram in this paper is the well-known and efficient winged-edge data structure. Note that, in this paper, the terms of *vertex* and *edge* mean the corresponding Voronoi vertex and Voronoi edge, respectively.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 847-855, 2003.
© Springer-Verlag Berlin Heidelberg 2003

This paper is organized as follows. Section 2 shows the basic concept of the proposed algorithm. In Section 3, we discuss the construction of the correct topology of the desired Voronoi diagram. The computation of vertex positions and edge equations are discussed in Section 4. Section 5 presents implementation issues and shows some examples. Then, we conclude this paper.

2 Basic Idea

The overall approach of the proposed algorithm is as follows. First, we compute a *seed topology*, which is incorrect for the result in general, for the desired Voronoi diagram. Then, the seed topology is refined by a number of *edge-flip operations* so that the correct topology can be obtained. Lastly, the equations of edges are computed.

Computing a seed topology also consists of two steps. Let *center points* denote the centers of inner circles. First, we compute the ordinary point Voronoi diagram of the center points. Then, this ordinary Voronoi diagram is trimmed against the encloser to get the topological constructs of vertices defined by encloser and appropriate inner circles. Lastly, the necessary topological constructs of edges are also defined by connecting appropriate vertices.

The idea of edge-flipping has appeared in a number of previous researches. Sibson constructed the Delaunay triangulation by flipping edges of triangles from an arbitrary triangulation [10], Edelsbrunner showed that $O(n^2)$ edge-flips are enough to obtain the Delaunay triangulation from any triangulation [3]. Adopting this edge-flip approach, we have recently reported an exact and efficient algorithm to compute the Voronoi diagram of circles from the Voronoi diagram of centers of the circles [6]. In that paper, we firstly computed a Voronoi diagram of center of circles and then updated the topology by a series of edge-flips.

The problem we solve in this paper adds a few more issues to be handled in addition to the previously reported Voronoi diagram of circles. In the topology part, the constructs for Voronoi edges and vertices should be created appropriately by trimming the Voronoi diagram of center points against encloser. On the other hand, in the geometry part, there could be a situation that the correct vertex position can be defined by two inner circles as well as an encloser. Note that this configuration is different from the situation with circumcircle of three inner circles. Similarly, there could be cases that the edge equation can be elliptic in addition to hyperbolic.

On the other hand, the proposed algorithm inherits its numerical stability from our previous algorithm since this approach used only edge-flipping from the seed topology.

3 Topology Construction

Topology construction consists of the procedures of a seed construction and a topology update. To get the seed topology, we firstly compute the ordinary Voronoi diagram of center points. Then, the ordinary Voronoi diagram is trimmed against encloser and the necessary topological constructs are created. This modified Voronoi

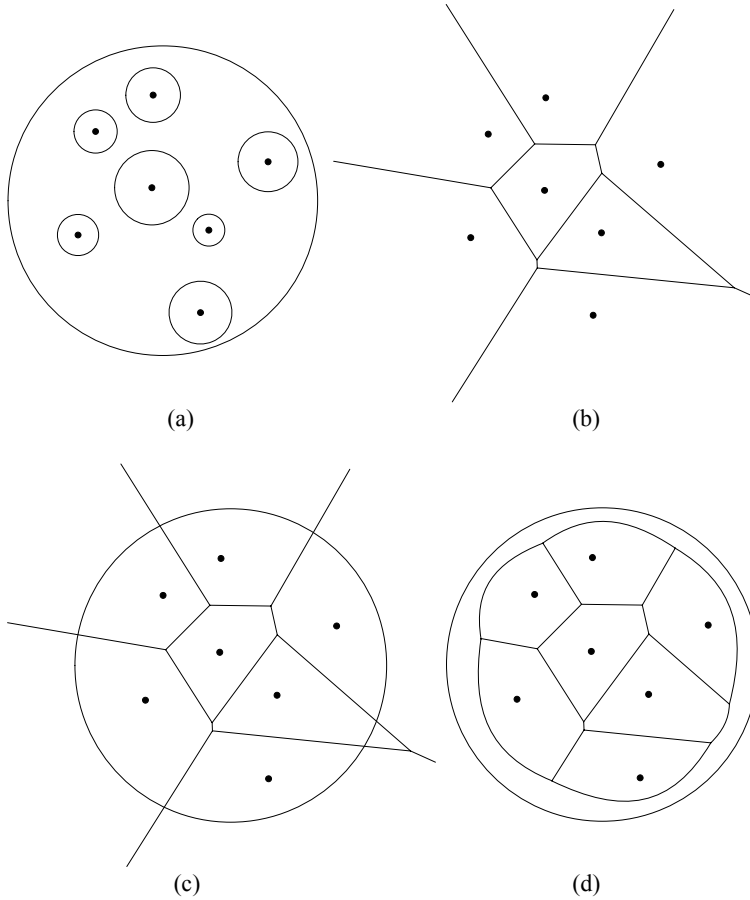


Fig. 1 Procedure for constructing a seed topology: (a) Input circles, (b) Voronoi diagram for center points, (c) the Voronoi diagram and the encloser, (d) Constructed seed topology.

diagram is then refined to have the correct topology of desired Voronoi diagram by a number of edge-flipping operations.

3.1 Seed Topology Construction

Illustrated in Fig.1 is the procedure for constructing a seed topology. Suppose that circles shown in Fig.1(a) are input circles: seven inner circles and an encloser. We compute the ordinary point Voronoi diagram of center points as shown in Fig.1(b). Note that the problem of computing the point Voronoi diagram has been extensively studied by many researchers and numerically robust and efficient codes are available [12][13]. In our implementation of the proposed algorithm, in fact, we have used a code written by one of the authors. Note that this code employed the exact computation scheme. Fig.1(c) simultaneously shows the point Voronoi diagram and the encloser. Since the topological information about the encloser should be contained

in the seed topology as well, it is necessary to merge the point Voronoi diagram and the encloser. Our approach of merging them is as follows. First of all, we compute intersections between edges of ordinary Voronoi diagram and the encloser. Second, we create new constructs of vertices resulting from the intersection process. Then, constructs for new edges are created by connecting two new adjacent vertices in a counterclockwise order along the boundary of encloser. Hence, new edges form a chain and define the boundary of encloser. Note that the part of edges as well as vertices, exterior to the encloser, of Voronoi diagram of center points are trimmed away in the middle of the process.

As a result, every circle including encloser has its own topologically bounded *Voronoi region* as shown in Fig.1(d). In this figure, the curved edges are newly born edges. Note that geometric information, e.g. vertex positions and edge equations, does not have to be considered in the construction of a seed topology.

Advantages for using a point Voronoi diagram to get a seed topology are in its structural simplicity and similarity between the seed and the resulting correct Voronoi diagram. The former property is well used to get a seed topology because intersections between a line and a circle can be easily computed. Note that this structural simplicity makes the proposed algorithm robust. The latter property will be discussed in the next subsection.

3.2 Topology Update

The approach to modify the seed topology to the correct one in the proposed algorithm is similar to the one used in the authors previous article [6]. In overall, the correct topology is obtained by a recursive edge-flip method. In this scheme, finding the edge-flip condition becomes a key part in this topology update procedure.

Even though the details of the edge-flipping operation are discussed in [6], we briefly summarize them here for the completeness of this article. The edge-flip criterion we have adopted is as the following. An edge should be flipped if tangent circles exist at both end vertices of the edge and both tangent circles intersect with their *mating circles*. Note that the mating circle at a vertex, when we decide an edge s flippability, means a circle which is not in the three surrounding circles of the vertex but among three circles around the other end vertex of the edge.

This simple criterion is applied to all of the edges. If an edge satisfies the flipping condition, the edge is flipped and the four neighboring edges connected to the flipped one are recursively tested for the flippability. After performing this operation for all edges the correct topology of circle Voronoi diagram can be constructed. Note that the upper bound of the time taken in the updating process of topology by this scheme is $O(n^2)$ as the similar to its counterpart for circle Voronoi diagram [6].

4 Geometry Construction

The computation of vertex positions depends on the tangent circles to a given circle configuration, and the computation of edge equations depends on the vertex positions.

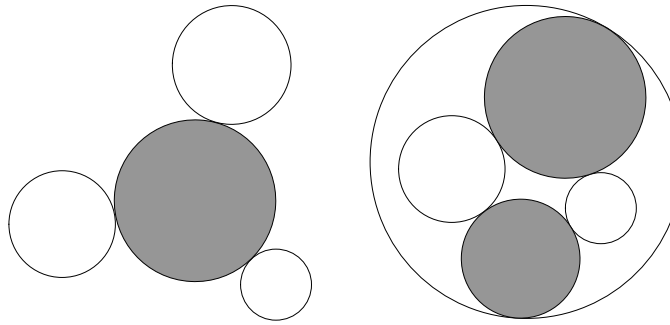


Fig. 2 Tangent circles. White circles are given circles. (a) A circumcircle. (b) Circles tangent inside an enclosing circle and outside two given circles.

Explained in this section are the computations of edge equations as well as vertex positions. Note that the computation of vertex position is a core part of computing the correct topology from a seed topology.

4.1 Vertex Geometry

The problem of computing the position of a vertex is equivalent to the problem of finding a tangent circle to three given circles around the vertex. This problem has been known as Apollonius□tenth problem [1][2].

In the course of computing the tangent circles in this problem, two different situations occur as illustrated in Fig.2. Shown in Fig.2(a) is the case that the tangent circle, denoted as a gray one, is tangent to three inner circles from outside. This is the case discussed in detail in our previous paper [7] and we called this tangent circle a circumcircle. The other case, illustrated in Fig.2(b), is the case that the desired tangent circle is tangent to two inner circles as well as the encloser. Note that there can be two instances of such tangent circles in this configuration. This configuration never appeared in the computation of a plain circle Voronoi diagram.

While the tangent circle in Fig.2(a) can be computed by using the inversion on a complex plane [7], the tangent circles in Fig.2(b) can be solved differently. There are two different approaches to find the tangent circles in Fig.2(b). First, it can be directly found by solving quadratic simultaneous equations in the given Euclidean space after a few simple transformations. On the other hand, it can be found by solving a quartic equation formulated as an intersection between two quadratic equations representing the equations of Voronoi edges. Note that a quartic equation can be easily solved by employing either Ferrari formula or a numerical process [5].

Note that there are always two instances of tangent circles for the situation in Fig.2(b), while the situation in Fig.2(a) is not. Hence, it is necessary to decide which one is the correct one for the current vertex. This question can be easily answered by checking the orientation among two inner circles and encloser around the topology definition of vertex. If the order of circles obtained from the topology of vertex is

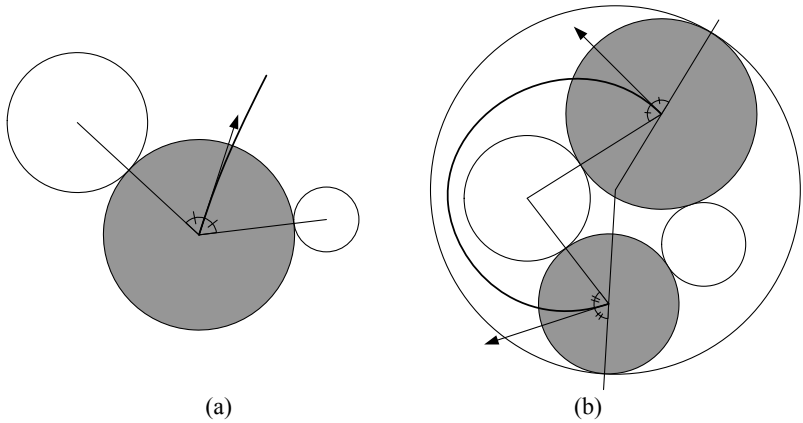


Fig. 3 Tangent vectors. (a) tangent vector computation between two inner circles, and (b) tangent vector between an inner circle and the encloser.

identical to the order of the circles on the boundary of a tangent circle, then this tangent circle is the correct one.

4.2 Edge Geometry

The geometry of an edge in our problem, a circle Voronoi diagram in a circle, can be a hyperbolic arc, an elliptic arc, or a straight line. When an edge is defined between two inner circles, it can be either a hyperbolic arc or a straight line, while an edge between encloser and an inner circle can only be an elliptic one.

It is known that a segment of conics can be exactly represented by a *rational quadratic Bézier form* [4]. In addition, the rational quadratic Bézier form of a conic curve segment can be computed by knowing five parameters: the positions of both end vertices of the segment, the tangent vectors at these vertices, and an arbitrary point through which the curve passes.

The positions of both end vertices correspond to the centers of tangent circles as was explained in the above subsection. Tangent vector at a vertex can be computed as the angle bisector of two lines passing through the vertex and center points of two involved circles as illustrated in Fig. 3. A passing point can be also easily obtained by taking a point equidistant from two circles on a line passing through the centers of the involved circles. Shown in Fig.3(a) is when the curve is defined between two inner circles where Fig.3(b) shows the computation of edge equation defined between an inner circles and the encloser.

In our implementation, we have represented the curve as a standard form of rational quadratic Bézier [4], i.e. weights of the first and the last are fixed to one. If a passing point is located in a triangle formed by the control points, weight of the second control point becomes positive. In addition, if a passing point is not on the desired curve segment, we can simply obtain desired curve segment by switching the sign of the second weight. Note that the second weight can become negative when the curve is elliptic.

5 Examples

The algorithm for computing circle Voronoi diagram in a circle has been implemented in C++ and has been tested on Intel Pentium IV 2.4 GHz processor. Fig. 4 shows examples of resulting Voronoi diagrams for non-intersecting random circles in an enclosing circle. The number of inner circles in Fig.4(a), 4(b), 4(c), and 4(d) are 10, 100, 500, and 1000, respectively.

Fig. 5 shows the computation time behavior with respect to the number of inner circles. In the figure, the computation time is divided into two stages: the time for seed topology construction and the time for the complete Voronoi diagram

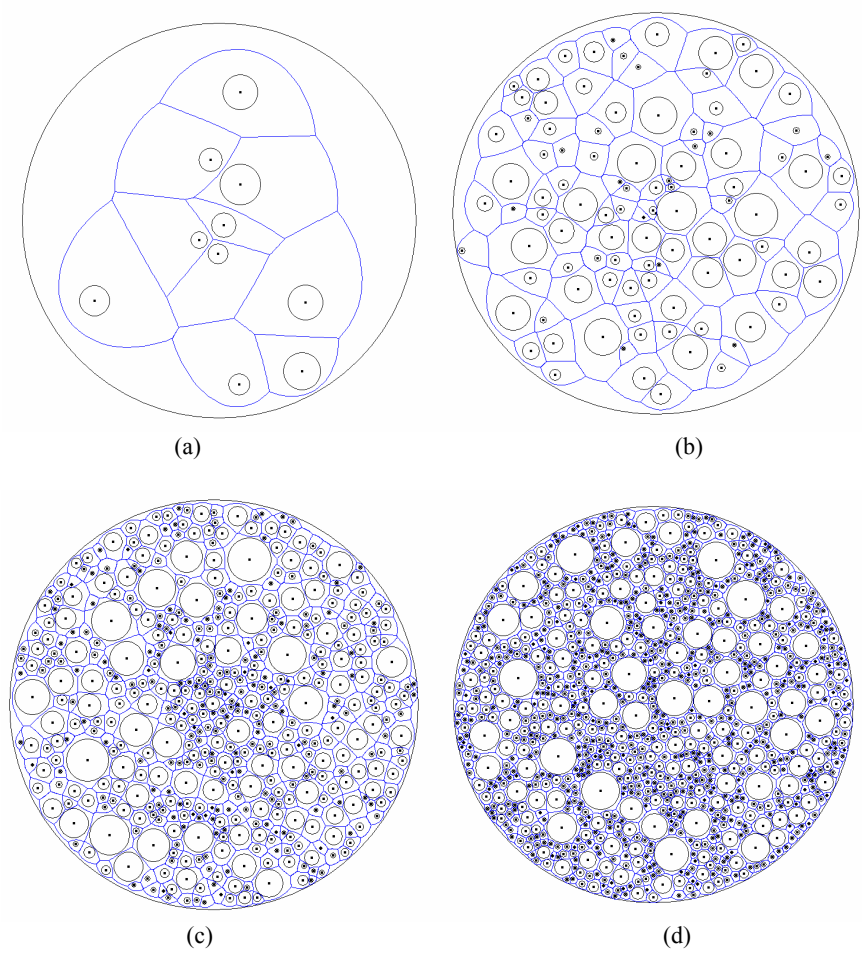


Fig. 4 Example of the resulting Voronoi diagrams. Non-intersecting random circles are contained in an enclosing circle. (a) 10 circles, (b) 100 circles, (c) 500 circles, and (d) 1,000 circles.

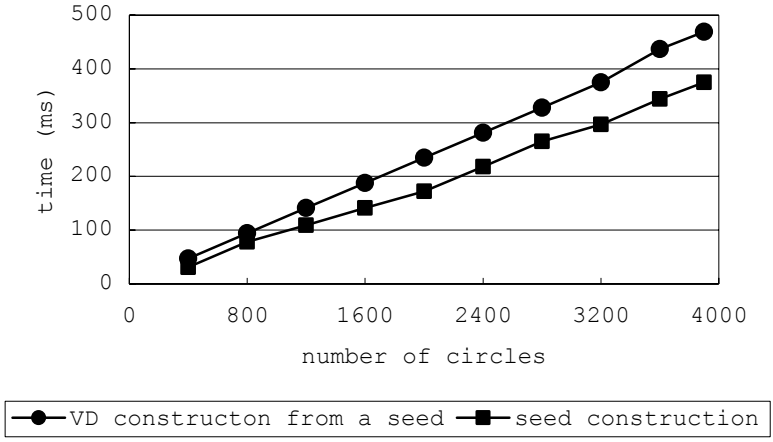


Fig. 5 Computation time behavior.

construction after the seed topology computation. Hence, the time necessary to compute equations of edges, positions of vertices, and a correct topology are included in the latter time. Therefore, the total computation time to construct a whole circle Voronoi diagram in a circle is the accumulation of both computation times. As shown in the figure, we want to emphasize that the total computation time for 4,000 inner circles took less than one second in the given computing environment.

6 Conclusions

Presented in this paper is an algorithm for computing a circle Voronoi diagram in an enclosing circle. In order to obtain a seed topology we compute the point Voronoi diagram for the centers of inner circles. Then, we have introduced new Voronoi vertices and edges bounding the enclosing circle by considering the spatial relationship between the point Voronoi diagram and the enclosing circle. After finding a proper seed topology, the algorithm transforms the seed topology to the correct topology of the desired Voronoi diagram by a number of edge-flipping operations. After the correct topology is constructed, the geometric information of the edges is computed and the Voronoi edges are represented in a rational B□zier curve form.

Acknowledgements

The first author was supported by the Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center (CPRC) at Hanyang

University, and the third author was supported by Grant-in-Aid for Scientific Research of the Japanese Ministry of Education, Science, Sports and Culture.

References

1. Boyer, C.B.: A History of Mathematics, Wiley, New York (1968).
2. Dörrie, H.: 100 Great Problems of Elementa ry Mathematics -Their History and Solutions, Dover, New York (1965).
3. Edelsbrunner, H.: The computational geometry column, EATCS (European Association for Theoretical Computer Science) Bulletin, Vol. 37 (1988) 109-116.
4. Farin, G.: Curves and Surfaces for Computer-Aided Geometric Design -A Practical Guide, 4th edn. Academic Press, San Diego (1996).
5. Kim, D.-S., Lee, S.-W., Shin, H.: A cocktail algorithm for planar Bézier curve intersections, Computer-Aided Design 30, (1998) 1047-1051.
6. Kim, D.-S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology, Computer Aided Geometric Design 18 (2001) 541-562.
7. Kim, D.-S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry, Computer Aided Geometric Design 18 (2001) 563-585.
8. Mäntyl, M.: An introduction to solid modeling, Computer Science Press (1988).
9. Preparata, F.P., Shamos, M.I.: Computational Geometry -An Introduction, Springer-Verlag, New York (1985).
10. Sibson, R.: Locally equiangular triangulations, The Computer Journal 21 (1978) 243-245.
11. Sugihara, K.: A simple method for avoiding numerical errors and degeneracy in Voronoi diagram construction, IEICE Transactions of Fundamentals E75-A (1992) 468-477.
12. Sugihara, K., Iri, M.: Construction of the Voronoi diagram for one million generators in single-precision arithmetic, Proc. IEEE 80 (1992) 1471-1484.
13. Sugihara, K.: http://www.simplex.t.u-tokyo.ac.jp/~sugihara/ (2003).

On Estimating Result Sizes of Multi-way Spatial Joins

Ho-Hyun Park

School of Electrical and Electronics Engineering, Chung-Ang University,
221, HukSuk-Dong, DongJak-Ku, Seoul 156-756, Korea
hohyun@cau.ac.kr

Abstract. In a real life environment, spatial data is highly skewed. In general, there are two kinds of skews in spatial data. One is the *placement skew* and the other is the *area skew*. This paper introduces methods and the characteristics of estimating the accurate result sizes of the multi-way join for the area skewed spatial data. Especially, this paper describes the number and sort of the statistics which the optimizer should keep in order to calculate the multi-way join result size. Experimental results show our approach generally provides accurate estimation for the multi-way join for the area skewed spatial data.

1 Introduction

The spatial join is a common spatial query type which requires high processing cost due to the high complexity and large volume of spatial data. Therefore, the exact estimation of the result size of the spatial join has a great influence on the query optimizer and the spatial database management system. The n-way (n≥2) spatial join combines n spatial relations using $n-1$ or more spatial predicates. An example of a 3-way spatial join is "Find all buildings which are adjacent to roads that intersect with boundaries of districts."

In a real life environment, spatial data is highly skewed. In general, there are two kinds of skews in spatial data. One is the *placement skew* and the other is the *area skew* [1].[1]

Recently Papadias et. al [6] proposed formulae of estimating the result size of the multi-way spatial join for two kinds of query types, *tree* and *clique*. However, the formulae considered only the placement distribution of spatial data by assuming that the placements of all spatial objects are uniformly distributed. Therefore, the formulae of [6] are satisfied only in some special cases of the area distribution.

This paper extends the formula to the arbitrary area of spatial data, and then analyzes the characteristics and complexities of the extended formula. Especially, we describe the number and sort of the statistics which the optimizer should keep in order to calculate the join result size from the formula for each join

[1] Originally in [1], the terminology *size skew* was used. However we use *area skew* instead of *size skew* to avoid confusion with the *size* of "join result size".

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 856–865, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Table 1. Notations

symbol	description
r_i	relation i
$s_{r_i,j}$	j-th object in r_i
$s_{r_i,j,x}$	x-length of j-th object in r_i
$s_{r_i,j,y}$	y-length of j-th object in r_i
$s_{r_i,j,x^k y^l}$	(x-length)^k*(y-length)^l of j-th object in r_i $(= s^k_{r_i,j,x} * s^l_{r_i,j,y})$
q_x	x-length of query window q
q_y	y-length of query window q
N_{r_i}	number of objects in r_i
$\bar{s}_{r_i,x}$	average x-length for all objects in r_i
$\bar{s}_{r_i,y}$	average y-length for all objects in r_i

query type. Experimental results show while the estimation using the formulae of [6] gives very high error rates or is nearly unacceptable in some cases, our approach generally provides accurate estimation results.

2 Background

In this paper, we consider spatial objects only in the two dimensional unit space $[0, 1)^2$, and assume that the objects are rectangular, i.e., consider only the MBR (Minimum Bounding Rectangle) of a real object. The notations to be used in this paper are summarized in Table 1.

Under the assumption that the placement distribution is uniform, the formula for estimating the result size of the window query on r_i is shown in the following formula [4,5]:

$$Size(\sigma_q(r_i)) = \sum_{j=1}^{N_{r_i}} (s_{r_i,j,x} + q_x) * (s_{r_i,j,y} + q_y) \tag{1}$$

In the above formula, a query optimizer does not know object information such as $s_{r_i,j,x}$ and $s_{r_i,j,y}$ in optimization time. For the estimation of the query result size, a query optimizer generally keeps statistics such as $N_{r_i}, L_{r_i}x, L_{r_i}y$ and $L_{r_i}xy$ in the system catalog. Therefore, we must transform the above formula into the expression in terms of statistics as follows [4]:

$$Size(\sigma_q(r_i)) = L_{r_i}xy + q_x * L_{r_i}y + q_y * L_{r_i}x + N_{r_i} * q_x * q_y \tag{2}$$

The following shows the formula for estimating the result size of the spatial join between r_i and r_j under the uniform assumption of the placement distribution [2,9]:

$$Size(r_i \bowtie_{intersect} r_j) = \sum_{k=1}^{N_{r_i}} \sum_{l=1}^{N_{r_j}} (s_{r_i,k,x} + s_{r_j,l,x}) * (s_{r_i,k,y} + s_{r_j,l,y}) \tag{3}$$

The following expression is the transformation of the above formula in terms of the expression of statistics:

$$Size(r_i \bowtie_{intersect} r_j) = L_{r_i}xy * N_{r_j} + L_{r_i}x * L_{r_j}y + L_{r_i}y * L_{r_j}x + N_{r_i} * L_{r_j}xy \quad (4)$$

Equation (2) and Equation (4) do not assume anything about the area of spatial data, i.e., the above expressions can be applied to the arbitrary area distribution of spatial data. In Equation (2) and Equation (4), we need 4 statistics per relation to estimate the result size of the window query and the spatial join (e.g., for r_i, $N_{r_i}, L_{r_i}x, L_{r_i}y$ and $L_{r_i}xy$). If we assume that the x-length distribution is independent of the y-length distribution, we need only 3 statistics per relation because $L_{r_i}xy$ can be represented by $L_{r_i}x * L_{r_i}y/N_{r_i}$ (Remind that if random variables X and Y are mutually independent, $E(XY) = E(X) * E(Y)$).

3 Result Sizes for Multi-way Spatial Joins

First, we define some terminologies. A *term* is a component of an expression. An *atomic term* is an atomic component of an expression such as $s_{r_i,j,x}, q_y$ and $L_{r_i}xy$ in Equation (1) and Equation (2). Among atomic terms, object information $s_{r_i,j,x^k y^l}$ is called the *object term*, and statistics information $L_{r_i}x^k y^l$ the *statistics term*. A *compound term* is a composition of terms by operators such as $+$ and $*$. For example, $s_{r_i,j,x} + q_x$ and $q_x * s_{r_i,j,y}$ are compound terms. Among compound terms, if the outer operators are $+$, the term is called the *addition term*, and if $*$, the *production term*. For example, Equation (4) is an addition term which consists of 4 production terms each of which consists of 2 statistics terms.

The multi-way spatial join can be modeled by a query graph whose node represents a relation and edge represents a spatial relationship. We treat only the tree type (acyclic query graph) and the clique type (complete query graph) as in the case of [6] among various query graphs. First, we focus on the tree typed query graph. Under the assumption that the placement distribution is uniform, Papadias et al. [6] estimated the result size of the multi-way intersection join among n relations for the tree typed query as the following formula:

$$Size(Q) = \prod_{k=1}^{n} N_{r_k} * \prod_{\forall i,j:Q(r_i,r_j)=1} (\bar{s}_{r_i,x} + \bar{s}_{r_j,x}) * (\bar{s}_{r_i,y} + \bar{s}_{r_j,y}) \quad (5)$$

In the above formula, Q is a query graph which represents a tree typed multi-way spatial join among relations $r_1, \ldots, r_n$, and $Q(r_i, r_j) = 1$ means there is an intersection relationship between r_i and r_j. According to Papadias et al. [6], the above formula is an extension of the formula of the 2-way join. However, this extension has a serious restriction. Equation (5) first approximates the x-lengths and y-lengths of all data to the average value per axis. As it will be explained later in this section, the above formula is satisfied only in the case that the x-lengths and y-lengths of all data per relation are equal. If we extend Equation (3) without such an approximation, we obtain the following formula:

$$Size(Q) = \sum_{k_1=1}^{N_{r_1}} \cdots \sum_{k_n=1}^{N_{r_n}} \prod_{\forall i,j:Q(r_i,r_j)=1} (s_{r_i,k_i,x} + s_{r_j,k_j,x}) * (s_{r_i,k_i,y} + s_{r_j,k_j,y}) \quad (6)$$

In the above formula, the inner product term $\prod_{\forall i,j:Q(r_i,r_j)=1}(s_{r_i,k_i,x} + s_{r_j,k_j,x}) * (s_{r_i,k_i,y} + s_{r_j,k_j,y})$ is the probability with which an arbitrary n-tuple $s_{r_1,k_1}, \ldots, s_{r_n,k_n}$ from relations $r_1, \ldots, r_n$ satisfies the query Q. We call the probability the *selectivity* S for Q. For the estimation of the query result size, Equation (6) should also be derived using the statistics terms. If we derive the formula of Equation (6) using the statistics terms in a tree typed 3-way join ($r_1 \bowtie r_2$ and $r_1 \bowtie r_3$), the following expression is obtained:

$$\begin{aligned} Size(Q_3) = {} & L_{r_1}x^2y^2 * N_{r_2} * N_{r_3} + L_{r_1}x^2y * N_{r_2} * L_{r_3}y + L_{r_1}xy^2 * N_{r_2} * L_{r_3}x \\ & + L_{r_1}xy * N_{r_2} * L_{r_3}xy + L_{r_1}x^2y * L_{r_2}y * N_{r_3} + L_{r_1}x^2 * L_{r_2}y * L_{r_3}y \\ & + L_{r_1}xy * L_{r_2}y * L_{r_3}x + L_{r_1}x * L_{r_2}y * L_{r_3}xy + L_{r_1}xy^2 * L_{r_2}x * N_{r_3} \\ & + L_{r_1}xy * L_{r_2}x * L_{r_3}y + L_{r_1}y^2 * L_{r_2}x * L_{r_3}x + L_{r_1}y * L_{r_2}x * L_{r_3}xy \\ & + L_{r_1}xy * L_{r_2}xy * N_{r_3} + L_{r_1}x * L_{r_2}xy * L_{r_3}y + L_{r_1}y * L_{r_2}xy * L_{r_3}x \\ & + N_{r_1} * L_{r_2}xy * L_{r_3}xy \end{aligned} \quad (7)$$

Equation (7) is more complex than Equation (2) and Equation (4). The statistics for r_2 and r_3 are just $N_{r_i}, L_{r_i}x, L_{r_i}y$ and $L_{r_i}xy$ as in the case of the window query and the 2-way join. However, there are more statistics for r_1.

As we show in Equation (7), if we transform the formula of object terms for a tree typed multi-way spatial join into that of statistics terms, the resulting expression is a complex addition term of production terms of statistics terms. Now, we have two questions. If we derive the formula of Equation (6) using statistics terms, how many terms do we have? And how many statistics should we keep per relation in order to estimate the result size of the tree typed multi-way spatial join? The following two lemmas give the answers. Due to the space limitation, we will skip the proofs for all lemmas in this paper. For the details of the proofs, refer to [8].

Lemma 1 *If we transform Equation (6) into the statistics terms, the resulting expression is an addition term which consists of distinct* 4^{n-1} *production terms each of which consists of distinct* n *statistics terms each of which is from* r_i *(*$1 \leq i \leq n$*).*

In the 5-way join, the statistics expression consists of $4^4 = 256$ production terms, and each production term consists of 5 statistics terms. Therefore, the optimizer needs memory for saving $4^4 * 5 = 1,280$ statistics terms. And it also needs computation time for $4^4 - 1 = 255$ additions and $4^4 * 4 = 1024$ productions.

Lemma 2 *If* Q *is a query graph representing a tree typed multi-way spatial join of* n *relations and* r_i *is a node in* Q *representing a relation, and* δ_{r_i} *is the number of edges incident on* r_i*, the following* $(\delta_{r_i} + 1)^2$ *statistics for* $r_i (1 \leq i \leq n)$ *are needed to estimate the result size of* Q*:*

$$\begin{array}{ccccc} N_{r_i} & L_{r_i}x & L_{r_i}x^2 & \cdots & L_{r_i}x^{\delta_{r_i}} \\ L_{r_i}y & L_{r_i}xy & L_{r_i}x^2y & \cdots & L_{r_i}x^{\delta_{r_i}}y \\ L_{r_i}y^2 & L_{r_i}xy^2 & L_{r_i}x^2y^2 & \cdots & L_{r_i}x^{\delta_{r_i}}y^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{r_i}y^{\delta_{r_i}} & L_{r_i}xy^{\delta_{r_i}} & L_{r_i}x^2y^{\delta_{r_i}} & \cdots & L_{r_i}x^{\delta_{r_i}}y^{\delta_{r_i}} \end{array}$$

In Equation (7), since the degree of r_1 is 2, the statistics for r_1 are the following:

$$\begin{array}{ccc} N_{r_1} & L_{r_1}x & L_{r_1}x^2 \\ L_{r_1}y & L_{r_1}xy & L_{r_1}x^2y \\ L_{r_1}y^2 & L_{r_1}xy^2 & L_{r_1}x^2y^2 \end{array}$$

Among the tree typed query graphs with n nodes, if the topology of the query graph is a chain, we need $3^2 = 9$ statistics for $n-2$ central nodes and $2^2 = 4$ statistics for 2 terminal nodes to estimate the n-way join result size. For star topology, we need n^2 statistics for 1 central node and $2^2 = 4$ for $n-1$ terminal nodes. Therefore, in the worst case, n^2 statistics are needed for the tree typed query.

If the x-length distribution is independent of the y-length distribution, we need only $(2 * \delta_{r_i} + 1)$ statistics such as $N_{r_i}, L_{r_i}x, L_{r_i}x^2, \ldots, L_{r_i}x^{\delta_{r_i}}, L_{r_i}y, L_{r_i}y^2, \ldots, L_{r_i}y^{\delta_{r_i}}$ because $L_{r_i}x^k y^l = L_{r_i}x^k * L_{r_i}y^l / N_{r_i}$. Furthermore, if both x-length and y-length comply known distributions, respectively, we need only 3 statistics such as $N_{r_i}, L_{r_i}x, L_{r_i}y$ per relation because the others can be derived using the distribution function and the 3 statistics. However, the x-length and y-length of most real spatial data are not independent and do not comply a specific distribution. If $(\bar{s}_{r_i,x})^k = L_{r_i}x^k/N_{r_i}$ and $(\bar{s}_{r_i,y})^l = L_{r_i}y^l/N_{r_i}$, Equation (5) and Equation (6) are the same. However, this is only the case in which all data lengths for x-axis and y-axis are fixed to $\bar{s}_{r_i,x}$ and $\bar{s}_{r_i,y}$, respectively.

Now, we consider the result size for the clique typed query. Under the uniform assumption of the placement distribution, Papadias et al. [6] also estimated the result size of the clique typed spatial join as the following formula:

$$Size(Q) = \prod_{k=1}^{n} N_{r_k} * \Big(\sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} \bar{s}_{r_j,x} \Big) * \Big(\sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} \bar{s}_{r_j,y} \Big) \tag{8}$$

In the above formula, Q represents a clique typed multi-way spatial join among relations $r_1, \ldots, r_n$. As in the case of Equation (5), the average data length per axis is first applied. If we also rewrite the above formula without such an approximation, we obtain the following formula:

$$Size(Q) = \sum_{k_1=1}^{N_{r_1}} \cdots \sum_{k_n=1}^{N_{r_n}} \Big(\sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} s_{r_j,k_j,x} \Big) * \Big(\sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} s_{r_j,k_j,y} \Big) \tag{9}$$

Lemma 3 *If we transform Equation (9) into the statistics terms, the resulting expression is an addition term which consists of distinct n^2 production terms and each production term consists of n statistics terms.*

Lemma 4 *If Q is a query graph representing a clique typed multi-way spatial join of n relations and r_i is a node in Q representing a relation, the following 4 statistics per $r_i (1 \leq i \leq n)$ are needed to estimate the result size of Q:*

$$N_{r_i}\ L_{r_i}x\ L_{r_i}y\ L_{r_i}xy$$

From the above two lemmas, we are able to know that the complexity of the result size estimation for the clique typed query is simpler than that for the tree typed query. In Lemma 4, if the x-length and y-length distributions are independent, we do not need the $L_{r_i}xy$ statistics because it is calculated by $L_{r_i}x * L_{r_i}y/N_{r_i}$. And in such a case, Equation (8) and Equation (9) give the same result.

4 Experiments

To measure the differences between Equation (5) and Equation (6) and between Equation (8) and Equation (9), we conducted some experiments using generated data sets and real data sets.

4.1 Generated Data Sets

As the first data set, we generated several data sets each of which consists of 10000 rectangles. The placements of the rectangles in a data set are uniformly distributed on domain size (100000,100000). The x-length and y-length of the rectangles are also uniformly distributed on 0 to 1000. Using the data sets, we estimated the number of solutions for tree and clique typed multi-way spatial join queries. To obtain the real execution results for the multi-way spatial join, we performed the M-way R-tree join (MRJ) presented in [7].

Figure 1(a) shows estimation results and the real execution result for the join queries. "Est_All" stands for the estimation result using all statistics shown in Lemma 2 and Lemma 4, and "Est_Avg" stands for the estimation result using only the number of rectangles and the average x-length and y-length (i.e., N_{r_i}, $\bar{s}_{r_i,x}$ and $\bar{s}_{r_i,y}$). "Real" stands for the real execution result. For chain and star queries, Est_All estimates query results well, while Est_Avg shows large difference from real execution results when the number of relations (M) increases. The difference is larger in the star query than in the chain query. This is because the star query needs more statistics for central relations than the chain query. In the clique query, there are little difference among all estimation results and the real execution result. This is because the x-length and y-length distributions are mutually independent in our generated data sets and in this case Equation (8) and Equation (9) give nearly the same result.

M	3	4	5	6
Est_All	11267	12813	14583	16587
Est_Avg	9564	9354	9148	8947
Real	11503	13206	15151	17569

Chain

M	3	4	5	6
Est_All	114690	108594	321008	429911
Est_Avg	3646	1021	286	80
Real	120805	109945	345919	383199

Chain

M	3	4	5	6
Est_All	11267	14724	21157	32585
Est_Avg	9564	9354	9148	8947
Real	11359	15054	21785	35730

Star

M	3	4	5	6
Est_All	114690	5.25E+6	3.73E+8	3.2E+10
Est_Avg	3646	1021	286	80
Real	108195	4.74E+6	3.30E+8	2.6E+10

Star

M	3	4	5	6
Est_All	5394	2349	899	317
Est_Avg	5380	2338	893	314
Real	5551	2368	870	339

Clique

M	3	4	5	6
Est_All	5057	1126	225	42
Est_Avg	2051	255	28	3
Real	5091	1176	241	44

Clique

(a) Placement Uniform, Area Uniform (b) Placement Uniform, Area Skewed

Fig. 1. Join Result Size Estimation for Generated Data Sets

Next, we conducted an experiment for uniform placement and area skewed data. Experimental data were generated according to the following steps. First, we selected a real data set from the Tiger data [10]. Then, since the real data are both placement skewed and area skewed, we redistributed the data set to generate several uniform placement and area skewed data sets. The selected data set is the road data of Sacramento county in the California state. The number of objects are 46516. The average x-length and y-length are 111 and 86. The original domain size and density are (75771,71218) and 0.24, respectively. We redistributed the objects to a new domain. The new domain size and density are (90000,70000) and 0.21, respectively.

Figure 1(b) shows estimation results and the real execution result using the above data sets for chain, star and clique queries. Est_All generally estimates the query results well. However, the estimation results of Est_Avg are nearly unacceptable especially in the high-way join. Estimation errors especially in the star query are tremendous. This shows that, to estimate well the result size of the multi-way spatial join among area skewed data sets, we need all statistics presented in Lemma 2 and Lemma 4.

4.2 Real Data Sets

Finally, we conducted experiments using real data sets which are both placement skewed and area skewed. The real data sets in our experiments were extracted from the TIGER/Line data of US Bureau of the Census [10]. We used the road segment data of 7 counties of the California State in the TIGER data. The

Table 2. Basic Statistical Information of the California TIGER Data

county	# of obj	domain area	avg length	density
Ala.	49070	86222,44995	102,80	0.23
Ker.	113407	257781,100758	212,169	0.26
Mon.	35417	175744,112068	234,192	0.20
Ora.	91970	69999,55588	80,66	0.21
Sac.	46516	75771,71218	111,86	0.24
S.D.	103420	151241,96476	122,104	0.22
S.B.	64037	99301,58696	100,81	0.22

statistical information of the California TIGER data is summarized in Table 2. We randomly extracted a data combination which consists of 6 counties from the TIGER data shown in Table 2. A multi-way join for the data combination was performed for the first M counties of the data combination for $M = 3, \ldots, 6$.

Figure 2 shows estimation results and error rates of the several multi-way spatial join types against the real execution result for the above data sets. Since the placement distribution of our data is highly skewed, we used the equi-area histogram [1] for all estimations with the varying number of buckets. We define the error rate as the following formula:

$$\text{error rate} = \frac{|\text{estimation result} - \text{real result}|}{\min(\text{estimation result}, \text{real result})} * 100 \tag{10}$$

In Figure 2, All_X*Y stands for the estimated number of solutions using all statistics shown in Lemma 2 and Lemma 4 in X*Y histogram buckets, and Avg_X*Y stands for the estimated number of solutions using only the number of objects and the average x-length and y-length (i.e., N_{r_i}, $\bar{s}_{r_i,x}$ and $\bar{s}_{r_i,y}$) in X*Y histogram buckets.

In the small number of buckets, All_50*25 gives reasonable error rates although some times it gives high error rates above 100. On the other hand, Avg_50*25 gives very high error rates so that many cases of the error rates are not seen in Figure 2. Therefore, Avg_50*25 is unacceptable. In the large number of buckets, Avg_200*100 also gives much smaller error rates than Avg_50*25. None the less, Avg_200*100 shows worse estimation quality than All_200*100. In comparison among Avg_X*Y, the estimation quality increases as the number of buckets increases. However, in All_X*Y, this is not always true.

The estimation for the star query gives higher error rates than that for the chain query. We believe this is because the estimation for the star query needs more statistics than that for the chain query. On the other hand, the estimation results for the clique query are better than those for the tree query because the estimation of the clique query result needs less statistics than that for the tree query result. Actually estimating the number of solutions in the multi-way join is known to be very difficult because of error propagation [3]. None the less, our approach obtained reasonable estimation result for the multi-way spatial join.

M	3		4		5		6	
All_50*25	1.20E+6	4	1.19E+6	34	1.81E+7	48	353955	145
Avg_50*25	91510	1.3E+3	59259	2.6E+3	85846	1.4E+4	7975	1.7E+3
All_100*50	1.24E+6	0	1.43E+6	11	1.53E+7	24	124037	16
Avg_100*50	156619	696	84581	1.8E+3	199349	6.0E+3	12643	1.0E+3
All_200*100	1.31E+6	5	1.52E+6	4	2.77E+6	342	118641	22
Avg_200*100	773136	61	461334	244	593260	2.0E+3	60316	139
Real	1.25E+6	%	1.59E+6	%	1.23E+7	%	144369	%

(a) Chain Query Type

M	3		4		5		6	
All_50*25	265486	40	1.79E+7	10	8.42E+8	355	1.35E+8	287
Avg_50*25	40188	372	27610	7.1E+4	17771	1.0E+6	3400	1.5E+7
All_100*50	152205	25	5.81E+6	240	6.60E+7	181	4.61E+8	13
Avg_100*50	77499	145	4.14E+6	377	1.31E+7	1.3E+3	3.60E+6	1.4E+4
All_200*100	154415	23	2.04E+7	3	1.39E+8	33	4.62E+8	13
Avg_200*100	105010	81	1.98E+7	0	1.16E+8	60	3.29E+7	1.5E+3
Real	189804	%	1.97E+7	%	1.85E+8	%	5.24E+8	%

(b) Star Query Type

M	3		4		5		6	
All_50*25	40489	4	18022	3	4298	4	205	7
Avg_50*25	26226	48	9449	96	3423	21	57	238
All_100*50	38187	2	17668	5	3805	9	140	37
Avg_100*50	27270	43	10510	76	4025	3	87	120
All_200*100	37167	5	17054	8	3292	26	148	30
Avg_200*100	31020	26	12792	45	4660	12	247	29
Real	38931	%	18496	%	4147	%	192	%

(c) Clique Query Type

Fig. 2. Join Result Size Estimation for Real Data Sets

5 Conclusions

Recent research introduced formulae, i.e., Equation (5) and Equation (8), for estimating the result size of the two kinds of the multi-way spatial join, tree and clique, under the assumption of the uniform placement distribution. However, the formulae are applied only for special cases of data areas. Equation (5) is satisfied only in the case that the x-lengths and y-lengths of all data per relation are equal, and Equation (8) in the case that the x-length and y-length distributions are independent.

In this paper, we extended the formulae to the arbitrary area of spatial data (Equation (6) and Equation (9)). The resulting formulae consist of object terms. In order for a query optimizer to estimate the query result size, we transformed the formulae of the object terms into those of the statistics terms. Then we analyzed the characteristics of the formulae represented by the statistics terms

(Lemma 1 and Lemma 3). Most importantly we derived the number and sort of the statistics which are required for the estimation of the multi-way join result size (Lemma 2 and Lemma 4). According to the result of the analysis, the complexity and the number of statistics for the tree typed multi-way spatial join are much more than those for the 2-way join and the clique typed multi-way join. Experimental results showed while the estimation using the formulae of [6] gives very high error rates or is nearly unacceptable in some cases, our approach generally provides accurate estimation results.

Although we considered rectangles only in the two dimensional space, the result of this paper can be easily extendable to the n-dimensional space.

Acknowledgements

This work was supported by the Strategic Research Program (ITRI: Information and Telecommunication Research Institute) of Chung-Ang University in 2003.

References

1. S. Acharya, V. Poosala and S. Ramaswamy, "Selectivity Estimation in Spatial Databases," Proc. of ACM SIGMOD, 1999.
2. Y.-W. Huang, N. Jing and E. A. Rundensteiner, "A Cost Model for Estimating the Performance of Spatial Joins Using R-trees," Proc. of SSDBMS, 1997.
3. Y. E. Ioannidis and S. Christodoulakis, "On the Propagation of Errors in the Size of Join Results," Proc. of ACM SIGMOD, 268-277, 1991.
4. I. Kamel and C. Faloutsos, "On Packing R-tree," Proc. of CIKM, 490-499, 1993.
5. B. Pagel, H. Six, H. Toben and P. Widmayer, "Towards an Analysis of Range Query Performance in Spatial Data Structures," Proc. of ACM PODS, 214-221, 1993.
6. D. Papadias, N. Mamoulis and Y. Theodoridis, "Constraint-Based Processing of Multiway Spatial Joins," Algorithmica, Vol. 30, No. 2, 188-215, 2001.
7. H.-H. Park, G.-H. Cha and C.-W. Chung, "Multi-way Spatial Joins Using R-trees: Methodology and Performance Evaluation," Proc. of SSD, 229-250, Lecture Notes in Computer Science 1651, Springer-Verlag, 1999.
8. H.-H. Park, Early Separated Filter/Refinement Strategies and Multi-way Spatial Joins for Spatial Query Optimization, Ph.D Thesis, KAIST, 2001.
9. Y. Theodoridis, E. Stefanakis and T. Sellis, "Cost Models for Join Queries in Spatial Databases," Proc. of IEEE ICDE, 476-483, 1998.
10. U.S. Bureau of the Census, Washington, DC., "TIGER/Line Files, 1995, Technical Documentation."

The Integer Hull of a Convex Rational Polytope

Jean B. Lasserre

LAAS-CNRS
7 Avenue du Colonel Roche, 31077 Toulouse cedex 4, France,
lasserre@laas.fr
http://www.laas.fr/∼lasserre

Abstract. Given $A \in \mathbf{Z}^{m\times n}$ and $b \in \mathbf{Z}^m$, we consider the integer program $\max\{c'x | Ax = b; x \in \mathbf{N}^n\}$ and provide an *equivalent* and *explicit* linear program $\max\{\widehat{c}'q | \mathrm{M}q = r; q \geq 0\}$, where $\mathrm{M}, r, \widehat{c}$ are easily obtained from A, b, c with no calculation. We also provide an explicit algebraic characterization of the integer hull of the convex polytope $\mathrm{P} = \{x \in \mathbf{R}^n | Ax = b; x \geq 0\}$. All strong valid inequalities can be obtained from the generators of a convex cone whose definition is explicit in terms of M.

1 Introduction

Let $A \in \mathbf{Z}^{m\times n}, b \in \mathbf{Z}^m, c \in \mathbf{R}^n$ and consider the integer program

$$\mathbf{P} \rightarrow := \max\{\, c'x \quad | \quad Ax = b; \quad x \in \mathbf{N}^n\}, \tag{1}$$

where the convex polyhedron $\mathrm{P} := \{x \in \mathbf{R}^n \,|\, Ax = b;\, x \geq 0\}$ is compact. If P_1 denotes the *integer hull* of P, then solving **P** is equivalent to solving the linear program $\max\{c'x \,|\, x \in \mathrm{P}_1\}$.

However, finding the integer hull P_1 of P is a difficult problem. As mentioned by Wolsey in [8, p. 15]), and to the best of our knowledge, no *explicit* (or "simple") characterization (or description) of P_1 has been provided so far. In the general *cutting plane* methods originated by Gomory and Chvátal in the early sixties, and the *lift-and-project* methods described by e.g. Laurent in [4], one obtains P_1 as the final iterate of a *finite* nested sequence $\mathrm{P} \supseteq \mathrm{P}' \supseteq \mathrm{P}'' \cdots \supseteq \mathrm{P}_1$ of polyhedra. However, in all those procedures, P_1 has no explicit description in terms of the initial data A, b. On the other hand, for specific polytopes P, one is often able to provide some *strong valid inequalities* in explicit form, but very rarely all of them (as for the matching polytope of a graph). For more details the interested reader is referred to Cornuejols and Li in [1], Jeroslow in [2], Laurent in [4], Nemhauser and Wolsey in [6], Schrijver in [7, §23], Wolsey in [8, §8,9], and the many references therein.

Contribution. The main goal of this paper is to provide a *structural* result on the integer hull P_1 of a convex rational polytope P, in the sense that we obtain an *explicit algebraic characterization* of the defining hyperplanes of P_1, in terms of generators of a convex cone C which is itself described directly from the initial

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 866–875, 2003.
© Springer-Verlag Berlin Heidelberg 2003

data A, with *no* calculation. Hence *all* strong valid inequalities can be obtained from the generators of the cone C.

We first show that the integer program **P** is equivalent to a linear program in the *explicit* form

$$\max_{q \in \mathbf{R}^s} \{\, \widehat{c}\,' q \quad | \quad \mathrm{M}\, q = r; \quad q \geq 0\}. \tag{2}$$

By *explicit* we mean that the data $\mathrm{M}, r, \widehat{c}$ of the linear program (2) are constructed *explicitly* and *easily* from the initial data A, b, c. In particular, *no* calculation is needed and M, r have all their entries in $\{0, \pm 1\}$. In addition M is very sparse. Of course, and as expected, the dimension of the matrix M is in general exponential in the problem size. However, for the class of problems where A has nonnegative integral entries and b is bounded, then (2) is solvable in time polynomial in the problem size.

2 Notation and Preliminary Results

For a vector $b \in \mathbf{R}^m$ and a matrix $A \in \mathbf{R}^{m \times n}$, denote by b' and $A' \in \mathbf{R}^{n \times m}$ their respective transpose. Let $\mathbf{R}[x_1, \ldots, x_n]$ be the ring of real-valued polynomials in the variables $x_1, \ldots, x_n$. A polynomial $f \in \mathbf{R}[x_1, \ldots, x_n]$ is written

$$x \;\mapsto\; f(x) \;=\; \sum_{\alpha \in \mathbf{N}^n} f_\alpha x^\alpha \;=\; \sum_{\alpha \in \mathbf{N}^n} f_\alpha x_1^{\alpha_1} \cdots x_n^{\alpha_n},$$

for *finitely many* real coefficients $\{f_\alpha\}$, in the (usual) basis of monomials.

Given a matrix $A \in \mathbf{Z}^{m \times n}$, let $A_j \in \mathbf{Z}^m$ denote its j-th column (equivalently, the j-th row of A'); then z^{A_j} stands for

$$z^{A_j} \;:=\; z_1^{A_{1j}} \cdots z_m^{A_{mj}} \;=\; \mathrm{e}^{\langle A_j, \ln z \rangle} \;=\; \mathrm{e}^{(A' \ln z)_j},$$

and if $A_j \in \mathbf{N}^m$ then z^{A_j} is a monomial of $\mathbf{R}[z_1, \ldots, z_m]$. We first recall the following result :

Theorem 1 (A discrete Farkas lemma [3]). *Let $A \in \mathbf{N}^{m \times n}, b \in \mathbf{N}^m$. Then the following two statements (i) and (ii) are equivalent :*

(i) *The linear system $Ax = b$ has a solution $x \in \mathbf{N}^n$.*

(ii) *The real-valued polynomial $z \;\mapsto\; z^b - 1 := z_1^{b_1} \cdots z_m^{b_m} - 1$ can be written*

$$z^b - 1 \;=\; \sum_{j=1}^{n} Q_j(z)(z^{A_j} - 1) \tag{3}$$

for some real-valued polynomials $Q_j \in \mathbf{R}[z_1, \ldots, z_m]$, $j = 1, \ldots, n$, all of them with nonnegative coefficients.

In addition, the degree of the Q_j's in (3) is bounded by

$$b^* \;:=\; \sum_{j=1}^{m} b_j - \min_k \sum_{j=1}^{m} A_{jk}. \tag{4}$$

Discussion. (a) With b^* as in (4) denote by $s := s(b^*) := \binom{m+b^*}{b^*}$ the dimension of the vector space of polynomials of degree b^* in m variables. In view of Theorem 1, and given $b \in \mathbf{N}^m$, checking the existence of a solution $x \in \mathbf{N}^n$ to $Ax = b$ reduces to checking whether or not there exists a nonnegative solution q to a system of linear equations

$$\mathrm{M}\, q \;=\; r; \quad q \geq 0 \tag{5}$$

for some matrix $\mathrm{M} \in \mathbf{Z}^{p \times ns}$, and vector $r \in \mathbf{Z}^p$, with :

- $n\,s$ variables $\{q_{j\alpha}\}$, the nonnegative coefficients of the Q_j's.
- p equations to identify the terms of same power in both sides of (3); obviously one has $p \leq s(b^* + a) := \binom{m+b^*+a}{b^*+a}$ (with $a := \max_k \sum_{j=1}^m A_{jk}$).

This in turn reduces to solving a linear programming (LP) problem. Observe that in view of (3), the matrix of constraints $\mathrm{M} \in \mathbf{Z}^{p \times ns}$ which has only 0 and ± 1 coefficients, is *easily* deduced from A with *no* calculation (and is very sparse). The same is true for $r \in \mathbf{Z}^p$ which has only two non zero entries (equal to ± 1).

(b) In fact, from the proof of Theorem 1, it follows that one may even enforce the weights Q_j in (3) to be polynomials in $\mathbf{Z}[z_1, \ldots, z_m]$ (instead of $\mathbf{R}[z_1, \ldots, z_m]$) with nonnegative coefficients (and even with coefficients in $\{0, 1\}$) However, (a) above shows that the strength of Theorem 1 is precisely to allow $Q_j \in \mathbf{R}[z_1, \ldots, z_m]$ as it permits to check feasibility by solving a (continuous) linear program. Enforcing $Q_j \in \mathbf{Z}[z_1, \ldots, z_m]$ would result in an *integer* program of size larger than that of the original problem.

Next, with $A \in \mathbf{N}^{m \times n}, b \in \mathbf{N}^m$ let $\mathrm{P} \subset \mathbf{R}^n$ be the convex polyhedron

$$\mathrm{P} \;:=\; \{x \in \mathbf{R}^n \quad | \quad Ax = b; \quad x \geq 0\}. \tag{6}$$

Similarly, with M, r as in (5), let

$$\Omega \;:=\; \{q \in \mathbf{R}^{ns} \quad | \quad \mathrm{M}\, q \;=\; r; \quad q \geq 0\}, \tag{7}$$

be the convex polyhedron of feasible solutions $q \in \mathbf{R}^{ns}$ of (5).

Define the row vector $e_s := (1, \ldots, 1) \in \mathbf{R}^s$ and let $\mathrm{E} \in \mathbf{N}^{n \times ns}$ be the block diagonal matrix, whose each diagonal block is the row vector e_s, that is,

$$\mathrm{E} \;:=\; \begin{bmatrix} e_s & 0 & \ldots & 0 \\ 0 & e_s & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & e_s \end{bmatrix}. \tag{8}$$

Proposition 1. *Let $A \in \mathbf{N}^{m \times n}, b \in \mathbf{N}^m$ be given and* M *be as in (2). Let* P, Ω *be the convex polyhedra defined in (6)-(7).*

(a) *Let $q \in \Omega$. Then $x := \mathrm{E}q \in \mathrm{P}$. In particular, if $q \in \Omega \cap \mathbf{Z}^{ns}$ then $x \in \mathrm{P} \cap \mathbf{Z}^n$.*

(b) *Let $x \in \mathrm{P} \cap \mathbf{Z}^n$. Then $x = \mathrm{E}q$ for some $q \in \Omega \cap \mathbf{Z}^{ns}$.*

Proof. (a) With $q \in \Omega$, let $\{Q_j\}_{j=1}^n \subset \mathbf{R}[z_1, \ldots, z_m]$ be the set of polynomials (with vector of nonnegative coefficients q) which satisfy (3). Taking the derivative of both sides of (3) with respect to z_k, at the point $z = (1, \ldots, 1)$, yields

$$b_k \;=\; \sum_{j=1}^{n} Q_j(1, \ldots, 1) A_{kj} \;=\; \sum_{j=1}^{n} A_{kj} x_j \quad j = 1, \ldots, n,$$

with $x_j := Q_j(1, \ldots, 1)$ for all $j = 1, \ldots, n$. Next, use the facts that (a) all the Q_j's have nonnegative coefficients $\{q_{j\alpha}\}$, and (b), $Q_j(1, \ldots, 1) = \sum_{\alpha \in \mathbf{N}^m} q_{j\alpha} = (\mathrm{E}q)_j$ for all $j = 1, \ldots, n$, to obtain $x := \mathrm{E}q \in \mathrm{P}$. Moreover, if $q \in \Omega \cap \mathbf{Z}^{ns}$ then obviously $x \in \mathrm{P} \cap \mathbf{Z}^n$.

(b) Let $x \in \mathrm{P} \cap \mathbf{Z}^n$ so that $x \in \mathbf{N}^n$ and $Ax = b$; write

$$z^b - 1 \;=\; z^{A_1 x_1} - 1 + z^{A_1 x_1}(z^{A_2 x_2} - 1) + \cdots + z^{\sum_{j=1}^{n-1} A_j x_j}(z^{A_n x_n} - 1),$$

and

$$z^{A_j x_j} - 1 \;=\; (z^{A_j} - 1)\left[1 + z^{A_j} + \cdots + z^{A_j(x_j - 1)}\right] \quad j = 1, \ldots, n,$$

to obtain (3) with

$$z \;\mapsto\; Q_j(z) \;:=\; z^{\sum_{k=1}^{j-1} A_k x_k}\left[1 + z^{A_j} + \cdots + z^{A_j(x_j - 1)}\right], \quad j = 1, \ldots, n.$$

We immediately see that each Q_j has all its coefficients $\{q_{j\alpha}\}$ nonnegative (and even in $\{0, 1\}$). Moreover, $Q_j(1, \ldots, 1) = x_j$ for all $j = 1, \ldots, n$, or equivalently, $x = \mathrm{E}q$ with $q \in \Omega \cap \mathbf{Z}^{ns}$. □

3 Main Result

We first prove our results in the case $A \in \mathbf{N}^{m \times n}$; then we show that the general case $A \in \mathbf{Z}^{m \times n}$ reduces to the former by adding one variable and one contraint to the original problem. So let $A \in \mathbf{N}^{m \times n}, b \in \mathbf{N}^m$, and with no loss of generality, we may and will assume that every column of A has at least one non zero entry, in which case P in (6) is a polytope.

Recall that with every $q \in \Omega$ we may associate a set of polynomials $\{Q_j\} \subset \mathbf{R}[z_1, \ldots, z_m]$ that satisfy (3). Conversely, to such a set of polynomials $\{Q_j\}$, its associated vector of coefficients $q = \{q_{j\alpha}\}$ is in Ω.

Theorem 2. *Let $A \in \mathbf{N}^{m \times n}, b \in \mathbf{N}^m$ be given and let $\mathrm{M} \in \mathbf{Z}^{p \times ns}, r \in \mathbf{Z}^p$ and $\mathrm{E} \in \mathbf{N}^{n \times ns}$, be as in (5) and (8), respectively.*

Then for every vertex $\widehat{q} \in \mathbf{R}^{ns}$ of the convex polyhedron Ω defined in (7)

$$x \;:=\; \mathrm{E}\,\widehat{q} \in \mathbf{N}^n \quad \textit{and} \quad Ax \;=\; b, \tag{9}$$

that is, $x \in \mathrm{P} \cap \mathbf{Z}^n$.

The proof is postponed to §4.

3.1 An Equivalent Linear Program

Consider the integer program $\mathbf{P}$. For every $c \in \mathbf{R}^n$ let $\widehat{c} \in \mathbf{R}^{ns}$ be defined as

$$\widehat{c}' \;=\; (\widehat{c_1}', \ldots, \widehat{c_n}') \quad \text{with} \quad \widehat{c}'_j \;=\; c_j(1, \ldots, 1) \in \mathbf{R}^s \quad \forall j = 1, \ldots, n \tag{10}$$

Equivalently, $\widehat{c}' = c'\mathrm{E}$ with E as in (8). It also follows that $\widehat{c}'q = c'x$ whenever $x = \mathrm{E}q$. As a consequence of Theorem 1 we obtain immediately

Corollary 1. *Let $A \in \mathbf{N}^{m\times n}, b \in \mathbf{N}^m, c \in \mathbf{R}^n$ be given. Let $\mathrm{M} \in \mathbf{Z}^{p\times ns}, r \in \mathbf{Z}^p$ and $\mathrm{E} \in \mathbf{N}^{n\times ns}$, be as in (5) and (8), respectively.*

(a) *The integer program*

$$\mathbf{P} \;\rightarrow\quad \max_x \{\, c'x \quad | \quad Ax = b; \quad x \in \mathbf{N}^n\} \tag{11}$$

has same optimal value as the linear program

$$\mathbf{Q} \;\rightarrow\quad \max_{q\in\mathbf{R}^{ns}} \{\, \widehat{c}'\, q \quad | \quad \mathrm{M}\, q \;=\; r; \quad q \geq 0\} \tag{12}$$

(including the case $-\infty$).

(b) *In addition, let $q^* \in \mathbf{R}^{ns}$ be a vertex of Ω in (7), optimal solution of the linear program $\mathbf{Q}$. Then $x^* := \mathrm{E}q^* \in \mathbf{N}^n$ and x^* is an optimal solution of the integer program $\mathbf{P}$.*

Proof. Let $\max \mathbf{P}$ and $\max \mathbf{Q}$ denote the respective optimal values of $\mathbf{P}$ and $\mathbf{Q}$. We first treat the case $-\infty$. $\max \mathbf{P} = -\infty$ only if $\mathrm{P} \cap \mathbf{Z}^n = \emptyset$. But then $\Omega = \emptyset$ as well, which in turn implies $\max \mathbf{Q} = -\infty$. Indeed, by Theorem 1, if $\mathrm{P} \cap \mathbf{Z}^n = \emptyset$, i.e., if $Ax = b$ has no solution $x \in \mathbf{N}^n$, then one cannot find polynomials $\{\mathbf{Q}_j\} \subset \mathbf{R}[z_1, \ldots, z_m]$ with nonnegative coefficients, that satisfy (3). Therefore, from the definition of Ω, if $\Omega \neq \emptyset$ one would have a contradiction.

Conversely, if $\Omega = \emptyset$ (so that $\max \mathbf{Q} = -\infty$) then by definition of Ω, one cannot find polynomials $\{\mathbf{Q}_j\} \subset \mathbf{R}[z_1, \ldots, z_m]$ with nonnegative coefficients, that satisfy (3). Therefore, by Theorem 1, $Ax = b$ has no solution $x \in \mathbf{N}^n$ which in turn implies $\max \mathbf{P} = -\infty$, i.e., $\mathrm{P} \cap \mathbf{Z}^n = \emptyset$.

In the case when $\max \mathbf{P} \neq -\infty$, we necessarily have $\max \mathbf{P} < \infty$ because the convex polyhedron P is compact. Next, consider a feasible solution $q \in \Omega$ of $\mathbf{Q}$. From Proposition 1(a) $x := \mathrm{E}q \in \mathrm{P}$. Therefore, as x is bounded then so is $\mathrm{E}q$, which, in view of the definition (8) of E, also implies that q is bounded. Hence Ω is compact which in turn implies that the optimal value of $\mathbf{Q}$ is finite and attained at some vertex q^* of Ω.

Now, let $x^* \in \mathbf{N}^n$ be an optimal solution of $\mathbf{P}$. By Proposition 1(b) there exists some $q \in \Omega$ with $\mathrm{E}q = x^*$. From the definition (10) of the vector $\widehat{c}$ we have

$$\widehat{c}'q \;=\; c'\mathrm{E}q \;=\; c'x^*,$$

which implies $\max \mathbf{Q} \geq \max \mathbf{P}$.

On the other hand, let $q^* \in \Omega$ be a vertex of Ω, optimal solution of $\mathbf{Q}$. By Theorem 2, $x := \mathrm{E}q^* \in \mathrm{P} \cap \mathbf{Z}^n$, that is, $x \in \mathbf{N}^n$ is a feasible solution of $\mathbf{P}$. Again, from the definition (10) of the vector $\widehat{c}$, we have $c'x = c'\mathrm{E}q^* = \widehat{c}'q^*$, which, in view of $\max \mathbf{P} \leq \max \mathbf{Q}$, implies $\max \mathbf{P} = \max \mathbf{Q}$, and $x \in \mathbf{N}^n$ is an optimal solution of $\mathbf{P}$. This completes the proof of (a) and (b). □

3.2 The Integer Hull

We now describe the integer hull P_1 of P, i.e., the convex hull of $\mathrm{P} \cap \mathbf{Z}^n$.

Theorem 3. *Let $A \in \mathbf{N}^{m\times n}, b \in \mathbf{N}^m$, and let $\mathrm{E} \in \mathbf{N}^{n\times ns}, \mathrm{M} \in \mathbf{Z}^{p\times ns}, r \in \mathbf{Z}^p$ be as in (8) and (5), respectively. Let $\{(u^k, v^k)\}_{k=1}^t \subset \mathbf{R}^{n\times p}$ be the (finite) set of generators of the convex cone $C \subset \mathbf{R}^{n\times p}$ defined by*

$$C := \{(u, v) \in \mathbf{R}^{n\times p} \,|\, \mathrm{E}'u + \mathrm{M}'v \geq 0\}. \tag{13}$$

The integer hull P_1 of P is the convex polyhedron defined by

$$\langle u^k, x\rangle + \langle v^k, r\rangle \geq 0 \qquad \forall k = 1, \ldots, t, \tag{14}$$

or, equivalently,

$$\mathrm{P}_1 := \{x \in \mathbf{R}^n \quad | \quad U\,x \geq u\}, \tag{15}$$

where the matrix $U \in \mathbf{R}^{t\times n}$ has row vectors $\{u^k\}$, and the vector $u \in \mathbf{R}^t$ has coordinates $u_k = \langle -v^k, r\rangle$, $k = 1, \ldots, t$.

Proof. Given $x \in \mathbf{R}^n$, consider the following linear system :

$$\mathrm{E}\,q = x; \quad \mathrm{M}\,q = r; \quad q \geq 0, \tag{16}$$

where M, E are defined in (5) and (8) respectively. Invoking the celebrated Farkas lemma, the system (16) has a solution $q \in \mathbf{R}^{ns}$ if and only if (14) holds.

Therefore, let $x \in \mathbf{R}^n$ satisfy $Ux \geq u$ with U, u as in (15). By Farkas lemma, the system (16) has a solution $q \in \Omega$. As Ω is compact, q is a convex combination $\sum_k \gamma_k \widehat{q}^k$ of the vertices $\{\widehat{q}^k\}$ of Ω. By Theorem 2, for each vertex $\widehat{q}^k$ of Ω we have $\widehat{x}^k := \mathrm{E}\widehat{q}^k \in \mathrm{P} \cap \mathbf{Z}^n$. Therefore,

$$x = \mathrm{E}q = \sum_k \gamma_k \mathrm{E}\widehat{q}^k = \sum_k \gamma_k \widehat{x}^k, \tag{17}$$

that is, x is a convex combination of points $\widehat{x}^k \in \mathrm{P} \cap \mathbf{Z}^n$, i.e., $x \in \mathrm{P}_1$;

Conversely, let $x \in \mathrm{P}_1$, i.e., $x \in \mathbf{R}^n$ is a convex combination $\sum_k \gamma_k \widehat{x}^k$ of points $\widehat{x}^k \in \mathrm{P} \cap \mathbf{Z}^n$. By Proposition 1(b), for each k, $\widehat{x}^k = \mathrm{E}q^k$ for some vector $q^k \in \Omega \cap \mathbf{Z}^{ns}$. Therefore, as each $(\widehat{x}^k, q^k)$ satisfies (16), then so does their convex combination $(x, q) := \sum_k \gamma_k(\widehat{x}^k, q^k)$. By Farkas lemma again, we must have $Ux \geq u$, and so, $\mathrm{P}_1 \subseteq \{x \in \mathbf{R}^n \,|\, Ux \geq u\}$, which completes the proof. □

Observe that the convex cone C in (13) of Theorem 3, is defined *explicitly* in terms of the initial data A, and with *no* calculation. Indeed, the matrix M in (5) is easily obtained from A and E is explictly given in (8). Thus, the interest of Theorem 3 is that we obtain an algebraic characterization (15) of P_1 via the generators of a convex cone C simply related to A.

3.3 The General Case $A \in \mathbf{Z}^{m\times n}$

We now consider the case $A \in \mathbf{Z}^{m\times n}$, that is, A may have negative entries. We will assume that the convex polyhedron $\mathrm{P} \subset \mathbf{R}^n$, defined in (6) is compact.

Let $\alpha \in \mathbf{N}^n, \beta \in \mathbf{N}$ be such that for all $j = 1, \ldots, m$,

$$\widehat{b}_j := b_j + \beta \geq 0\,; \quad \widehat{A}_{jk} := A_{jk} + \alpha_k \geq 0; \quad k = 1, \ldots, n. \tag{18}$$

As P is compact we have

$$\max_{x\in\mathbf{N}^n} \{\sum_{j=1}^n \alpha_j x_j \,|\, Ax = b\} \leq \max_{x\in\mathbf{R}^n; x\geq 0} \{\sum_{j=1}^n \alpha_j x_j \,|\, Ax = b\} =: \rho^*(\alpha) < \infty.$$

Given $\alpha \in \mathbf{N}^n$ as in (18), the scalar $\rho^*(\alpha)$ is easily calculated by solving a LP problem. Choose $\rho^*(\alpha) \leq \beta \in \mathbf{N}$, and let $\widehat{A} \in \mathbf{N}^{m\times n}, \widehat{b} \in \mathbf{N}^m$ be as in (18).

The feasible solutions $x \in \mathbf{N}^n$ of $Ax = b$, i.e., the points of $\mathrm{P} \cap \mathbf{Z}^n$, are in one-to-one correspondence with the solutions $(x, u) \in \widehat{\mathrm{P}} \cap \mathbf{Z}^{n+1}$ where $\widehat{\mathrm{P}} \subset \mathbf{R}^{n+1}$ is the convex polytope

$$\widehat{\mathrm{P}} := \left\{(x, u) \in \mathbf{R}^n \times \mathbf{R} \,|\, \left\{ \begin{array}{c} \widehat{A}x + e_m u = \widehat{b} \\ \alpha' x + u = \beta \end{array} \right. ; \quad x, u \geq 0 \right\}; \tag{19}$$

Indeed, if $x \in \mathrm{P} \cap \mathbf{Z}^n$, i.e., $Ax = b$ with $x \in \mathbf{N}^n$, then

$$Ax + e_m \sum_{j=1}^n \alpha_j x_j - e_m \sum_{j=1}^n \alpha_j x_j = b + (\beta - \beta) e_m,$$

or equivalently, $\widehat{A}x + (\beta - \sum_{j=1}^n \alpha_j x_j) e_m = \widehat{b}$, and thus, as $\beta \geq \rho^*(\alpha) \geq \alpha' x$, letting $u := \beta - \alpha' x \in \mathbf{N}$, yields $(x, u) \in \widehat{\mathrm{P}} \cap \mathbf{Z}^{n+1}$. Conversely, let $(x, u) \in \widehat{\mathrm{P}} \cap \mathbf{Z}^{n+1}$. Using the definitions of $\widehat{A}$ and $\widehat{b}$, it then follows immediately that

$$Ax + e_m \sum_{j=1}^n \alpha_j x_j + u e_m = b + \beta e_m; \quad \sum_{j=1}^n \alpha_j x_j + u = \beta,$$

so that $Ax = b$ with $x \in \mathbf{N}^n$, i.e., $x \in \mathrm{P} \cap \mathbf{Z}^n$. In other words,

$$x \in \mathrm{P} \cap \mathbf{Z}^n \quad \Leftrightarrow \quad (x, \beta - \alpha' x) \in \widehat{\mathrm{P}} \cap \mathbf{Z}^{n+1}. \tag{20}$$

The convex polytope $\widehat{\mathrm{P}}$ can be written

$$\widehat{\mathrm{P}} := \{(x, u) \in \mathbf{R}^{n+1} \,|\, B \begin{bmatrix} x \\ u \end{bmatrix} = (\widehat{b}, \beta); \quad x, u \geq 0\}, \tag{21}$$

with

$$B := \begin{bmatrix} \widehat{A} & | & e_m \\ - & & - \\ \alpha' & | & 1 \end{bmatrix}.$$

As $B \in \mathbf{N}^{(m+1)\times(n+1)}$, we are back to the case analyzed in Theorem 2.

In particular, the integer program $\mathbf{P} \to \max\{c'x \mid Ax = b; x \in \mathbf{N}^n\}$ is equivalent to the integer program

$$\widehat{\mathbf{P}} \quad \to \max\left\{ c'x \quad | \quad B\begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} \widehat{b} \\ \beta \end{bmatrix}; \quad (x,u) \in \mathbf{N}^n \times \mathbf{N} \right\}. \tag{22}$$

Hence, Theorem 1, Theorem 2, Corollary 1 and Theorem 3 are still valid with $B \in \mathbf{N}^{(m+1)\times(n+1)}$ in lieu of $A \in \mathbf{N}^{m\times n}$, $(\widehat{b}, \beta) \in \mathbf{N}^m \times \mathbf{N}$ in lieu of $b \in \mathbf{N}^m$, and $\widehat{\mathrm{P}} \subset \mathbf{R}^{n+1}$ in lieu of $\mathrm{P} \subset \mathbf{R}^n$.

So again, as in previous sections, the polytope $\widehat{\Omega}$ associated with $\widehat{\mathrm{P}}$ is explicitly defined from the initial data A, because $\widehat{A}$ is simply defined from A and α. In turn, as the convex cone C in Theorem 3 is also defined explicitly from A via M, again one obtains a simple characterization of the integer hull $\widehat{\mathrm{P}}_1$ of $\widehat{\mathrm{P}}$ via the generators of C.

If we are now back to the initial data A, b then P_1 is easily obtained from $\widehat{\mathrm{P}}_1$. Indeed, by Theorem 3, let

$$\widehat{\mathrm{P}}_1 \;=\; \{(x,u) \in \mathbf{R}^{n+1} \mid \langle w^k, x\rangle + \delta^k u \geq \rho^k; \quad k = 1, \ldots, t\},$$

for some $\{(w^k, \delta^k) \in \mathbf{R}^n \times \mathbf{R}\}_{k=1}^t$, and some $t \in \mathbf{N}$. Then from (20) it immediately follows that

$$\mathrm{P}_1 \;=\; \{x \in \mathbf{R}^n \mid \langle w^k - \delta^k\alpha, x\rangle \geq \rho^k - \beta\delta^k; \quad k = 1, \ldots, t\}.$$

4 Proof of Theorem 2

Proof. Let $\widehat{q} \in \mathbf{R}^{ns}$ be a vertex of Ω in (7) and let $\{\widehat{Q}_j\} \subset \mathbf{R}[z_1, \ldots, z_m]$ be the set of polynomials with vector of coefficients $\widehat{q} = \{\widehat{q}_{j\alpha}\}$. By definition, the Q_j's have nonnegative coefficients and satisfy (3). Let $x := \mathrm{E}\widehat{q}$. Then $Ax = b$ and $x \geq 0$ follow from Proposition 1(a).

It thus remains to prove that $\widehat{q} = \{\widehat{q}_{j\alpha}\} \in \mathbf{N}^{ns}$ whenever $\widehat{q}$ is a vertex of Ω, in which case $x = \mathrm{E}\widehat{q} \in \mathbf{N}^n$, from which (9) follows.

We proceed by contradiction and show that if $\widehat{q}_{j\alpha} \notin \mathbf{N}$ for some j and $\alpha \in \mathbf{N}^m$, then there exists $0 < \gamma < 1$ such that for all $j = 1, \ldots, n$, we may write each polynomial $\widehat{Q}_j$ as $\gamma Q_j^1 + (1-\gamma)Q_j^2$, where both sets of polynomials $\{Q_j^1\}$ and $\{Q_j^2\}$ are distinct and satisfy (3). Therefore, $\widehat{q} = \gamma q_1 + (1-\gamma)q_2$ for two distinct $0 \leq q_1, q_2 \in \Omega$, in contradiction with $\widehat{q}$ being a vertex of Ω. The technique of the proof uses ideas from Mayr and Meyer in [5].

Thus, assume that $\widehat{q}_{j\alpha} \notin \mathbf{N}$ for some $j \in \{1, \ldots, n\}, \alpha \in \mathbf{N}^m$. As $\widehat{q}$ is a vertex of the rational convex polyhedron Ω, all the entries of $\widehat{q}$ are in Q. Therefore, let $1 < \delta \in \mathbf{N}$ be an integer such that $\delta\widehat{q}_{j\alpha} \in \mathbf{N}$ for all $j = 1, \ldots, n$, and all $\alpha \in \mathbf{N}^m$, and write

$$w_{j\alpha} \;=\; \delta\widehat{q}_{j\alpha} \in \mathbf{N} \qquad j = 1, \ldots, n; \quad \alpha \in \mathbf{N}^m.$$

As the polynomials $\widehat{Q}_j$'s satisfy (3), we have

$$\delta(z^b-1) = \sum_{j=1}^n \delta Q_j(z)(z^{A_j}-1)$$
$$= \sum_{j=1}^n \sum_{\alpha\in\mathbf{N}^m} w_{j\alpha} z^\alpha (z^{A_j}-1) = \sum_{j=1}^n \sum_{\alpha\in\mathbf{N}^m} \sum_1^{w_{j\alpha}} z^\alpha (z^{A_j}-1), \quad (23)$$

that we can rewrite

$$(\delta-1)z^b - \delta + z^b = \sum_{k=1}^t g_k(z^{A_{j_k}}-1) \quad (24)$$

for some $t\in\mathbf{N}$, and where
- $j_k \in \{1,\ldots,n\}$ for all $k=1,\ldots,t$, and
- g_k is a monomial $z^{\beta(k)}$ with coefficient 1.

Next, set $I_0 := \{1,2,\ldots,t\}$ and consider the monomial z^b on the left-hand-side of the identity (24). In view of the form of the right-hand-side of (24) we must have $z^b = g_{k_1} z^{A_{j_{k_1}}}$ for some index $k_1 \in I_0$. Therefore,

$$(\delta-1)z^b - \delta + g_{k_1} = \sum_{k\in I_0-\{k_1\}} g_k(z^{A_{j_k}}-1) = \sum_{k\in I_1} g_k(z^{A_{j_k}}-1), \quad (25)$$

(with $I_1 := I_0 \setminus \{k_1\}$). If $g_{k_1}\equiv 1$ then we stop; otherwise we repeat the same above step with the monomial g_{k_1} in lieu of z^b, to obtain

$$(\delta-1)z^b - \delta + g_{k_2} = \sum_{k\in I_1\setminus\{k_2\}} g_k(z^{A_{j_k}}-1) = \sum_{k\in I_2} g_k(z^{A_{j_k}}-1),$$

for some $k_2\in I_1$ etc, until we finally obtain $g_{k_i}\equiv 1$ for some $i\le t$, $k_i\in I_{i-1}$, and

$$(\delta-1)z^b - \delta + g_{k_i} = \sum_{k\in I_{i-1}\setminus\{k_i\}} g_k(z^{A_{j_k}}-1) = \sum_{k\in I_i} g_k(z^{A_{j_k}}-1),$$

with $I_i\neq\emptyset$. Indeed if we do not find $g_{k_i}\equiv 1$, and as $|I_0|=t$, we end up with $I_t=\emptyset$ and

$$(\delta-1)z^b + g_{k_t} - \delta \equiv 0,$$

which is clearly impossible. Therefore, we must terminate with $g_{k_i}\equiv 1$ and in addition, with $i<t$ because otherwise, if $i=t$, we would obtain $(\delta-1)(z^b-1)\equiv 0$, in contradiction with $\delta>1$. Thus, we have $i<t$ and $g_{k_i}\equiv 1$, so that

$$z^b - g_{k_1} = g_{k_1}(z^{A_{j_{k_1}}}-1);\ g_{k_1}-g_{k_2} = g_{k_2}(z^{A_{j_{k_2}}}-1);\ldots;\ g_{k_{i-1}}-1 = (z^{A_{j_{k_i}}}-1).$$

Summing up yields

$$z^b - 1 = \sum_{l=1}^i g_{k_l}(z^{A_{j_{k_l}}}-1) = \sum_{j=1}^n Q_j^1(z)(z^{A_j}-1), \quad (26)$$

for some polynomials $Q_j^1 \in \mathbf{Z}[z_1, \ldots, z_m]$. By construction, for all $j = 1, \ldots, n$, the polynomial Q_j^1 has all its coefficients in $\{0, 1\}$. On the other hand,

$$(\delta - 1)(z^b - 1) \;=\; \sum_{k \in I_i} g_k(z^{A_{j_k}} - 1) \;=\; \sum_{j=1}^{n} Q_j^2(z)(z^{A_j} - 1), \tag{27}$$

for some polynomials $Q_j^2 \in \mathbf{Z}[z_1, \ldots, z_m]$ with nonnegative coefficients. If we sum up (26) and (27), we obtain

$$\begin{aligned}\delta(z^b - 1) \;=\; \sum_{k \in I_0} g_k(z^{A_{j_k}} - 1) &= \sum_{k \in I_i} g_k(z^{A_{j_k}} - 1) + \sum_{k \in I_0 \setminus I_i} g_k(z^{A_{j_k}} - 1) \\ &= \sum_{j=1}^{n} \left[Q_j^1(z) + Q_j^2(z)\right]\, (z^{A_j} - 1).\end{aligned}$$

Recalling the definition of the g_k in (23), then for all $j = 1, \ldots, n$ we must have

$$\widehat{Q}_j(z) \;\equiv\; \frac{1}{\delta}\left[Q_j^1(z) + Q_j^2(z)\right] \;=\; \frac{1}{\delta} Q_j^1(z) + \frac{(\delta - 1)}{\delta} \frac{Q_j^2(z)}{(\delta - 1)}.$$

In other words, $\widehat{Q}_j$ is a convex combination of the polynomials Q_j^1 and $Q_j^2/(\delta - 1)$, with respective (nontrivial) weights $1/\delta$ and $(\delta - 1)/\delta$. As both Q_j^1 and Q_j^2 are in $\mathbf{Z}[z_1, \ldots, z_m]$ for all $j = 1, \ldots, n$, the polynomials Q_j^1 and $Q_j^2/(\delta - 1)$ cannot be all identical, otherwise we would have $\widehat{Q}_j \equiv Q_j^1 \in \mathbf{Z}[z_1, \ldots, z_n]$ for all $j = 1, \ldots, n$, in contradiction with $\widehat{q}_{j\alpha} \notin \mathbf{N}$ for some j and $\alpha \in \mathbf{N}^m$. In addition, from (26)-(27), both sets of polynomials $\{Q_j^1\}$ and $\{Q_j^2/(\delta - 1)\}$ are solutions of (3). Therefore, let $0 \leq q_1, q_2 \in \Omega$ be the respective (distinct) vectors of coefficients of the polynomials $\{Q_j^1\}$ and $\{Q_j^2/(\delta - 1)\}$, and let $0 < \gamma := 1/\delta < 1$. We have $\widehat{q} = \gamma q_1 + (1 - \gamma) q_2$, in contradiction with $\widehat{q}$ being a vertex of Ω. □

References

1. Cornuejols, G., Li, Y.: Elementary closures for integer programs. Oper. Res. Letters **28** (2001) 1–8.
2. Jeroslow, R.J.: An introduction to the theory of cutting-planes. Ann. Discrete Math. **5** (1979) 71–95
3. Lasserre, J.B.: A discrete Farkas lemma, Technical report #00145, LAAS-CNRS, Toulouse, France, 2002.
4. Laurent, M.: A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. Technical Report # PNA R0-108, CWI, Amsterdam, Netherlands, June 2001.
5. Mayr, E.W., Meyer, A.R.: The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals. Adv. Math. **46** (1982) 305–329.
6. Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.
7. Schrijver, A.: *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester, 1986.
8. Wolsey, L.A.: *Integer Programming.* John Wiley & Sons, Inc., New York, 1998.

Straight-Line Drawings of General Trees with Linear Area and Arbitrary Aspect Ratio*

(Extended Abstract)

Ashim Garg and Adrian Rusu

Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260
{agarg,adirusu}@cse.buffalo.edu

Abstract. Trees are usually drawn planar, i.e. without any crossings. In this paper we investigate the area requirement of planar straight-line drawings of trees. A *degree-d tree* is one in which each node has at most d edges incident on it. Let T be a degree-d tree with n nodes, such that $d = O(n^{\delta})$, where $\delta < 1/2$ is a constant. We show that T admits a planar straight-line grid drawing with area $O(n)$ and with any prespecified aspect ratio in the range $[1, n^{\alpha}]$, where α is a constant such that $0 \leq \alpha < 1$. We also show that such a drawing can be constructed in $O(n \log n)$ time.

1 Introduction

A *drawing* Γ of a tree T maps each node of T to a distinct point in the plane and each edge (u, v) of T to a simple Jordan curve with endpoints u and v. Γ is a *straight-line* drawing if each edge is drawn as a single straight-line segment. Γ is a *grid* drawing if all the nodes have integer coordinates in it. Γ is a *planar* drawing if edges do not intersect each other in the drawing. In this paper, we concentrate on grid drawings. Let R be a rectangle with sides parallel to the X- and Y-axes. The *width* (*height*) of R is equal to the number of grid points with the same y (x) coordinate contained within R. The *area* of R is equal to the number of grid points contained within R. The *aspect ratio* of R is the ratio of its longer and shorter sides. R is the *enclosing rectangle* of Γ, if it is the smallest rectangle that covers the entire drawing. The *width*, *height*, *area*, and *aspect ratio* of Γ is equal to the width, height, area, and aspect ratio, respectively, of its enclosing rectangle. The *degree* of a node of T is the number of edges incident on it. The *degree* of T is equal to the maximum degree of a node of T. T is a *binary* tree if it has degree 3.

* Research supported by NSF CAREER Award IIS-9985136 and NSF CISE Research Infrastructure Award No. 0101244.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 876–885, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Our Result

It is generally desirable to draw trees using planar staight-line grid drawings with small area and give control to the user over its aspect ratio [3,1]. We, therefore, investigate the problem of constructing planar straight-line grid drawings of trees with small area and user-defined aspect ratio. Clearly, any planar grid drawing of a tree with n nodes requires $\Omega(n)$ area. A fundamental question, therefore, has been that whether this is a tight bound also, i.e., given a tree T with n nodes, can we construct a planar straight-line grid drawing of T with area $O(n)$?

In [3], it is shown that a binary tree can be drawn in this fashion in $O(n)$ area. However, trees with degree greater than 3 appear quite commonly in practical applications. Hence, an important natural question arises, if this result can be generalized to higher degree trees also. In this paper, we give a partial answer to this question, by giving an algorithm that constructs a planar straight-line grid drawing of an n-node tree with degree $O(n^{\delta})$, where $\delta < 1/2$ is any constant, in $O(n)$ area in $O(n \log n)$ time. Moreover, the drawing can be parameterized for its aspect ratio, i.e., for any constant α, where $0 \leq \alpha < 1$, the algorithm can construct a drawing with any user-specified aspect ratio in the range $[1, n^{\alpha}]$. Our result is summarized in Theorem 3. Our algorithm is a generalization of the algorithm of [3], which draws binary trees.

Previously, the best-known bound on area of planar straight-line grid drawings of general trees was $O(n \log n)$, which can be achieved by a simple modification of the HV-drawing algorithm of [2]. [1] studies the relationship between aspect ratio and area of planar straight-line grid drawings of binary trees.

3 Preliminaries

Throughout this paper, by the term *drawing*, we will mean a planar straight-line grid drawing. Let T be a degree-d tree, with one distinguished node v, which has at most $d-2$ children. v is called the *link* node of T. Let n be the number of nodes in T. T is an *ordered* tree if the children of each node are assigned a left-to-right order. A *partial tree* of T is a connected subgraph of T. If T is an ordered tree, then the *leftmost path* p of T is the maximal path consisting of nodes that are leftmost children, except the first one, which is the root of T. The last node of p is called the *leftmost* node of T. A *sibling* of a node v of T is a node that has the same parent as v. T is an *empty tree*, i.e., $T = \phi$, if it has zero nodes in it.

Let Γ be a drawing of T. Let R be a rectangle, such that Γ is entirely contained within R. R has a *good* aspect ratio, if its aspect ratio is in the range $[1, n^{\alpha}]$, where α is a constant, such that $0 \leq \alpha < 1$.

Let r be the root of T. Let u^* be the link node of T. Γ is a *feasible* drawing of T, if it has the following three properties:

- **Property 1**: The root r is placed at the top-left corner of Γ.
- **Property 2**: If $u^* \neq r$, then u^* is placed at the bottom boundary of Γ. Moreover, we can move u^* downwards in its vertical channel by any distance without causing any edge-crossings in Γ.

– **Property 3**: If $u^* = r$, then no other node or edge of T is placed on, or crosses the vertical and horizontal channels occupied by r.

Theorem 1 ([5]). *Let n_0 and α be two constants, where $n_0 \geq 1$ and $0 < \alpha < 1/2$. Let $m_1, m_2, \ldots, m_{n_0}$ be a set of positive numbers. Let $G(n)$, and $g(n)$ be two functions such that*

– *$\forall n > n_0$, and $\forall x$, where $0 < x \leq 1-\alpha$, $G(xn) \leq x(G(n) - g(n)\sqrt{G(n)})$, and*
– *$\forall n \leq n_0$, $G(n) \geq m_n$.*

Then, if $g(n) \leq cn^\beta$, where $c > 0$ and β are constants, and $0 \leq \beta < 1/2$, then $G(n) = O(n)$.

Theorem 2. *In any degree-d tree T, there is a node u, such that removing u and its incident edges splits T into at most d trees, where each tree has at most $(2/3)n$ nodes in it, where $n \geq 2$ is the number of nodes in T. Node u is called a* separator *of T. Moreover, u can be found in $O(n)$ time.*

4 Algorithm *u^*-HV-Draw*

We first describe Algorithm *u^*-HV-Draw*, which is used by our main drawing algorithm (described in Section 5). Algorithm *u^*-HV-Draw* constructs a feasible drawing of a tree T with a link node u^* within a large enough rectangle R, and is based on the well-known HV-Drawing algorithm of [2], which is defined for binary trees, but can be easily extended to higher-degree trees. Lemma 1 summarizes the main result on HV-Drawing algorithm:

Lemma 1 ([2]). *Let T be an n-node degree-d tree with root r. Let R be a rectangle with width (height) n and height (width) $\log_2 n + 2$. We can construct a drawing of T within R in $O(n)$ time, such that r is placed at the top-left corner of R.*

Algorithm *u^*-HV-Draw* can be described as follows: Let W and H be the width and height of R, respectively.

– Order the children of each node such that u^* becomes the leftmost node of T.
– Let $r = r_1, r_2, \ldots, r_{k-1}, r_k = u^*$, be the nodes in the leftmost path of T (see Figure 1(a)). Let $c_j^{i_j}$ be the other children of r_j, $1 \leq j \leq k$, respectively. Let $T_j^{i_j}$ be the subtrees of T rooted at $c_j^{i_j}$, $1 \leq j \leq k$, respectively. For each j, where $1 \leq j \leq k$, let $n_i^{i_j}$ be the number of nodes in $T_i^{i_j}$.
– If $W \leq H$ ($H < W$), for each j, where $1 \leq j \leq k$, construct a drawing $\Gamma_j^{i_j}$ of $T_j^{i_j}$, using HV-Drawing algorithm, within a rectangle $R_j^{i_j}$ with height (width) $n_j^{i_j}$ and width (height) $\log_2 n_j^{i_j} + 2$ (see Lemma 1).
– T is drawn by combining $\Gamma_1^1, \ldots, \Gamma_1^{i_1}, \ldots, \Gamma_k^1, \ldots, \Gamma_k^{i_k}$, as shown in Figures 1(b,c) (when $k \geq 2$) and Figures 1(e,f) (when $k = 1$). If $W \leq H$, then $\Gamma_1^1, \ldots, \Gamma_1^{i_1}, \ldots, \Gamma_k^1, \ldots, \Gamma_k^{i_k}$ are stacked one-above the other (see Figures 1(b,e)). If $W > H$, then $\Gamma_1^1, \ldots, \Gamma_1^{i_1}, \ldots, \Gamma_k^1, \ldots, \Gamma_k^{i_k}$ are placed left-to-right (see Figures 1(c,f)). We omit the details here for the lack of space, but have given the details in [4].

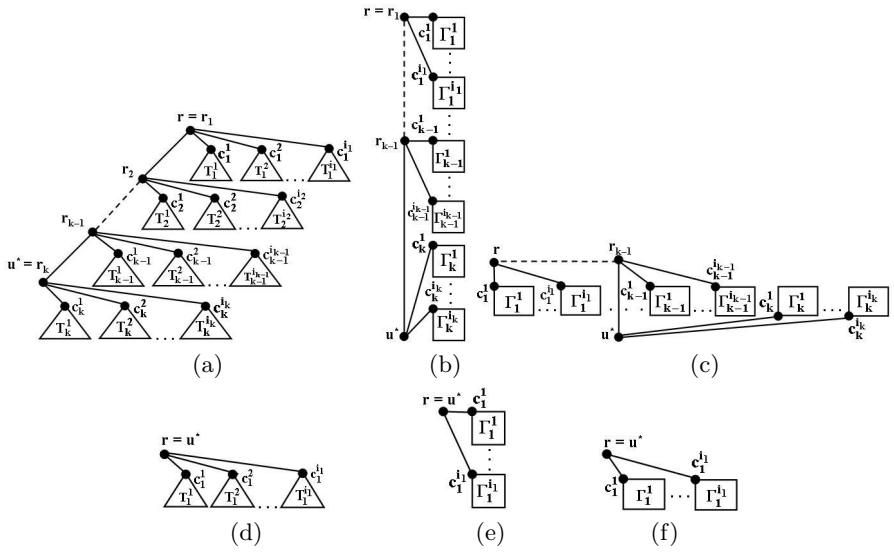


Fig. 1. (a) General form of a degree-d tree T with link node u^* and root r, where u^* is also the leftmost node of T. (b,c) Drawing T when $k \geq 2$: if (b) $W \leq H$, and if (c) $H < W$. (d) Tree T when $k = 1$. (e,f) Drawing T when $k = 1$: if (e) $W \leq H$, and if (f) $H < W$. For simplicity, we have shown $\Gamma_1^1, \ldots, \Gamma_1^{i_1}, \ldots, \Gamma_k^1, \ldots, \Gamma_k^{i_k}$ as identically sized boxes, but in actuality, they may have different sizes.

Lemma 2. *Let T be an n-node degree-d tree with a link node u^*. Algorithm u^*-*HV-Draw *can be used to construct in $O(n)$ time, a feasible drawing of T within a rectangle with width n and height $\log_2 n + 4$, as well as within a rectangle with width $\log_2 n + 4$ and height n.*

5 Our Overall Tree Drawing Algorithm

Let T be a degree-d tree with a link node u^*, where $d = O(n^\delta)$, for a constant $\delta < 1/2$, and n is the number of nodes in T. Let A and ϵ be two numbers such that $(2\delta + 1)/(3 - 2\delta) < \epsilon < 1$, and A is in the range $[1, n^\epsilon]$. Let R be a large enough rectangle with aspect ratio A, width W and height H (we will determine later, how large R should be). Notice that since $1 \leq A < n^\epsilon$, R has good aspect ratio.

Our tree drawing algorithm, called *DrawTree*, uses the divide-and-conquer paradigm to recursively construct a feasible drawing Γ of T within R, by performing the following actions at each recursive step:

- *Split Tree*: Split T into at most $2d - 1$ partial trees by removing at most two nodes and their incident edges from it. Each partial tree has at most $\frac{2}{3}n$ nodes. Based on the arrangement of these partial trees within T, we get two cases, which are shown in Figures 2 and 3, and described later in Section 5.1.

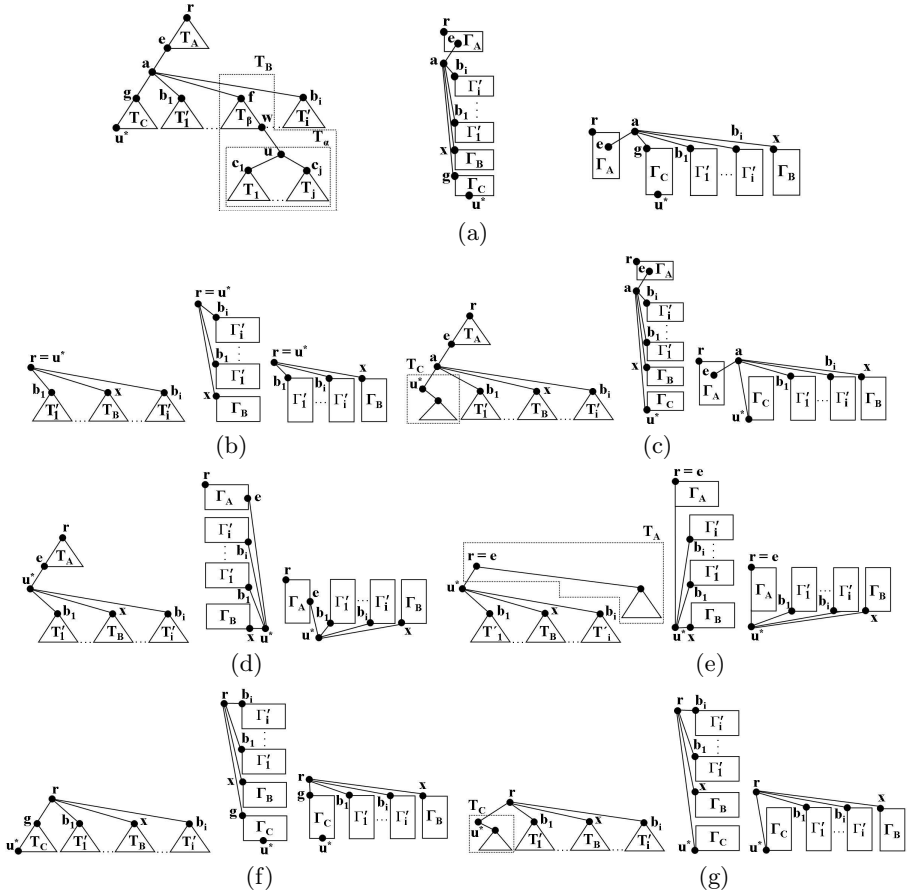


Fig. 2. Drawing T in all the subcases of Case 1 (when the separator node u is not in the leftmost path of T): (a) $T_A \neq \emptyset$, $T_C \neq \emptyset$, $g \neq u^*$, $0 \leq i \leq d-3$. (b) $T_A = \emptyset$, $T_C = \emptyset$, $0 \leq i \leq d-3$. (c) $T_A \neq \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$. (d) $T_A \neq \emptyset$, $T_C = \emptyset$, $r \neq e$, $0 \leq i \leq d-3$. (e) $T_A \neq \emptyset$, $T_C = \emptyset$, $r = e$, $0 \leq i \leq d-3$. (f) $T_A = \emptyset$, $T_C \neq \emptyset$, $g \neq u^*$, $0 \leq i \leq d-3$. (g) $T_A = \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$. For each subcase, we first show the structure of T for that subcase, then its drawing when $W \leq H$, and then its drawing when $H < W$. Note that $x = f$ if $T_\beta \neq \phi$, and $x = u$ if $T_\beta = \phi$. In Subcases (a) and (c), for simplicity, e is shown to be in the interior of Γ_A, but actually, either it is the same as r, or if $W \leq H$ ($H < W$), then it is placed on the bottom (right) boundary of Γ_A.

- *Assign Rectangles*: Correspondingly, split R into at most $2d-1$ smaller rectangles, and assign each smaller rectangle to a partial tree. The splitting is done by cutting the longer side (height or width) of R.
- *Draw Partial Trees*: Recursively construct a feasible drawing of each partial tree within its assigned rectangle.

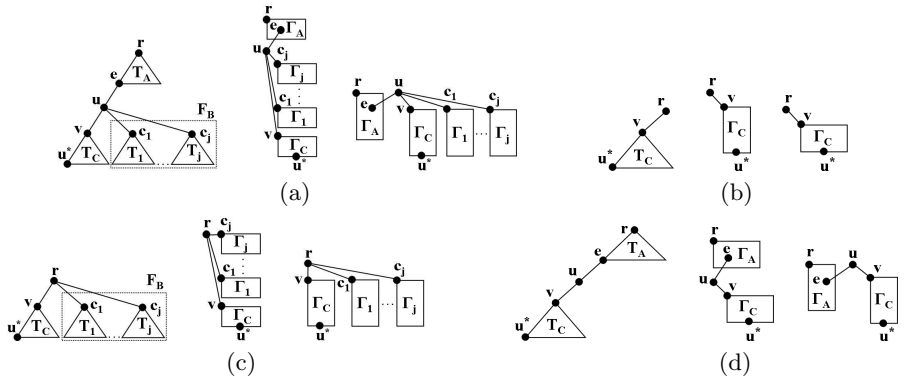


Fig. 3. Drawing T in the various subcases of Case 2 (when the separator node u is in the leftmost path of T): (a) $T_A \neq \emptyset, T_C \neq \emptyset$, and $1 \leq j \leq d-2$, (b) $T_A = \emptyset, T_C \neq \emptyset$, and $j = 0$, (c) $T_A = \emptyset, T_C \neq \emptyset$, and $1 \leq j \leq d-2$, and (d) $T_A \neq \emptyset, T_C \neq \emptyset$, and $j = 0$. For each subcase, we first show the structure of T for that subcase, then its drawing when $W \leq H$, and then its drawing when $H < W$. In Subcases (a) and (d), for simplicity, e is shown to be in the interior of Γ_A, but actually, either it is same as r, or if $W \leq H$ ($H < W$), then it is placed on the bottom (right) boundary of Γ_A. In subcase (e) $T_A = \emptyset, T_C = \emptyset$, and $1 \leq j \leq d-2$, the structure of T and its drawing is same as in Case 1(b) (see Figure 2(b)). In subcase (f) $T_A \neq \emptyset, T_C = \emptyset$, and $1 \leq j \leq d-2$, the structure of T and its drawing is same as in Case 1(d) (see Figure 2(d)).

– *Compose Drawings*: Within R, arrange the drawings of the partial trees, and draw the nodes and edges, that were removed from T to split it, such that the drawing Γ of T thus obtained is a feasible drawing. Note that the arrangement of these drawings is done based on the cases shown in Figures 2 and 3.

We now give details of each action performed by Algorithm *DrawTree*:

5.1 Split Tree

The splitting of tree T into partial trees is done as follows:

– Order the children of each node such that u^* becomes the leftmost node of T.
– Using Theorem 2, find a separator node u of T.
– Based on whether, or not, u is in the leftmost path of T, we get two cases:
  • *Case 1: u is not in the leftmost path of T.* We get seven subcases: (a) In the general case, T has the form as shown in Figure 2(a). In this figure: r is the root of T, $c_1, \ldots, c_j$ are the children of u, $T_1, \ldots, T_j$ are the trees rooted at $c_1, \ldots, c_j$ respectively, $0 \leq j \leq d-1$, T_α is the subtree rooted at u, w is the parent of u, a is the last common node of the path $r \rightsquigarrow v$ and the leftmost path of T, f is the child of a that is contained in the path $r \rightsquigarrow v$, T_β is the maximal tree rooted at f that contains w

but not u, T_B is the tree consisting of the trees T_α and T_β, and the edge (w, u), e is the parent of a, g is the leftmost child of a, T_A is the maximal tree rooted at r that contains e but not a, T_C is the tree rooted at g, $b_1, \ldots, b_i$ are the siblings of f and g, and $T'_1, \ldots, T'_i$ are the trees rooted at $b_1, \ldots, b_i$ respectively, $0 \leq i \leq d-3$.

In addition to this general case, we get six special cases: (b) $T_A = \emptyset$, $T_C = \emptyset$, $0 \leq i \leq d-3$ (see Figure 2(b)), (c) $T_A \neq \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$ (see Figure 2(c)), (d) $T_A \neq \emptyset$, $T_C = \emptyset$, $r \neq e$, $0 \leq i \leq d-3$ (see Figure 2(d)), (e) $T_A \neq \emptyset$, $T_C = \emptyset$, $r = e$, $0 \leq i \leq d-3$ (see Figure 2(e)), (f) $T_A = \emptyset$, $T_C \neq \emptyset$, $g \neq u^*$, $0 \leq i \leq d-3$ (see Figure 2(f)), and (g) $T_A = \emptyset$, $T_C \neq \emptyset$, $g = u^*$, $0 \leq i \leq d-3$ (see Figure 2(g)).

In each case, we remove nodes a and u, and their incident edges, to split T into at most $2d-1$ partial trees T_A, T_C, T_β, $T'_1, \ldots, T'_i$, $0 \leq i \leq d-3$, and $T_1, \ldots, T_j$, $0 \leq j \leq d-1$. We also designate e as the link node of T_A, w as the link node of T_β, and u^* as the link node of T_C. We randomly choose a leaf e_i of T'_i, $0 \leq i \leq d-3$, and a leaf e_j of T_j, $0 \leq j \leq d-1$, and designate them as the link nodes of T'_i and T_j, respectively.

- *Case 2: u is in the leftmost path of T.* We get four subcases: (a) In the general case, T has the form as shown in Figure 3(a). In this figure, r is the root of T, $c_1, \ldots, c_j$ are the siblings of v, $T_1, \ldots, T_j$ are the trees rooted at $c_1, \ldots, c_j$ respectively, $1 \leq j \leq d-2$, e is the parent of u, T_A is the maximal tree rooted at r that contains e but not u, and T_C is the tree rooted at the child v of u.

  In addition to the general case, we get the following five special cases: (b) $T_A = \emptyset, T_C \neq \emptyset$, and $j = 0$ (see Figure 3(b)), (c) $T_A = \emptyset, T_C \neq \emptyset$, and $1 \leq j \leq d-2$ (see Figure 3(c)), (d) $T_A \neq \emptyset, T_C \neq \emptyset$, and $j = 0$ (see Figure 3(d)), (e) $T_A = \emptyset, T_C = \emptyset$, and $1 \leq j \leq d-2$, and (f) $T_A \neq \emptyset, T_C = \emptyset$, and $1 \leq j \leq d-2$. In each case, we remove node u, and its incident edges, to split T into at most d partial trees T_A, T_C, and $T_1, \ldots, T_j$, $0 \leq j \leq d-2$. We also designate e as the link node of T_A, and u^* as the link node of T_C. We randomly select a leaf e_j of T_j and designate it as the link node of T_j.

5.2 Assign Rectangles

Let T_k be a partial tree of T, where for Case 1, T_k is either T_A, T_C, T_β, $T'_1, \ldots, T'_i$, $0 \leq i \leq d-3$, or $T_1, \ldots, T_j$, $0 \leq j \leq d-1$, and for Case 2, T_k is either T_A, T_C, or $T_1, \ldots, T_j$, $0 \leq j \leq d-2$. Let n_k be number of nodes in T_k. Algorithm *DrawTree* assigns a rectangle R_k, with width W_k and height H_k, to T_k, where W_k and H_k are as follows: Let W and H be the width and height, respectively, of R. Assume that $H < W$ (the case when $W \leq H$ is handled similarly). Let $H' = H - 2$, and $W' = W - (2d-1)\max\{(2n)^{\frac{1-\epsilon}{1+\epsilon}}, \log_2 n + 4\} - 2$. Let $x = n_k/n$.

- If $n_k < H'$ (i.e., if T_k is a *very small* tree), then $W_k = \log_2 n_k + 4$, and $H_k = H'$;

- Otherwise, if $xW'/H' \leq n_k^{-\epsilon}$ (i.e., if T_k is a *moderately large* tree), then $W_k = H'/n_k^{\epsilon}$, and $H_k = H'$;
- Otherwise (i.e., if T_k is a *large* tree), $W_k = xW'$ and $H_k = H'$.

5.3 Draw Partial Trees

In Subcase (d) of Case 1, and if $H < W$, then in Subcases (a) and (c) of Case 1, and Subcases (a) and (d) of Case 2, we first change the orientation of R_A, so that its height becomes its width, and its width becomes its height. Also, in Case 1, if $H < W$, and $T_\beta \neq \phi$, we change the orientation of R_β. This is done so because, in these situations, as explained later in Subsection 5.4, we need to rotate R_A and R_β during the *Compose Drawings* step. In all other situations, we do not change the orientation of the rectangle assigned to a partial tree.

Next, we draw each partial tree T_k within its assigned rectangle R_k. If T_k is a *large* tree or a *moderately large* tree (see their definitions in Subsection 5.2), then T_k is drawn within R_k by calling Algorithm *DrawTree* recursively for it, and if T_k is a *very small* tree, then T_k is drawn within R_k by calling Algorithm *u*-HV-Draw* for it. Note that recursion for *DrawTree* stops when either T is empty, or contains exactly one node, or gets designated as a very small tree (in which case, it is drawn using *u*-HV-Draw*).

5.4 Compose Drawings

Let Γ_k denote the drawing of a partial tree T_k constructed in Step *Draw Partial Trees*. We now describe the construction of a feasible drawing Γ of T from the drawings of the partial trees in both Cases 1 and 2. (For simplicity, in Figures 4, 5, 2, and 3, we have shown the drawings of each T_k with equal-sized rectangles, but in an actual drawing of T, they may have different sizes.)

In Case 1, we first construct a feasible drawing Γ_α of the partial tree T_α by composing $\Gamma_1, \ldots, \Gamma_j$, $0 \leq j \leq d-1$, as shown in Figure 4, then construct a feasible drawing Γ_B of T_B by composing Γ_α and Γ_β as shown in Figure 5, and finally construct Γ by composing Γ_A, Γ_B, Γ_C, and $\Gamma'_1, \ldots, \Gamma'_i$, $0 \leq i \leq d-3$, as shown in Figure 2.

Γ_α is constructed as follows (see Figure 4): If $W \leq H$, place $\Gamma_1, \Gamma_2, \ldots, \Gamma_j$, $1 \leq j \leq d-1$, one above the other, in this order, such that their left boundaries are aligned. Place u at one unit to the left of c_j. Otherwise (i.e., if $W > H$) place $\Gamma_1, \Gamma_2, \ldots, \Gamma_j$, $1 \leq j \leq d-1$ in a left-to-right order, such that their top boundaries are aligned. Place u one unit above c_1.

Γ_B is constructed as follows (see Figure 5):

- if $T_\beta \neq \emptyset$ (see Figure 5(a)) then, if $W \leq H$, place Γ_β above Γ_α such that the left boundaries of Γ_β and Γ_α are aligned; otherwise (i.e., if $W > H$), first rotate Γ_β counterclockwise by 90° and then flip it upside-down, and then place Γ_β to the left of Γ_α such that the top boundaries of Γ_β and Γ_α are aligned.
- Otherwise (i.e., if $T_\beta = \emptyset$), Γ_B is the same as Γ_α (see Figure 5(b)).

We now describe the construction of Γ from Γ_A, Γ_B, and Γ_C in Case 1(d), which, we think, is the most technically-intricate case for constructing Γ: See Figure 2(d). If $W \leq H$, flip $\Gamma'_i, \ldots, \Gamma'_1, \Gamma_B$, first upside-down, and then left-to-right. First rotate Γ_A counterclockwise by 90°, and then flip it upside-down. Next, place $\Gamma_A, \Gamma'_i, \ldots, \Gamma'_1, \Gamma_B$ one above the other with unit vertical separation, such that their left boundaries are aligned, next move node e (which is the link node of T_A) to the right until it is either to the right of, or aligned with the rightmost boundary among $\Gamma'_i, \ldots, \Gamma'_1, \Gamma_B$ (since Γ_A is a feasible drawing, by Property 2, as given in Section 3, moving e will not create any edge-crossings), next move $\Gamma'_i, \ldots, \Gamma'_1, \Gamma_B$ to the right until their right boundaries are aligned with e, and then place u^* one unit to the right of x. If $H < W$, first rotate Γ_A counterclockwise by 90°, and then flip it upside-down. Flip $\Gamma'_i, \ldots, \Gamma'_1, \Gamma_B$ upside-down. Then, place Γ_A, u^*, $\Gamma'_1, \ldots, \Gamma'_i, \Gamma_B$ left-to-right in that order separated by unit horizontal distances, such that the top boundaries of $\Gamma_A, \Gamma'_1, \ldots, \Gamma'_i, \Gamma_B$ are aligned, and u^* is placed one unit below the bottom boundary of the drawing among $\Gamma_A, \Gamma'_1, \ldots, \Gamma'_i, \Gamma_B$ with the greatest height.

Construction of Γ from Γ_A, Γ_B and Γ_C in the other subcases of Case 1 and in Case 2 is similar, and is shown in Figures 2(a–g), and 3, respectively. We omit the details of the constructions here for lack of space, but have give them in [4].

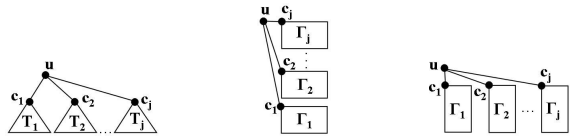


Fig. 4. Drawing T_α, Here, we first show the structure of T_α. then its drawing when $W \leq H$, and then its drawing when $H < W$.

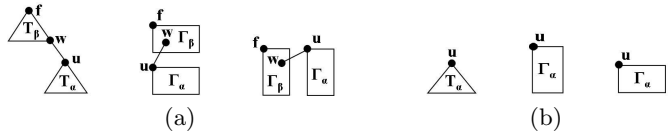


Fig. 5. Drawing T_B when: (a) $T_\beta \neq \emptyset$, and (b) $T_\beta = \emptyset$. For each case, we first show the structure of T_B for that case, then its drawing when $W \leq H$, and then its drawing when $H < W$. In Case (a), for simplicity, w is shown to be in the interior of Γ_β, but actually, it is either same as f, or if $W \leq H$ $(H < W)$, then is placed on the bottom (right) boundary of Γ_β.

Lemma 3 (Planarity and Time). *Given a degree-d tree T with a link node u^*, Algorithm* DrawTree *will construct a feasible drawing Γ of T in $O(n \log n)$ time.*

Lemma 4 (Area). *Let T be a degree-d tree with n nodes, where $d = O(n^\delta)$, for a constant $\delta < 1/2$. It is sufficient for rectangle R to have $O(n)$ area for Algorithm* DrawTree *can construct a drawing of T entirely contained within R.*

Sketch of Proof. The proof is similar to the proof of Theorem 5 of [5]. Let $D(n)$ be the area of R. We show that if $D(n)$ satisfies the following recurrence, then Algorithm *DrawTree* can draw T entirely within R: $D(xn) \leq x(D(n) - ((2d-1)\max\{(2n)^{\frac{1-\epsilon}{1+\epsilon}}, \log_2 n + 4\} + 2n^{\epsilon/2} + 2)\sqrt{D(n)})$, where $0 < x < 1$ is a constant. (We omit the proof here for the lack of space. The complete proof is given in [4].) Since $(2\delta+1)/(3-2\delta) < \epsilon < 1$, it then follows that $\delta(1-\epsilon)/(1+\epsilon)$ and $\epsilon/2$ are both less than $1/2$. Since $d = O(n^\delta)$, it will follow that $(2d-1)\max\{(2n)^{\frac{1-\epsilon}{1+\epsilon}}, \log_2 n + 4\} + 2n^{\epsilon/2} + 2 \leq pn^\beta$, for some constants p and β, where $p > 0$, and $0 \leq \beta < 1/2$. Hence, from Theorem 1, this proves that $D(n) = O(n)$.

Theorem 3 (Main Theorem). *Let T be a degree-d tree with n nodes, where $d = O(n^\delta)$, for a constant $\delta < 1/2$. Given any number A, where $1 \leq A \leq n^\alpha$, for some constant α, where $0 \leq \alpha < 1$, Algorithm* DrawTree *can construct in $O(n \log n)$ time, a planar straight-line grid drawing Γ of T with $O(n)$ area, and aspect ratio A.*

Proof. Let ϵ be a constant such that $1 \leq A \leq n^\epsilon$ and $(2\delta + 1)/(3 - 2\delta) < \epsilon < 1$. Let R be a rectangle with area $D(n)$ and aspect ratio A, where $D(n)$ is the function defined in the proof of Lemma 4. Note that from the proof of Lemma 4, $D(n) = O(n)$. Designate any leaf of T as its link node u^*. Construct a drawing Γ of T in R by calling Algorithm *DrawTree* with T, R and ϵ as input. From Lemmas 3, and 4, Γ will be a planar straight-line grid drawing of T contained entirely within R.

References

1. T. Chan, M. Goodrich, S. Rao Kosaraju, and R. Tamassia. Optimizing area and aspect ratio in straight-line orthogonal tree drawings. *Comput. Geom. Theory Appl.*, 23:153–162, 2002.
2. P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom. Theory Appl.*, 2:187–200, 1992.
3. A. Garg and A. Rusu. Straight-line drawings of binary trees with linear area and arbitrary aspect ratio. In *Proc. Graph Drawing'02*, pages 320–331, 2002.
4. A. Garg and A. Rusu. Straight-line drawings of general trees with linear area and arbitrary aspect ratio. Technical report no. 2002-14, Dept. of Computer Science and Engineering, University at Buffalo, Buffalo, NY, 2002.
5. L. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comput.*, C-30(2):135–140, 1981.

Connected Guards in Orthogonal Art Galleries

Val Pinciu

Mathematics Department
Southern Connecticut State University
New Haven, CT 06515
pinciuv1@southernct.edu

Abstract. In this paper we consider a variation of the Art Gallery Problem for orthogonal polygons. A set of points $\mathcal{G}$ in a polygon P_n is a connected guard set for P_n provided that is a guard set and the visibility graph of the set of guards $\mathcal{G}$ in P_n is connected. The polygon P_n is orthogonal provided each interior angle is 90° or 270°. First we use a coloring argument to prove that the minimum number of connected guards which are necessary to watch any orthogonal polygon with n sides is $n/2 - 2$. This result was originally established by induction by Hernández-Peñalver. Then we prove a new result for art galleries with holes: we show that $n/2 - h$ connected guards are always sufficient to watch an orthogonal art gallery with n walls and h holes. This result is sharp when $n = 4h + 4$. We also construct galleries that require at least $n/2 - h - 1$ connected guards, for all n and h.

1 Introduction

Throughout this paper P_n denotes a simple closed polygon with n sides, together with its interior. A point x in P_n is *visible* from point w provided the line segment wx does not intersect the exterior of P_n. (Every point in P_n is visible from itself.) A set of points $\mathcal{G}$ is a *guard set* for P_n provided that for every point x in P_n there exists a point w in $\mathcal{G}$ such that x is visible from w. We define $VG(\mathcal{G}, P_n)$ the visibility graph of the set of guards $\mathcal{G}$ in a polygon P_n as follows: the vertex set is the guard set $\mathcal{G}$, and two vertices are connected with an edge if the guards are visible from each other.

A guard set for P_n gives the positions of stationary guards who can watch over an art gallery with shape P_n. Let $g(P_n)$ denote the minimum cardinality of a guard set for P_n. Chvátal's celebrated art gallery theorem [1] asserts that among all polygons with n sides ($n \geq 3$), the maximum value of $g(P_n)$ is $\lfloor n/3 \rfloor$. The orthogonal art gallery theorem of Kahn, Klawe, Kleitman [5] shows that among all orthogonal polygons with n sides ($n \geq 4$), the maximum value of $g(P_n)$ is $\lfloor n/4 \rfloor$.

Over the years numerous "art gallery problems" have been proposed and studied with different restrictions placed on the shape of the galleries or the powers of the guards. (See the monograph by O'Rourke [9], and the surveys by Shermer [10] and Urrutia [11].) In this paper we analyze another variation

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 886–893, 2003.
© Springer-Verlag Berlin Heidelberg 2003

of art gallery problems. A set of points $\mathcal{G}$ in a polygon P_n is a *connected guard* or a *cooperative guard* set for P_n if it is a guard set for P_n, and $VG(\mathcal{G}, P_n)$ is connected. A set of points $\mathcal{G}$ in a polygon P_n is a *guarded guard* or a *weakly cooperative guard* set for P_n if it is a guard set for P_n, and $VG(\mathcal{G}, P_n)$ has no isolated vertices. Let $cg(P_n)$ and $gg(P_n)$ denote the minimum cardinality of a connected guard set and a guarded guard set for P_n respectively. Liaw, Huang, and Lee [6], [7] show that the computation of $cg(P_n)$ and $gg(P_n)$ is an NP-hard problem.

Several art gallery theorems provide explicit formulas for the functions

$$cg(n) = \max\{cg(P_n) : \ P_n \text{ is a polygon with } n \text{ sides}\},$$

$$cg_\perp(n) = \max\{cg(P_n) : \ P_n \text{ is an orthogonal polygon with } n \text{ sides}\},$$

$$gg(n) = \max\{gg(P_n) : \ P_n \text{ is a polygon with } n \text{ sides}\},$$

$$gg_\perp(n) = \max\{gg(P_n) : \ P_n \text{ is an orthogonal polygon with } n \text{ sides}\}.$$

Hernández-Peñalver [4] showed by induction that $gg_\perp(n) = \lfloor n/3 \rfloor$. T.S. Michael and the author [8] give a completely different proof based on a coloring argument. They also disprove a result of Hernández-Peñalver for arbitrary art galleries, and show that $gg(n) = \lfloor (3n-1)/7 \rfloor$. Also, Hernández-Peñalver [3] showed by induction that $cg(n) = \lfloor (n-2)/2 \rfloor$. From here it easily follows that $cg_\perp(n) = n/2 - 2$.

In this paper we give a coloring proof for the last result. We also generalize this to art galleries with holes. We show that $n/2 - h$ connected guards are always sufficient to cover an orthogonal art gallery with n walls, and h holes. We also show that for all n and h, there exist art galleries with n walls and h holes that require at least $n/2 - h - 1$ connected guards, and when $n = 4h + 4$, there exist art galleries with n walls and h holes that require at least $n/2 - h$ connected guards.

2 Simply Connected Art Galleries

In this section, we consider simply connected art galleries, bounded by a polygon P_n. Also

$$cg_\perp(n) = \max\{cg(P_n) : \ P_n \text{ is an orthogonal polygon with } n \text{ sides}\}.$$

Theorem 1. *For $n \geq 6$ we have*

$$cg_\perp(n) = \frac{n}{2} - 2. \tag{1}$$

Proof. We begin our proof by constructing extremal polygons. Orthogonal polygons P_n shaped like the one in Figure 1, with k interior angles of measure 270°, and $k + 4$ interior angles of measure 90°, have $n = 2k + 4$ sides, and require at least $cg(P_n) = k = n/2 - 2$ connected guards, therefore $cg(n) \geq n/2 - 2$.

In order to show the reverse inequality, we start with an arbitrary orthogonal polygon P_n, and we will show that P_n has a connected guard set with no more than $n/2-2$ guards. Since P_n is orthogonal, by a result of Kahn, Klawe, Kleitman [5], P_n has a convex quadrangulation, i.e. P_n has a decomposition into convex quadrilaterals by means of diagonals. Let Q_n be the quadrangulation graph naturally induced on the set of vertices of P_n. Note that Q_n is planar and bipartite with an even number of vertices. The (weak) planar dual of Q_n (i.e. the graph with a vertex for each bounded face of Q_n, where two vertices are adjacent provided the corresponding faces share an edge), is a tree. Since Q_n is bipartite, it is 2-colorable, that is, there exists a map from the vertex set to the color set $\{1,2\}$ such that adjacent vertices receive different colors. In Figure 2 we see the quadrangulation graph and a 2-coloring for an orthogonal polygon.

The least frequently used color gives us a set of vertices $\mathcal{G}'$ such that (i) $|\mathcal{G}'| \leq n/2$; (ii) every quadrilateral of Q_n contains exactly two vertices of $\mathcal{G}'$. Let's define the set of vertices $\mathcal{G} = \{v \in \mathcal{G}' : \deg(v) \neq 2 \text{ in } Q_n\}$. We claim that $\mathcal{G}$ is a connected set of guards for P_n, and $|\mathcal{G}| \leq n/2-2$. The set of connected guards $\mathcal{G}$ that corresponds to the the polygon from Figure 2 is shown in Figure 3.

Since the planar dual of Q_n is a tree, it has at least two leaves. Each quadrilateral that corresponds to a leaf has two adjacent vertices of degree 2 in the quadrangulation graph Q_n. Since they are adjacent, they must have different colors, so there exist at least two vertices of each color that have degree 2. And since $|\mathcal{G}'| \leq n/2$, we obtain $|\mathcal{G}| \leq n/2-2$. In order to see that $\mathcal{G}$ is a connected guard set for P_n, for each quadrilateral Q in the quadrangulation of P_n, we define $v(Q)$ to be the set of vertices of Q that are in $\mathcal{G}$. Since $n \geq 6$, a quadrilateral cannot have two opposite vertices that have degree 2 in Q_n, so $v(Q) \neq \emptyset$ for all quadrilaterals Q. And because the quadrangulation is convex, we obtain that $\mathcal{G}$ is a guard set. Also, two guards that are in $v(Q)$ for a quadrilateral Q are visible to each other, and if two quadrilaterals Q and Q' are adjacent, then $v(Q) \cap v(Q') \neq \emptyset$, since one of the endpoints of the diagonal that separates the two quadrilaterals has to be in $\mathcal{G}$. From here, together with the fact that the weak planar dual graph of Q_n is a tree, therefore connected, we obtain that $\mathcal{G}$ is a connected guard set.

□

3 Orthogonal Art Galleries with Holes

In this section we will consider art galleries with holes. Let $P_{n,h}$ be the boundary of such an art gallery, where n is the number of walls, and h is the number of holes. Obviously $n \geq 4h+4$. We define

$$cg_{\perp}(n,h) = \max\{cg(P_{n,h}) : \ P_{n,h} \text{ is an orth art gallery with } n \text{ sides and } h \text{ holes}\}.$$

Theorem 2. *For all $n \geq 8$ and $h \geq 1$ such that $n \geq 4h+4$ we have*

$$cg_{\perp}(n,h) \leq \frac{n}{2} - h. \tag{2}$$

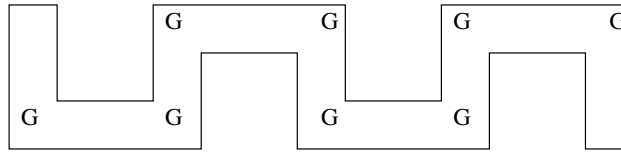


Fig. 1. Orthogonal polygon P_n for which $cg(P_n)$ is maximum.

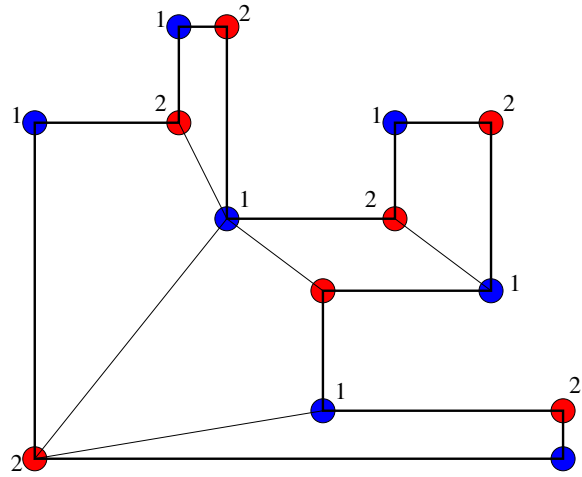


Fig. 2. A quadrangulation graph Q_n and a 2-coloring.

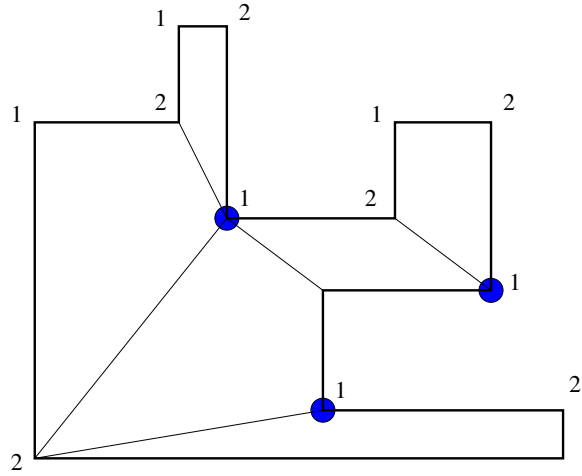


Fig. 3. A connected guard set for the polygon in Figure 2

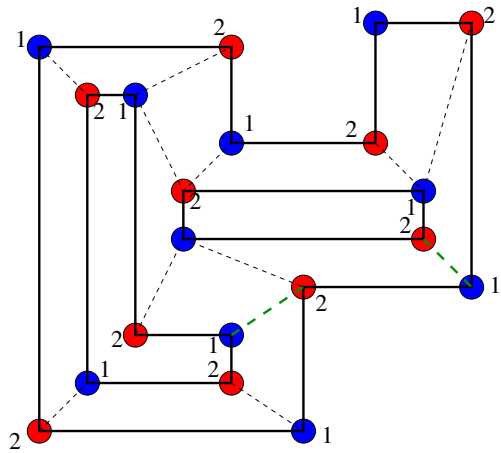


Fig. 4. A quadrangulation graph $Q_{n,h}$ and a 2-coloring. The solid lines represent the walls of the art gallery, and the dashed lines are diagonals used for the quadrangulation, where the thicker dashed lines represent the diagonals in D.

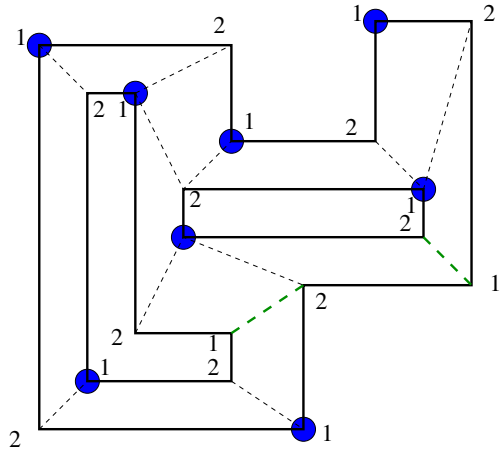


Fig. 5. A connected guard set for the art gallery with holes in Figure 4.

Proof. Let's start with an arbitrary orthogonal art gallery $P_{n,h}$ with n walls and h holes. We will show that $P_{n,h}$ has a connected guard set with no more than $n/2 - h$ guards. Since $P_{n,h}$ is orthogonal, by a result of Kahn, Klawe, Kleitman [5], $P_{n,h}$ has a convex quadrangulation. Also there exists a set D of h diagonals used in the quadrangulation of $P_{n,h}$, any two disjoint, that connect distinct pairs of connected components of the boundary of $P_{n,h}$. Let $Q_{n,h}$ be the quadrangulation graph naturally induced on the set of vertices of $P_{n,h}$. Note that $Q_{n,h}$ is planar and bipartite with an even number of vertices. The (weak) planar dual of $Q_{n,h}$ (i.e. the graph with a vertex for each bounded face of $Q_{n,h}$, where two vertices are adjacent provided the corresponding faces share an edge), is a connected graph with $n/2-1$ vertices and $n/2+h-2$ edges. We can obtain a spanning tree for the dual graph of $Q_{n,h}$ the following way: we delete an edge that connects two vertices if the corresponding quadrilaterals share an element of D. Since $Q_{n,h}$ is bipartite, it is 2-colorable, that is, there exists a map from the vertex set to the color set $\{1, 2\}$ such that adjacent vertices receive different colors. In Figure 4 we see the quadrangulation graph and a 2-coloring for an orthogonal art gallery with holes.

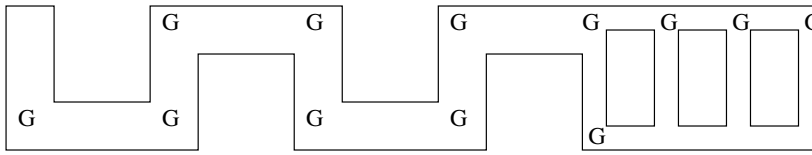


Fig. 6. An orthogonal art gallery $P_{n,h}$ for which $cg(P_{n,h}) = n/2 - h - 1$.

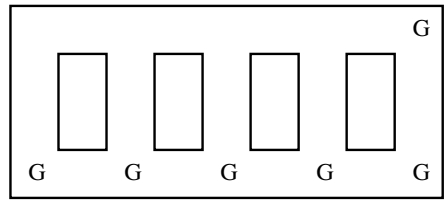


Fig. 7. An orthogonal art gallery $P_{n,h}$ for which $cg(P_{n,h}) = n/2 - h$.

The least frequently used color gives us a set of vertices $\mathcal{G}'$ such that (i) $|\mathcal{G}'| \leq n/2$; (ii) every quadrilateral of Q_n contains exactly two vertices of $\mathcal{G}'$. Let's define the set of vertices $\mathcal{G} = \{v \in \mathcal{G}' : \ v$ is not an endpoint for a diagonal in $D\}$. We claim that $\mathcal{G}$ is a connected set of guards for $P_{n,h}$, and $|\mathcal{G}| \leq n/2 - h$. The set

of connected guards $\mathcal{G}$ that corresponds to the the art gallery from Figure 4 is shown in Figure 5.

Indeed, since $|\mathcal{G}'| \leq n/2$, and every diagonal in D has an endpoint of each color, we obtain $|\mathcal{G}| \leq n/2 - h$. In order to see that $\mathcal{G}$ is a connected guard set for $P_{n,h}$, for each quadrilateral Q in the quadrangulation of $P_{n,h}$, we define $v(Q)$ to be the set of vertices of Q that are in $\mathcal{G}$. From the way the set D was constructed, no more than one element of D can be a side of a given quadrilateral Q, so $v(Q) \neq \emptyset$ for all quadrilaterals Q. (otherwise two elements of D would not be disjoint or would connect identical pairs of connected components of the boundary of $P_{n,h}$.) And because the quadrangulation is convex, we obtain that $\mathcal{G}$ is a guard set. Also, two guards that are in $v(Q)$ for a quadrilateral Q are visible to each other, and if two quadrilaterals Q and Q' are adjacent and are not separated by an element of D, then $v(Q) \cap v(Q') \neq \emptyset$, since one of the endpoints of the diagonal that separates the two quadrilaterals has to be in $\mathcal{G}$. From here together with the fact that the constructed spanning tree of the planar dual graph of $Q_{n,h}$ is connected, we obtain that $\mathcal{G}$ is a connected guard set.

□

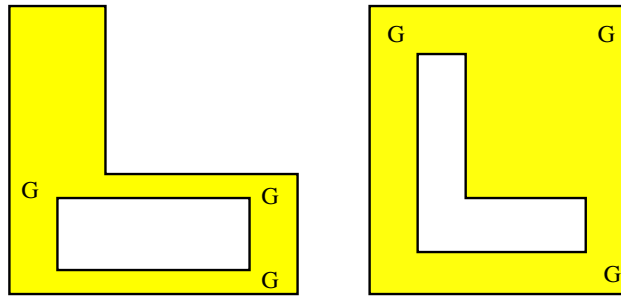


Fig. 8. An orthogonal art gallery with 10 walls and 1 hole has to be either an L shaped gallery with a rectangular hole, or a rectangular gallery with an L-shaped hole. Each gallery requires 3 connected guards.

Theorem 3. *For all $n \geq 8$ and $h \geq 1$ such that $n \geq 4h + 4$ we have*

$$cg_{\perp}(n, h) \geq \frac{n}{2} - h - 1. \tag{3}$$

Moreover, if $n = 4h + 4$ then

$$cg_{\perp}(n, h) = \frac{n}{2} - h. \tag{4}$$

Proof. By adding h rectangular holes to the orthogonal polygon from Figure 1 we obtain orthogonal art galleries $P_{n,h}$ shaped like the one in Figure 6. The

outside polygon has k interior angles of measure 270°, and $k+4$ interior angles of measure 90°. The gallery has $n = 2k + 4 + 4h$ walls, and requires at least $cg(P_{n,h}) = k + h + 1 = n/2 - h - 1$ connected guards, therefore $cg_{\perp}(n,h) \geq n/2 - h - 1$.

If $n = 4h + 4$, then the gallery is a rectangular gallery with h rectangular holes. An art gallery like the one in Figure 7 requires at least $h + 2 = n/2 - h$ connected guards, therefore by Theorem 2 we obtain $cg_{\perp}(n,h) = n/2 - h$.

□

Remark: Combining the results from Theorems 2 and 3 we obtain $cg_{\perp}(n,h) = n/2 - h - \epsilon$, where ϵ could be either 0 or 1. We saw that if $n = 4h+4$, $\epsilon = 0$, so a natural question if ϵ can ever be 1. And the answer is yes, for example when $n = 10$ and $h = 1$, the art gallery has to be either an L shaped gallery with a rectangular hole, or a rectangular gallery with an L-shaped hole, like the two galleries in Figure 8. In either case, the gallery requires only three guards, so $cg_{\perp}(10,1) = 3 = 10/2 - 1 - 1$. Therefore ϵ can be 1. An open question is to determine all values of n and h for which ϵ is 1.

References

1. V. Chvátal, A combinatorial theorem in plane geometry, *J. Combin. Theory Ser. B*, **18** (1975), 39–41.
2. S. Fisk, A short proof of Chvátal's watchman theorem, *J. Combin. Theory Ser. B*, **24** (1978), 374.
3. G. Hernández-Peñalver, Controlling guards (extended abstract), in: *Proceedings of the Sixth Canadian Conference on Computational Geometry (6CCCG), (1994),* pp. 387–392.
4. G. Hernández-Peñalver, Vigilancia vigilada de polígonos ortogonales, in: *Actes del VI Encuentros de Geometria Computacional, Barcelona, Spain (1995),* pp. 98–205.
5. J. Kahn, M. Klawe, and D. Kleitman, Traditional galleries require fewer watchmen, *SIAM J. Alg. Disc. Meth.,* **4** (1983), 194-206.
6. B.-C. Liaw, N.F. Huang, and R.C.T. Lee, The minimum cooperative guards problem on k-spiral polygons (Extended Abstract), in *Proc. 5-th Canadian Conf. on Computational Geometry (5CCCG),* Waterloo, Ontario, Canada, (1993), 97–102.
7. B.-C. Liaw and R.C.T. Lee, An optimal algorithm to solve the minimum weakly cooperative guards problem for 1-spiral polygons, *Inform. Process. Lett.,* **57** (1994), 69–75.
8. T.S. Michael and V. Pinciu, Art gallery theorems for guarded guards, to appear in *Computational Geometry: Theory and Applications.*
9. J. O'Rourke, *Art Gallery Theorems,* Oxford University Press, 1987.
10. T.C. Shermer, Recent results in art gallery theorems, *Proc. IEEE,* **80** (1992), 1384–1399.
11. J. Urrutia, Art gallery and illumination problems, *Handbook of Computational Geometry,* J.-R Sack and J. Urrutia eds., Elsevier Science B.V., 1999, 973-1027.

Four Colouring the Vertices of the Triangulation of a Polygon Containing a Hole

Graham M. Seed[1], Douglas E.R. Clark[2], Raffaella Ocone[1] and Xiaoyan Y. Yang[1]

[1] School of Engineering and Physical Sciences,
Heriot-Watt University, Edinburgh, EH14 4AS, Scotland
{g.m.seed, r.ocone, x.yang}@hw.ac.uk
[2] School of Mathematical and Computing Sciences,
Heriot-Watt University, Edinburgh, EH14 4AS, Scotland
d.e.r.clark@hw.ac.uk

Abstract. A simple linear-time algorithm is presented for four-colouring the vertices of a triangulation of a polygon containing a single hole. The algorithm consists of reducing a triangulation by the removal of both polygon and hole ear vertices, if any, followed by the removal of colour-isolated vertices until a 3-coloured tessellation remains. The triangulation is then re-built, using at most four colours. The paper concludes by recognising the similarity between the colouring of triangulations of polygons containing a hole and the colouring of bipartite and permutation graphs.

1 Introduction

It is a well known property of polygon triangulations that the vertices can be coloured using a maximum of three colours such that no two adjacent vertices share the same colour, see [1]. The proof follows from Meister's two ears theorem in that ear vertices can be repeatedly removed until the polygon is reduced to a single triangle. Incrementally reinserting the degree two ear vertices requires a maximum of three colours so that no colour clashes occur. In the case of a polygon containing a hole, this technique is insufficient since the triangulation may contain no ear triangles. This paper presents an algorithm for four-colouring triangulations of a polygon containing a single hole. As with the three-colouring of polygon triangulations, the presented algorithm is simple, linear-time and uses reduction.

The next section presents definitions and properties that will prove useful. This is followed in §3 by a discussion on ear-reduced polygons and holes. Section 4 presents the main result of this paper, the four-colouring of a triangulation of a polygon containing a hole. Section 5 presents an analogy between the colouring of ear-reduced polygon and hole triangulations and the colouring of bipartite and permutation graphs. The main conclusions that can be drawn from the present study are presented in §6.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 894-902, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Definitions and Properties

A polygon, P, is the bounded region enclosed by a finite, closed cycle of edges, E_p. Each edge consists of a pair of vertices, with two adjacent edges sharing a single vertex. The unique set of $|E_p|(\geq 3)$ edge end-points represents the set of $|E_p|$ polygon vertices, V_p (v_i; i=0,...,$|V_p|$-1). The degree, d_i, of a vertex i is the number of edges incident with that vertex. A polygon is referred to as even or odd if it has an even and odd number of vertices respectively.

A triangulation of a polygon P, $T(P)$, consists of a set of $|V_p|$-2 triangles, whose edges are the $|V_p|$ edges of P and the $|V_p|$-3 diagonals of P. A diagonal of P is a line segment whose end-points are vertices of P. Two diagonals are non-crossing such that their intersection is a subset of their end-points.

A hole, H, is a polygon with vertices V_h (v_i; i=0,...,$|V_h|$-1) contained entirely within the region of a bounding polygon. Vertex sets V_p and V_h have no vertices in common. In other words, the region between P and H is bounded and closed and a closed curve can be constructed that encloses H in the region between P and H. The region enclosed by a hole is empty and contains no diagonals.

An ear triangle of $T(P)$ is defined as a triangle containing an ear vertex, that is a vertex of degree 2 with no incident diagonal. Meister's two ears theorem, [1], states that $T(P)$ has at least two ear triangles. Let P' and H' denote an ear-reduced polygon and ear-reduced hole, respectively, in which all ear vertices have been removed. As a result, triangulation $T(P'H')$ consists entirely of diagonals that are interlayer edges extending between P' and H' and a triangle is one of two types: i) vertex on P' and edge on H' or ii) edge on P' and vertex on H'. Let $T(P'H')$ contain $|V_{p'h'}|=|V_{p'}|+|V_{h'}|$ vertices. Inserting a cut along a diagonal of $P'H'$ that joins P' and H' eliminates the hole and increases the number of vertices by 2. Thus, $T(P'H')$ consists of $|V_{p'h'}|$ triangles. From Euler's formula (V-E+F-2=0) the total number of edges and diagonals of $T(P'H')$ is $2|V_{p'h'}|$. Therefore, the number of interlayer edges is $2|V_{p'h'}|$-$(|V_{p'}|+|V_{h'}|)=|V_{p'}|+|V_{h'}|$. For complete interlayer connectivity (every vertex of P' is connected to every vertex of H') then the number of diagonals would equal $|V_{p'}||V_{h'}|$. Hence, the number of *missing* interlayer edges is $|V_{p'}||V_{h'}|$-$(|V_{p'}|+|V_{h'}|)$. For $|V_{p'}|,|V_{h'}|\geq 3$ the number of missing interlayer edges is non-zero, implying that not all $V_{p'}$ are visible to all $V_{h'}$.

An interesting property of $T(P'H')$ is that the average vertex degree is equal to 4 and follows from Euler's formula and that the sum of all vertex degrees is equal to twice the number of edges; namely $\sum d_i = 2\left(|V_{p'}|+|V_{h'}|\right)$. This informs us that certain $T(P'H')$ exist in which all vertices are of degree 4; one such example is shown in Figure 1. As a consequence, it is not possible to repeatedly eliminate vertices of degree 3 or less followed by using a fourth colour upon reinserting the removed vertices; informing us that an alternative technique must be sought.

A k-colouring(V_p) of $T(P)$ uses k colours or less such that no two adjacent vertices share the same colour. If $T(P)$ is k-colourable but not (k-1)-colourable then the chromatic number of $T(P)$ is k, and similarly for $T(PH)$. It is known that $T(P)$ is 3-colourable and §4 will show that $T(P'H')$ is 4-colourable but first we need to demonstrate that $T(PH)$ can be reduced to $T(P'H')$.

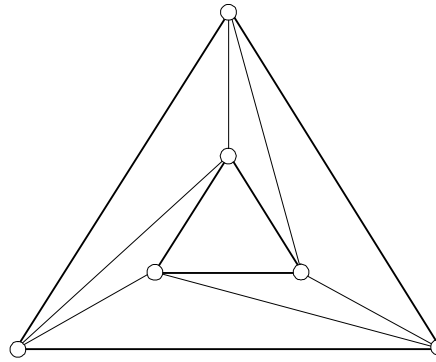


Fig. 1. Triangulation $T(P'H')$ in which all vertices are of degree 4.

3 Removal of Polygon and Hole Ears

Some diagonals of $T(PH)$ may have both their end-points on P, see Figure 2a). The ear vertices associated with these diagonals are removed and placed on a stack until all diagonals are interlayer edges with one end-point lying on P' and the other on H. Figure 2b) illustrates that $T(P'H)$ contains no ear vertices because all $|V_{p'}|+|V_h|$ diagonals are interlayer edges.

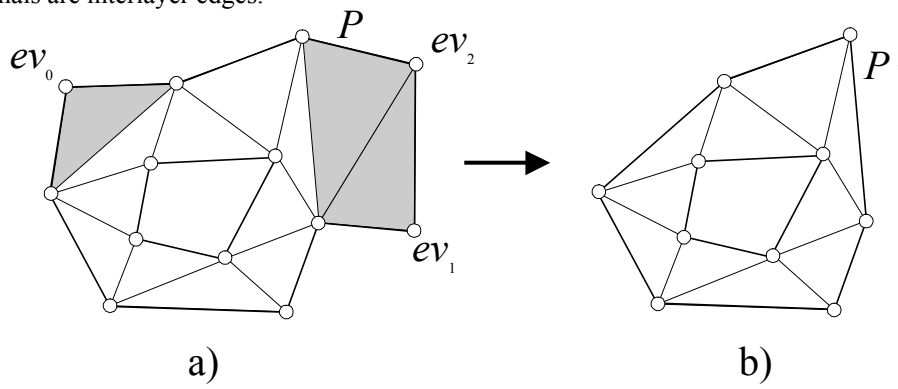


Fig. 2. Triangulations. a) Original, $T(PH)$. b) Ear-reduced polygon, $T(P'H)$, with stack of ear vertices $[ev_2,ev_1,ev_0]$ removed.

It may be required to apply a similar process of removing ear vertices for H, see Figure 3a). The hole is also a polygon and hole ear vertices can be removed in an analogous manner to the removal of ear vertices of P until only interlayer edges remain.

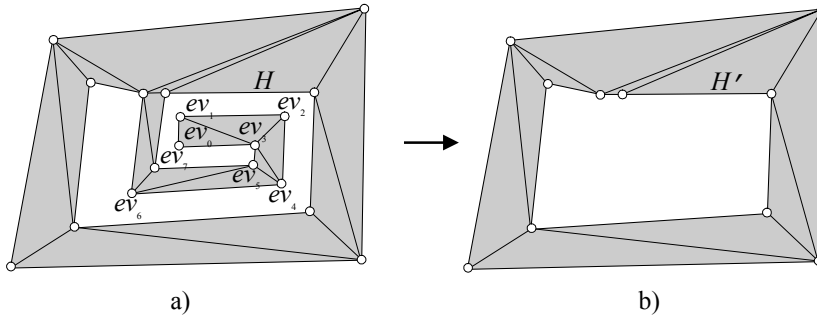


Fig. 3. Triangulations. a) Original, $T(PH)$. b) Ear-reduced hole, $T(PH')$, with stack of ear vertices $[ev_7,ev_6,ev_5,ev_4,ev_3,ev_2,ev_1,ev_0]$ removed.

4 Four Colouring an Ear-Reduced Triangulation

Consider the vertex colourings of the wheel triangulation shown in Figure 4a), where colours are denoted by Red(R), Green(G), Blue(B) and Yellow(Y). The wheel triangulation consists of an odd number of rim vertices coloured using the 3 colours (R,G,B) and a hub vertex coloured using the fourth colour, Y; noting that if the rim consisted of an even number of vertices then a total of three colours would be required, alternate (R,G) pairs and B for the hub. With respect to colouring, vertex i can be viewed as *isolated* since the wheel rim can be coloured using at most 3 colours with the hub vertex coloured independently of the vertex colouring outside of i's rim.

For the ear-reduced triangulation, $T(P'H')$, analogous colour-isolated vertices can be identified on H', see the portion of $T(P'H')$ shown in Figure 4b). An isolated vertex of H' can be viewed as a collapsed spoke for the wheel triangulation, with a single rim vertex coincident with the hub vertex. The isolated set of $\lfloor |V_{h'}|/2 \rfloor$ vertices of H' consists of either the even (v_0,v_2, ...) or odd (v_1,v_3, ...) subsets of $V_{h'}$. The isolated neighbours require a maximum of 3 colours, allowing a fourth colour for the isolated vertex itself.

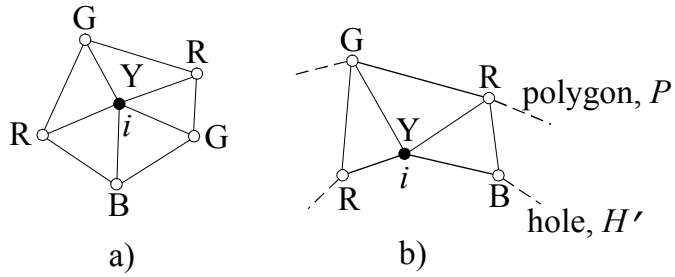


Fig. 4. Triangulation colourings. a) A wheel triangulation. b) Portion of $T(P'H')$.

The colouring of $T(P'H')$ now reduces to the removal of the set of isolated vertices (and their incident edges) of H', reducing the triangulation to one that can be 3-coloured. There are four cases to consider: i) even P'-even H', ii) even P'-odd H', iii)

odd P'-even H' and iv) odd P'-odd H'. There are a further two cases to consider with respect to the number of vertices of P' and H', and are classified as either minimal triangulations when both $|P'|$ and $|H'|$ are equal to 3, otherwise non-minimal. We begin by considering the more general triangulation.

4.1 Non-minimal Triangulations

We will now consider the four cases of even-odd P' and H', an example of each case is shown in Figure 5. The removal of two isolated vertices that share the same vertex of P' leaves a single interlayer edge; eg case iv) of Figure 5a). When H' is even then the only triangles that remain after removal of the isolated vertices are either not adjacent or two adjacent triangles sharing an interlayer edge. When H' is odd three types of triangles can remain after removal of the isolated vertices. The first two types are identical to when H' is even and the third type is a group of three triangles. Each of the possible remaining interlayer edges and triangles are shown in Figure 5.

Case i) is 3-coloured using alternate (R,G) colours for P' and B on the remaining vertices of H'. Case ii) is 3-coloured using alternate (R,G) colours for P' except for vertex v_p. The remaining single triangle or two adjacent triangle vertices are coloured using B. For the three adjacent triangle vertices the non-visible vertex to v_p on H' is coloured B with the remaining vertex on H' coloured G. Case iii) is equivalent to case i) using alternate (R,G) colours for P' and B for v_p and on the remaining single-triangle vertices of H', excluding the vertex adjacent to v_p on H' which is coloured R. Case iv) is analogous to case ii) because H' is even, except that case iv) of Figure 5 shows a single remaining interlayer edge and the other possibility of the group of three-triangles focused at v_p. In this case alternate (R,G) colours are used for P', except for v_p which is coloured B and the remaining vertices of H' are coloured R and G. It is worth mentioning that following the removal of isolated vertices for all four cases of even-odd P' and H', the remaining tessellation can be reduced to a single vertex by repeated removal of vertices of degree less than or equal to three. From an implementation perspective it is easier to continue reducing the remaining tessellations rather than having to specifically cater for cases i) to iv).

Given a 3-colouring of $T(P'H')$ minus the isolated vertices then the isolated vertices can be reinserted, followed by reinserting any PH-$P'H'$ stacks of polygon and hole ear vertices, providing at most a 4-colouring of $T(PH)$.

4.2 Minimal Triangulations

When both $|P'|$ and $|H'|$ are equal to 3 we have one of two topologically unique triangulations, one of which is shown in Figure 1, and the other in Figure 6a). With 6 interlayer arcs and a degree 2 cycle of vertices for H' then the total vertex degree of H' is 12. Since each vertex of H' is of degree greater than or equal to 3 then the only two valid degree permutations for the vertices of H' are (4,4,4) and (3,4,5), with corresponding triangulations shown in Figures 1 and 6a) respectively. The highlighted edges of Figure 6a) illustrate (using Graph Theory terminology) the complete regular-3 subgraph, K_4. The fully connected K_4 component requires 4 colours leading to a 5-

colouring of $T(P'H')$ if only one colour-isolated vertex is removed. As a result, this minimal configuration also requires the removal of a colour-isolated vertex on P'. Two non-visible vertices of H' and P' can always be identified because minimal triangulations have 3 missing interlayer edges. Figure 6b) illustrates a 3-colouring of the remaining 4 vertices following the removal of the isolated vertices and incident edges of P' and H'.

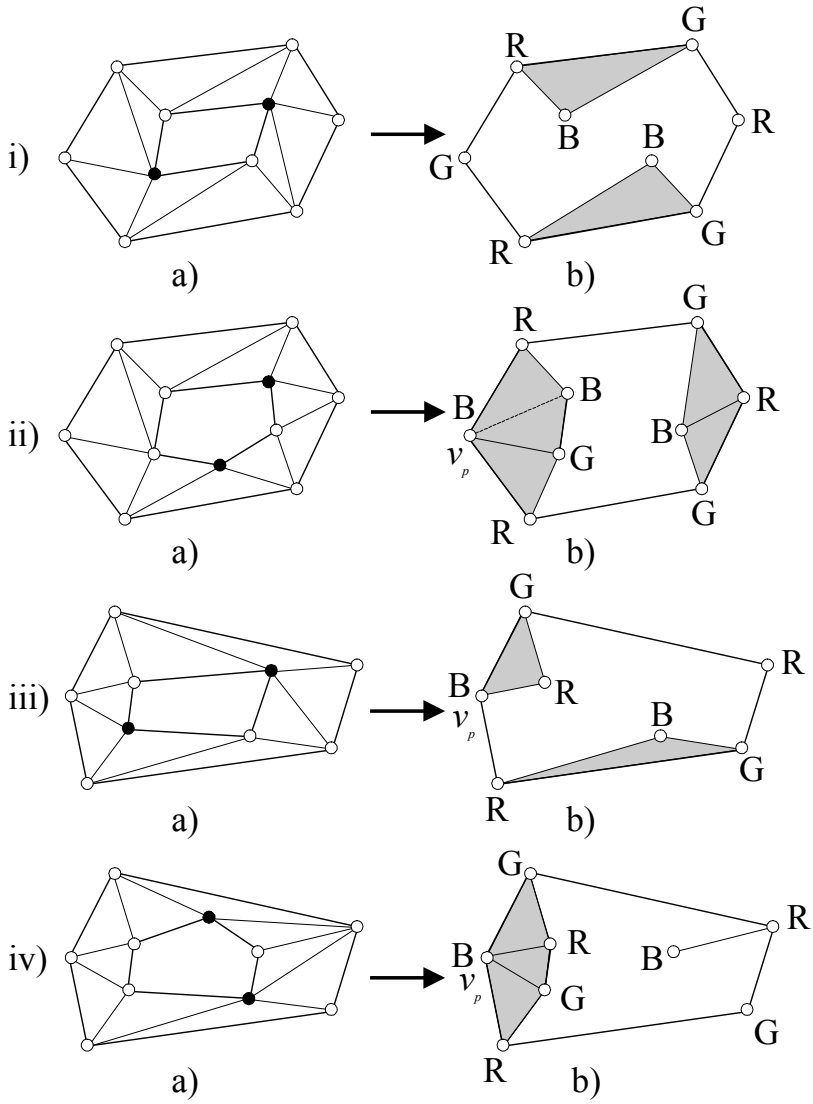


Fig. 5. Even-odd vertex cycles for polygon and hole for non-minimal triangulations. i) even polygon-even hole, ii) even polygon-odd hole, iii) odd polygon-even hole and iv) odd polygon-odd hole. Case a) represents the original $T(P'H')$ and case b) represents a 3-colouring with the removal of colour-isolated vertices which are shown as filled black circles.

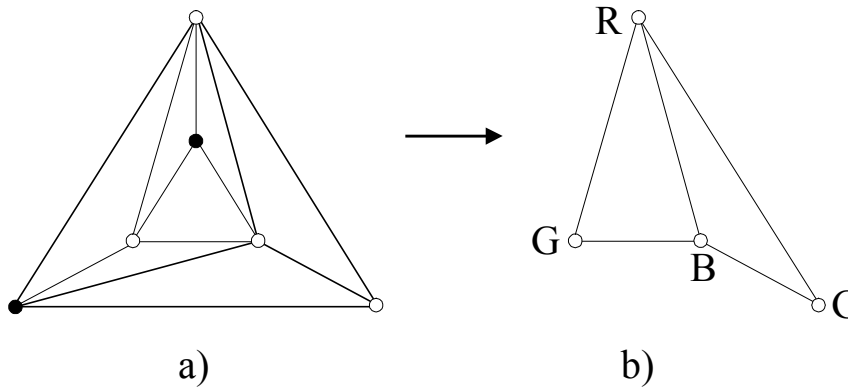


Fig. 6. Minimal triangulation. a) $T(P'H')$ with colour-isolated P' ad H' vertices shown as filled black circles. b) Three-colouring of the remaining 4 vertices following the removal of colour-isolated vertices.

4.3 Efficiency

Let us conclude this section by examining the efficiency of the presented algorithm. In the worst case all polygon and hole vertices would be removed by first removing ear vertices followed by removing colour-isolated vertices. A four-colouring is generated by rebuilding the triangulation by the incremental insertion of the removed vertices. Thus, the efficiency of the method is of the order $O(2(|V_p|+|V_h|))$. Efficiency is assisted by representing the polygon and hole as graphs, that is each vertex stores an adjacency list to its neighbouring polygon or hole vertices and interlayer vertices. Using a dynamic representation of polygon/hole vertices, maintaining linear-time efficiency is further assisted by the use of two data structures. The first is BeforeAfter and is used for removing ear vertices, and encapsulates the vertices before, v_{i-1}, and after, v_{i+1}, a given vertex, v_i, for both the polygon and hole. Index i of v_i is used as the key into a hashtable where the value is the before and after vertices. Derived from BeforeAfter is the second data structure, Isolated which appends a list of vertices of P' representing interlayer edges incident at a vertex of H'.

5 Bipartite and Permutation Graphs

Recognising the equivalence between the vertices and edges of a triangulation and the nodes and arcs of a graph, we observe the equivalence between $T(P'H')$ and a general form of bipartite graphs, [2], and permutation graphs, [3]. As an illustration consider once more the $T(P'H')$ of Figure 2b), redrawn in Figure 7a). Making a cut along interlayer edge (h_0,p_0), followed by straightening the curves of H' and P' we arrive at the topologically equivalent graph, G, of Figure 7b). Nodes of G are partitioned into the two subsets H' and P' such that arcs connect adjacent nodes of H' and P' (h_i is connected to h_{i-1} and h_{i+1} for $0<i<h_{|Vh'|}$ and h_0 connected to $h_{|Vh'|-1}$, and similarly for P') and

connect H' and P'. The 4-colouring of $T(P'H')$ presented in the previous section demonstrates the 4-colouring of the equivalent graph version.

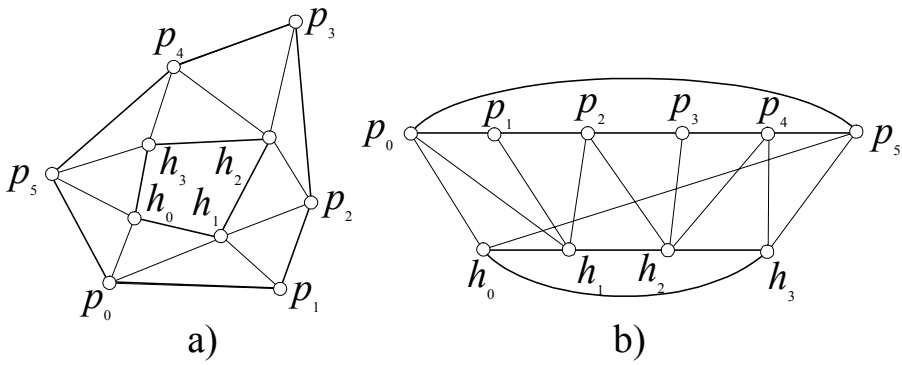


Fig. 7. Equivalence between $T(P'H')$ and general form of bipartite and permutation graphs. a) $T(P'H')$. b) Interlayer cut along (h_0,p_0) and straightening of $T(P'H')$.

6 Conclusion

This paper has shown that the vertices of a triangulation of a polygon containing a hole can be coloured using a maximum of four colours. The algorithm presented is easily implemented and executes in linear-time and consists of firstly reducing a triangulation to a triangulation in which all diagonals are interlayer edges. This is followed by a second reduction phase where alternate colour-isolated vertices are removed until a 3-colouring is prescribed. Finally, the removed vertices are incrementally reinserted using a fourth colour to prevent colour clashes.

This paper has focused entirely on the colouring of a polygon containing a single hole. In general, the method presented cannot be used for colouring polygons containing multiple holes since the reduced triangulation $T(P'H')$ can no longer be extracted. As a result, it is anticipated that for triangulations of polygons containing multiple holes that the general quadratic-time algorithm of [4] is required.

Acknowledgements

The authors would like to acknowledge the financial support of both the Engineering and Physical Sciences Research Council (GR/M92003) and Scottish Higher Education Funding Council (117), as part of the Joint Research Equipment Initiative 1999, for providing the necessary High Performance Computing facilities.

References

1. O'Rourke, J.: Computational Geometry in C, Cambridge University Press. Cambridge (1994)
2. Diestel, R.: Graph Theory. Electronic Edition, Springer-Verlag, New York (2000), http://www.math.uni-hamburg.de/home/diestel/books/graph.theory
3. Chartland, G. and Frechen, J. B.: On the Chromatic Number of Permutation Graphs. In: Harary, F. (ed.): Proof Techniques in Graph Theory, Proceedings of the Second Ann Arbor Graph Theory Conference, February 1968. Academic Press, New York (1969) 21-24
4. Robertson, N., Sanders, D. P., Seymour, P. D. and Thomas, R.: The Four Colour Theorem, J Combin Theory Ser, B70, 2-44 (1997)

Partitioning Polygons into Tree Monotone and Y-monotone Subpolygons

Ralph P. Boland[1] and Jorge Urrutia[2]

[1] University of Waterloo
rpboland@uwaterloo.ca
[2] Instituto de Mathemáticas, Universidad Nacional Autónoma de México
urrutia@matem.unam.mx

Abstract. A polygon Q is tree monotone if, for some highest or lowest point p on Q and for any point q interior to Q, there is a y-monotone curve from p to q whose interior is interior to Q. We show how to partition an n vertex polygon P in $\Theta(n)$ time into tree monotone subpolygons such that any y-monotone curve interior to P intersects at most two of the subpolygons. We then use this partition to further partition P into y-monotone subpolygons such that the number of subpolygons needed to cover any given y-monotone curve interior to P is $O(\log n)$. Our algorithm runs in $\Theta(n)$ time and space which is an improvement by an $O(\log n)$ factor in time and space over the best previous result.

Keywords: tree monotone partition, monotone cover number, circular ray shooting

1 Introduction

The monotone cover number of a decomposition Π of an n vertex simple polygon P into (possibly overlapping) subpolygons, denoted $\mathsf{C}_l(\Pi)$, is the smallest number k such that any y-monotone curve contained in P can be covered by at most k of the subpolygons and their interiors [2]. A Hierarchical Vertical Decomposition [6] of P (modified to use a horizontal visibility map instead of a vertical visibility map [8]) contains a set D of overlapping y-monotone subpolygons of P such that $\mathsf{C}_l(D) = O(\log n)$ and D requires $O(n \log n)$ storage.

We present two algorithms that also construct decompositions of P into y-monotone subpolygons such that the monotone cover number is $O(\log n)$. These algorithms have the advantages that they produce decompositions requiring $\Theta(n)$ storage, they run in $\Theta(n)$ time, and the decompositions of P constructed are partitions; that is the interiors of the subpolygons do not overlap.

We first present a partition of P of interest in its own right. We call a y-monotone curve whose interior is interior to P a *y-curve* of P. We call P *tree monotone* if there is a highest or lowest point p on P such that, for any point q interior to P, there is a y-curve of P from p to q. We call a partition, say Π, of P by chords a *tree monotone partition* of P if the following two conditions hold (opposition is defined in Section 3):

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 903–912, 2003.
© Springer-Verlag Berlin Heidelberg 2003

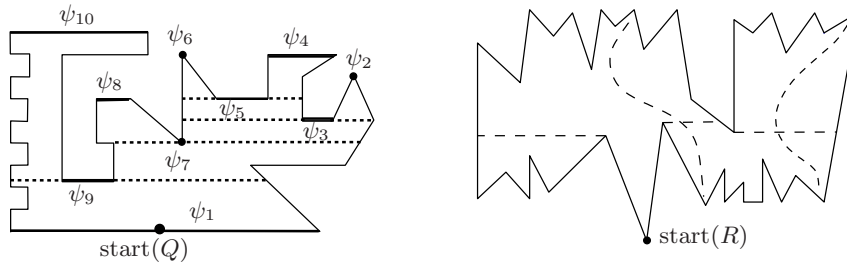


Fig. 1. *Left: A tree +y-monotone polygon Q, its cusps ($\psi_1 - \psi_{10}$), and its lids. Right: A tree monotone partition of a polygon R. Note that the interior of any y-curve of R intersects at most two subpolygons of the partition*

1. The subpolygons of Π are tree monotone.
2. Any pair of coincident subpolygons of Π are in opposition.

The key property of any tree monotone partition Π of P (shown to hold in Section 3) is that $\mathsf{C}_l(\Pi) \leq 2$. This property can sometimes be used to reduce a problem on polygons to the special case of polygons partition-able into two tree *monotone* subpolygons in opposition. This is the approach taken in [3, 2] to develop a solution to the circular ray shooting problem [1].[1] We present a simple algorithm (given the horizontal visibility map of P) that constructs a tree monotone partition of P in $\Theta(n)$ time.

We then present two algorithms that use this tree monotone partition of P to further partition P into y-monotone subpolygons such that the monotone cover number of the partition is $O(\log n)$. The first algorithm constructs a partition of P into y-monotone subpolygons by chords whose monotone cover number is within two of the minimum monotone cover number of any partition of P into y-monotone subpolygons by chords. The second algorithm pays more attention to the sizes of the y-monotone subpolygons of the partition. It produces a partition such that any given y-curve of P can be broken into two halves such that each half is covered by a sequence of $O(\log n)$ subpolygons (and their interiors) of the partition and the number of vertices of the subpolygons in each sequence decreases exponentially. This property of the partition is used in [3,2] to develop a solution to the circular ray shooting problem with the same space and query complexity as the best previous result [6] but which is simpler.[2]

The remainder of this paper is organized as follows. In Section 2 we develop some basic tools that we need. In Section 3 we establish a properties of tree monotone polygons and tree monotone polygon partitions and use them to develop an algorithm that builds a tree monotone partition of P. In Section 4 we use a tree monotone subpolygon partition of P to construct the y-monotone subpolygon partitions of P described in the previous paragraph.

[1] The circular ray shooting problem is to preprocess P so that, given a query directed circular arc ζ, one can quickly find the first intersection of ζ with P.

[2] The preprocessing complexity was also improved by an $O(\log n)$ factor.

2 Some Notation and Tools

Let P be an n vertex simple polygon with vertices $\{v_0, v_1, \ldots, v_{n-1}\}$ in counterclockwise order and edges $\{e_o, \ldots, e_{n-1}\}$ such that v_i and v_{i+1} are the vertices of e_i addition taken modulo n. We will use v_j to denote $v_{(j \bmod n)}$. To simplify the discussion we assume no adjacent edges of P are collinear. We denote the interior of P by interior(P) and denote region(P) = interior(P) $\cup$ P. By partition of P we mean a set of polygons whose interiors do not intersect and whose regions together contain region(P).

A cusp ψ of P is either a vertex v_i of P such that v_{i-1}, and v_{i+1} lie on the same side of the horizontal line through v_i, or a horizontal edge e_i of P such that v_{i-1} and v_{i+2} lie on the same side of the line through e_i. We say that ψ is a *reflex (convex) cusp* if it is either a reflex (convex) vertex or an edge whose endpoints are both reflex (convex) vertices. For any point or segment e, mid(e) denotes e if e is a point and the midpoint of e if e is a segment.

In this paper any subpolygon R of P, including P itself, has a special point we call its *start point* and which we denote by start(R). We assume start(P) = mid(ψ) where ψ is some convex cusp of P. The choice of start(P) has no impact on the results of this paper but it does cause the algorithms presented here to produce slightly different partitions. The chords of P used in this paper are assumed to not intersect start(P). For any chord γ of P we denote by P_γ the subpolygon of P, of the two determined by γ, that does not contain start(P).

Let Q be any simple subpolygon of P such that each edge of Q is the union of one or more chords, edges, or parts of edges of P. Then we call Q a *cut subpolygon* of P. All subpolygons in this paper will be cut subpolygons. When we say that a path is interior to P we mean its interior is contained in interior(P). By the *door* of Q, denoted door(Q), we mean the chord of P on an edge of Q that must be crossed by every path interior to P between any interior point of Q and start(P); if start(P) is a point on Q then the cusp of P containing start(P) is door(Q). By the *windows* of Q, denoted windows(Q), we mean the remaining chords of P that are edges of Q or parts thereof. We assume that start(Q) = mid(door(Q)).

Let Π be a partition of P into cut subpolygons. Then we call Π a *tree partition* of P. This term is appropriate because the graph in which each vertex corresponds to an subpolygon of Π and the edges of the graph correspond to coincident pairs of subpolygons of Π is a tree. All partitions used in this paper are tree partitions. Let R and S be two subpolygons of Π such that door(S) $\in$ windows(R). Then we say that R is the *parent subpolygon* of S and S is a *child subpolygon* of R. For any subpolygon U of Π we denote by parent(U) the parent of U if U has a parent and the value nil otherwise.

We denote by $\Pi_{\boxminus}(P)$ the tree partition of P into trapezoids by the horizontal visibility map of P. To simplify the discussion we assume $\Pi_{\boxminus}(P)$ contains at least two trapezoids. We may abbreviate the term horizontal chord to *h-chord*. Let t be any trapezoid of $\Pi_{\boxminus}(P)$ and p be the midpoint of the h-chord whose endpoints are the midpoints of the non-horizontal edges of t. If some convex cusp ψ of P is a top or bottom edge of t then let q = mid(ψ); otherwise let $q = p$. We denote mid(t) = q. Let $\mathcal{T}$ be the dual tree of $\Pi_{\boxminus}(P)$ such that the root node

of $\mathcal{T}$ corresponds to the subpolygon of $\Pi_{\boxminus}(P)$ that contains start(P) and each edge of $\mathcal{T}$ is directed towards its parent node. We denote by $\Lambda(P)$ the plane embedding of $\mathcal{T}$ such that:

1. For each trapezoid $t \in \Pi_{\boxminus}(P)$, mid(t) is the vertex of $\Lambda(P)$ corresponding to the the dual vertex of t in $\mathcal{T}$.
2. For any two trapezoids t and t_p of $\Pi_{\boxminus}(P)$ such that parent$(t) = t_p$, the chain $C = \text{mid}(t), \text{mid}(\text{door}(t)), \text{mid}(t_p)$ corresponds to the edge of $\mathcal{T}$ joining the vertices of $\mathcal{T}$ corresponding to t and t_p. Chain C is directed upwards if mid(t_p) is above mid(t) and downwards otherwise.
3. Each vertex of $\Lambda(P)$ is colored pink (brown) if its outgoing edge is directed upwards (downwards); the root vertex of $\Lambda(P)$ is given the color shared by its child vertices.

Note that $\Lambda(P)$ intersects each h-chord of P and start(P) is its root vertex; see Fig. 2.

3 Tree Monotone Polygons and Partitions

Many properties of tree monotone polygons are established in [2]. Here we need the following result.

Lemma 1. *P is tree monotone if and only if all vertices of $\Lambda(P)$ have the same color.*

Proof. $\Rightarrow$ Consider otherwise that there are two differently colored and adjacent vertices u_p and u_c of $\Lambda(P)$ such that u_p is the parent of u_c. Let t_p and t_c be the trapezoids of $\Pi_{\boxminus}(P)$ containing u_p and u_c respectively. Then some horizontal line ℓ contains both door(t_p) and door(t_c) since u_p and u_c have different colors. Let C be a y-curve of P from start(P) to u_c; C exists since P is tree monotone. Now C intersects both door(t_p) and door(t_c) so C intersects ℓ twice which is impossible.

$\Leftarrow$ We will show that any point p interior to P and start(P) are the endpoints of some y-curve of P. Let q be a point on $\Lambda(P)$ and $\wp$ be the path on $\Lambda(P)$ from start(P) to q such that some h-chord of P contains $\overline{pq}$ and $\overline{pq} \cap \wp = q$. Then the path $\wp \cup \overline{qp}$ forms a y-curve of P since all the vertices of $\Lambda(P)$ have the same color and thus the edges joining them have the same orientation.

Let ψ be a reflex cusp of P. Then we call the two h-chords of P with an endpoint on ψ the *lids* of ψ. We denote $\mathfrak{L}(P) = \{\gamma : \gamma \text{ is a lid of some reflex cusp of } P\}$. We use $\pi(P)$ to denote the partition of P induced by $\mathfrak{L}(P)$. Clearly, the subpolygons of $\pi(P)$ are y-monotone. Let ξ be the subset of edges of $\Lambda(P)$ whose endpoints are colored differently. We denote by $\mathfrak{L}_r(P)$ the subset of elements of $\mathfrak{L}(P)$ intersected by some element of ξ; note that $|\mathfrak{L}_r(P)| = |\xi|$. We denote by $\pi_r(P)$ be the partition of P induced by $\mathfrak{L}_r(P)$. We now show that $\pi_r(P)$ is a tree monotone partition of P.

Observation 1. *Let Q be a subpolygon of P such that $\Pi_{\boxminus}(Q) \subset \Pi_{\boxminus}(P)$. Then, for any trapezoid t in $\Pi_{\boxminus}(Q)$, vertex mid(t) has the same color in $\Lambda(Q)$ as in $\Lambda(P)$ (but not necessarily the same location).*

If P is tree monotone and start(P) is a lowest (highest) point of P then we call P *tree $+y$-monotone* (*tree $-y$-monotone*). We say that two tree monotone subpolygons of P are *in opposition* if one is $+y$-monotone and the other is $-y$-monotone and the door of one is a window of the other. The following theorem establishes the key property of tree monotone partitions.

Theorem 1. *Let Π be any tree monotone partition of P. Then the interior of any y-curve of P intersects at most two subpolygons of Π.*

Proof. Assume otherwise that the interior of some directed y-curve C of P intersects three different subpolygons Q,R,$T \in \Pi$ in succession. We consider only the case that door(Q) and door(T) are both windows of R. The remaining case, that door(R) is a window of either Q or T, is handled similarly; see [2]. Let $u_Q = \Lambda(P) \cap \text{door}(Q)$ and $u_T = \Lambda(P) \cap \text{door}(T)$. Let $\wp_Q$ be the path on $\Lambda(P)$ from u_Q to start(P) and $\wp_T$ be the path on $\Lambda(P)$ from u_T to start(P). Let v be the point on $\Lambda(P)$ that is the endpoint of $\wp_Q \cap \wp_T$ other than start(P). Then $v = \text{mid}(t)$ where t is some trapezoid of $\Pi_{\boxminus}(P)$. Let $\wp_Q^v$ be the portion of $\wp_Q$ from u_Q to v and $\wp_T^v$ be the portion of $\wp_T$ from u_T to v. By Lemma 1, $\wp_Q^v$ is y-monotone and the edges of $\wp_Q^v$ have the same orientation, either upwards or downwards. Similarly the edges of $\wp_T^v$ have the same orientation. Now Q and T are not in opposition since they are both in opposition with R so the edges of $\wp_Q^v$ and $\wp_T^v$ all have the same orientation. Thus $\wp_Q$ and $\wp_T$ enter t through different windows but from the same top or bottom edge, say e, of t. But now C must also cross both of these windows so C must cross e twice which contradicts that C is y-monotone.

We call a tree monotone cut subpolygon Q of P *maximal* if no tree monotone cut subpolygon of P with the same door as Q properly contains Q.

Lemma 2. *The subpolygons of $\pi_r(P)$ are maximal tree monotone.*

Proof. Consider any subpolygon Q of $\pi_r(P)$. By construction and Observation 1, the vertices of $\Lambda(Q)$ have the same color so Q is tree monotone by Lemma 1. Now assume that there exists a subpolygon Q' of P such that region$(Q) \subset$ region(Q') and door(Q') = door(Q). We will show that Q' is not tree monotone. Let t be a trapezoid of $\Pi_{\boxminus}(Q') - \Pi_{\boxminus}(Q)$ such that door$(t) \in$ windows(Q). Then region$(t) \subseteq$ region(t_P) for some trapezoid t_P of $\Pi_{\boxminus}(P)$. Now mid(t) and mid(t_P) have the same color and, by the construction of $\Lambda(P)$, this color differs from the shared color of the vertices of $\Lambda(Q)$. Therefore Q' is not tree monotone by Lemma 1.

We say that two subpolygons of a tree partition are coincident if one subpolygon is the parent of the other.

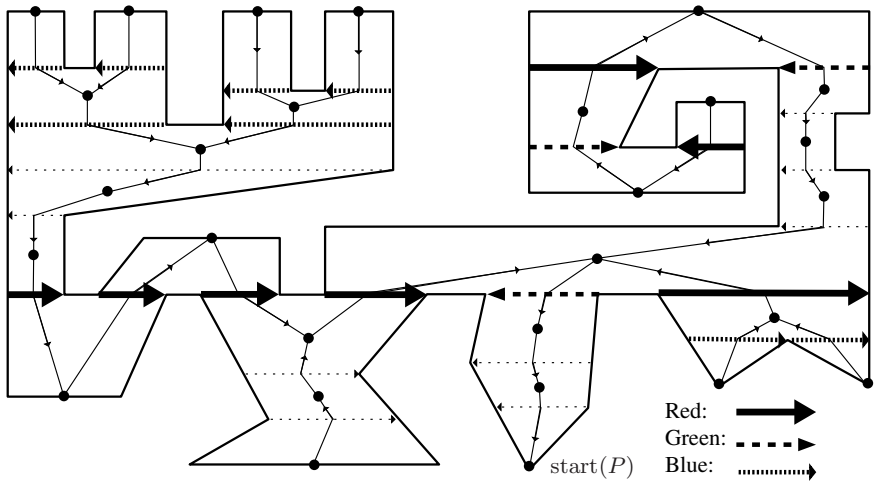


Fig. 2. *Tree $\Lambda(P)$ and the construction of $\mathfrak{L}_r(P)$ and $\pi_r(P)$ using Rule 1*

Lemma 3. *All coincident pairs of subpolygons of $\pi_r(P)$ are in opposition.*

Proof. Assume otherwise that $\pi_r(P)$ has a coincident pair of subpolygons, say Q_1 and Q_2, that are not in opposition. Assume, without loss of generality, that $Q_1 = \text{parent}(Q_2)$ and that both are tree $+y$-monotone. Let Q_3 be the polygon whose interior is the union of the interiors of Q_1, Q_2, and $\text{door}(Q_2)$; note that $\text{door}(Q_3) = \text{door}(Q_1)$. Now Q_3 is tree $+y$-monotone and properly contains Q_1 which is impossible by Lemma 2.

Theorem 2. *$\pi_r(P)$ is a tree monotone partition of P.*

Proof. This follows directly from Lemmas 2 and 3.

The following result is also of interest.

Theorem 3. *Any convex set S contained in region(P) intersects the interior of at most two subpolygons of $\pi_r(P)$.*

Proof. Let T be the subset of the subpolygons of $\pi_r(P)$ that intersect S. Then S contains a directed strictly y-monotone curve C with endpoints p_1 and p_2 such that p_1 and p_2 have the maximum y-coordinate and minimum y-coordinate respectively of any point on the boundary of S, C intersects every polygon in T, and C is a y-curve of P. Now, applying Theorem 1 to C, implies that $|T| \leq 2$.

The following additional results on $\pi_r(P)$ are established in [2].

Lemma 4. *Any curve in interior(P) and decomposable into k interior disjoint y-curves of P intersects the interior of at most $k+1$ of the subpolygons of $\pi_r(P)$.*

Lemma 5. *Any curve in interior*(P) *and on the boundary of a convex set intersects the interior of at most four subpolygons of* $\pi_r(P)$.

For any subpolygon Q of $\pi_r(P)$ with at least one window let family(Q) be the subpolygon of P whose interior is the union of the interiors of Q, the windows of Q, and the child subpolygons of Q. Let $D_{\text{fm}}(P) = \{P\}$ if $|\pi_r(P)| = 1$ and $D_{\text{fm}}(P) = \cup_{Q_i \in \pi_r(P) \text{ and windows}(Q_i) \neq \emptyset}(\text{family}(Q_i))$ otherwise.

Theorem 4. *Any convex set in region*(P) *is contained in one of the elements of* $D_{\text{fm}}(P)$ *and the total number of vertices in all the elements of* $D_{\text{fm}}(P)$ *is* $\Theta(n)$.

Theorem 4 suggests an approach to the well studied problem of finding convex objects in P: search each of the elements of $D_{\text{fm}}(P)$. This will prove useful if either the elements of $D_{\text{fm}}(P)$ are substantially smaller than P or if the properties of the elements of $D_{\text{fm}}(P)$ can be exploited in some way.

We now show that $\pi_r(P)$ can be constructed in $\Theta(n)$ time. Though $\mathfrak{L}_r(P)$ (and thus $\pi_r(P)$) can be found by first constructing $\Lambda(P)$, it is more easily found directly. For any window, say w, of any subpolygon Q of any tree partition Π of P we denote $\text{door}(\Pi, w) = \text{door}(Q)$. We now assign each of the elements of $\mathfrak{L}(P)$ one of the colors red, blue, and green such that the red ones constitute $\mathfrak{L}_r(P)$; see Fig. 2. Each element of $\mathfrak{L}(P)$ is initially colored blue. Next the following rule is applied to each chord $\gamma \in \mathfrak{L}(P)$ in any order:

Rule 1. Let $\sigma = \text{door}(\Pi_{\boxminus}(P), \gamma)$. If γ and σ are collinear then 1) color γ red and 2) color σ green if it is colored blue but not if it is colored red.

Theorem 5. $\mathfrak{L}_r(P)$ *is the subset of the elements of* $\mathfrak{L}(P)$ *that are red.*

Proof. Let u,v, and w be vertices of $\Lambda(P)$ and d and e be edges of $\Lambda(P)$ such that u, d, v, e, w is a path of $\Lambda(P)$ directed towards the root of $\Lambda(P)$. Let t_u, t_v, and t_w be the trapezoids of $\Pi_{\boxminus}(P)$ containing u,v, and w respectively. Then d crosses door(t_u) and e crosses door(t_v). By Rule 1, door(t_u) is colored red if and only if door(t_u) and door(t_v) are collinear. Also, door$(t_u) \in \mathfrak{L}_r(P)$ if and only if one of d and e are directed upwards and the other is directed downwards. But this occurs if and only if door(t_u) and door(t_v) are collinear so door$(t_u) \in \mathfrak{L}_r(P)$ if and only if it is red. Now u, d, v, e, w was an arbitrary path of $\Lambda(P)$ so door(t_u) can be any element of $\mathfrak{L}(P)$ except the elements of S where S is the subset of $\mathfrak{L}(P)$ that cross an edge of $\Lambda(P)$ connected to start(P). But any element λ of S is neither red nor collinear with door$(\pi(P), \lambda) = \text{door}(P)$ so $\mathfrak{L}_r(P)$ is exactly the set of red elements of $\mathfrak{L}(P)$.

Corollary 1. *[2] The blue lids of* $\mathfrak{L}(P)$ *are the lids of the subpolygons of* $\pi_r(P)$ *and the green lids of* $\mathfrak{L}(P)$ *are chords but not lids of the subpolygons of* $\pi_r(P)$.

Now let ψ be any reflex cusp of P. If the lids of ψ are not lids of any other cusp of P then an even simpler rule can be used to color them. Also, if the dual tree of $\Pi_{\boxminus}(P)$ is not available then Rule 1 can be modified so that the door of

each trapezoid can be found using an orientation of the chords of the horizontal visibility map of P. This orientation is shown in Fig. 2. For details of these modifications to Rule 1 see [2].

Theorem 6. *$\mathfrak{L}_r(P)$ and $\pi_r(P)$ can be constructed in $\Theta(n)$ time.*

Proof. In $\Theta(n)$ time $\Pi_{\boxminus}(P)$ can be constructed [4].[3] Then, the red chords of $\mathfrak{L}(P)$ can be found in $\Theta(n)$ time using $\Pi_{\boxminus}(P)$ and Rule 1. But these red chords are just $\mathfrak{L}_r(P)$ by Theorem 5. Finally, $\pi_r(P)$ is easily constructed from $\mathfrak{L}_r(P)$ and $\Pi_{\boxminus}(P)$ in $\Theta(n)$ time.

A variation of our algorithm for constructing $\pi_r(P)$ deals with polygons with holes. It constructs a partition of a polygon with holes into tree monotone subpolygons with holes such that the pockets of the holes are also tree monotone polygons; see [2].

4 *Y*-monotone Partitions

We now present an algorithm for constructing a partition of a tree monotone polygon into y-monotone subpolygons such that the monotone cover number of the partition is minimized. Using this algorithm we can partition a simple polygon into y-monotone subpolygons such that the monotone cover number of the partition is within two of the minimum monotone cover number of any partition of the polygon into y-monotone subpolygons. The proofs of the lemmas in this section are found in [2].

A tree, say T, can be partitioned into paths by deleting all edges on each internal node except the edge to the parent node and the edge of one child node. If, for each internal node, the edge kept is to the child node that is the root of the largest subtree the centroid path decomposition of T [7] is constructed. If, instead, for each internal node, the edge kept is to the child that is the root of the subtree of maximum height then the partition constructed is what we call the *longest path decomposition of* T. In both cases we break ties arbitrarily.

Let Q be an m vertex tree monotone polygon, T_w be the centroid path decomposition of $\Lambda(Q)$, and T_L be its longest path decomposition. We denote by $\pi_w(Q)$ the partition of Q induced by the edges of $\pi(Q)$ crossed by some edge of T_w and by $\pi_L(Q)$ the partition of Q induced by the edges of $\pi(Q)$ crossed by some edge of T_L.

Lemma 6. $\mathbb{C}_{\wr}(\pi_L(Q)) \leq \log m$ *and this value is the minimum monotone cover number of any partition of Q by chords.*

We denote $\pi_L(P) = \cup_{Q_i \in \pi_r(P)} \pi_L(Q_i)$ and $\pi_w(P) = \cup_{Q_i \in \pi_r(P)} \pi_w(Q_i)$; note that $\pi_L(P)$ and $\pi_w(P)$ are partitions of P.

Lemma 7. *Partitions $\pi_L(P)$ and $\pi_w(P)$ can be constructed in $\Theta(n)$ time.*

[3] There are many simpler $O(n \log n)$ time algorithms to choose from; e.g. [5, 8].

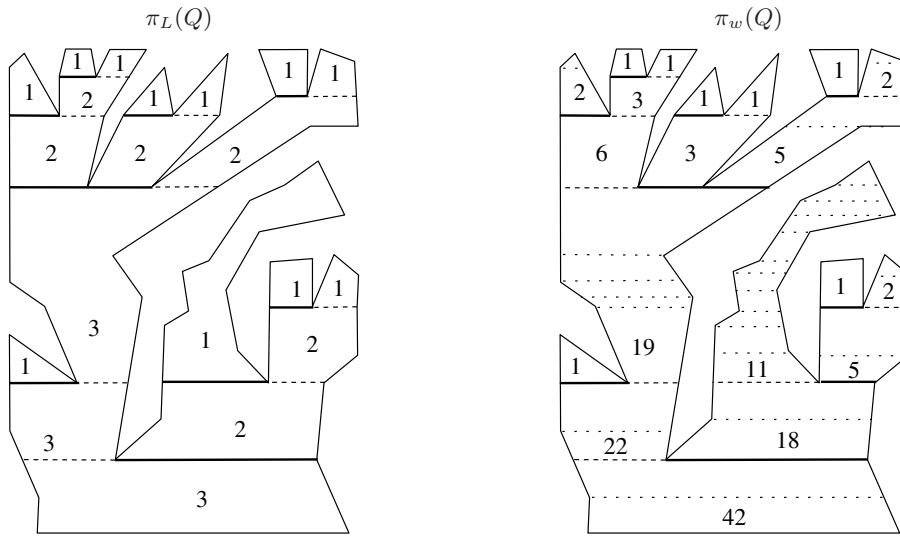


Fig. 3. *Partitions $\pi_L(Q)$ and $\pi_w(Q)$ of an tree +y-monotone polygon Q into y-monotone subpolygons; $\mathsf{C}_l(\pi_L(Q)) = 3$ and $\mathsf{C}_l(\pi_w(Q)) = 4$*

From Theorem 1 and Lemma 6 it is clear that $\mathsf{C}_l(\pi_L(P))$ is within a factor of two of the minimum monotone cover number of any partition of P by chords. However, this bound is actually much tighter.

Lemma 8. $\mathsf{C}_l(\pi_L(P))$ *is at most* $2\log n - 1$ *and within two (one) of the minimum monotone cover number of any partition of P by chords (h-chords).*

While this result is interesting mathematically it ignores the size of the subpolygons. For designing algorithms, partition $\pi_w(P)$ may be a better choice because of the following result. Let Π_U be a tree partition of a polygon U. We say that Π_U has the *telescoping property* if, for every subpolygon R of Π_U such that $\text{parent}(R) \neq \text{nil}$, that $|\Pi_{\boxminus}(\text{parent}(R))| > 2\,|\Pi_{\boxminus}(R)|$.

Lemma 9. $\pi_w(Q)$ *has the telescoping property.*

Note that this lemma implies that $\mathsf{C}_l(\pi_w(Q)) \leq \log m$. More importantly, this property may allow Q to be searched more efficiently in situations where we can assign weights to the trapezoids of a y-monotone polygon so that the trapezoids with large weight are found more quickly by the search algorithm, as for example occurs with weight-balanced trees.

In [2, 3] this property of $\pi_w(Q)$ is used find a solution to the circular ray shooting problem on Q matching the space and query complexity of the best previous result [6] but with a simpler algorithm. We point out that if $\pi_L(Q)$ were used instead of $\pi_w(Q)$, the solution would have a query time of $O(\log^2 m)$. However, if $\Lambda(Q)$ has a height of $O(\log m)$ the cost of a query is $O(\log m)$ even

using $\pi_L(Q)$. This matters because in this case the solution can be made much simpler and also faster by a constant factor if $\pi_L(Q)$ is used.

Finally, using Theorem 1, this solution is easily generalized to apply to simple polygons.

Perhaps this approach can also be used to develop a simpler ray shooting algorithm for polygons. This would require that a sufficiently simple algorithm for ray shooting in y-monotone polygons be found. We are exploring further applications of the results of this paper.

References

1. Pankaj K. Agarwal and Micha Sharir. Circle shooting in a simple polygon. *J. Algorithms*, 14:69–87, 1993.
2. R. P. Boland. *Polygon Visibility Decompositions with Applications.* Ph.D. thesis, March, 2002, University of Ottawa., Ottawa, Ontario, Canada, 2000.
3. R. P. Boland and J. Urrutia. A simpler circular ray shooting algorithm. In *Proc. 13th Canad. Conf. Comput. Geom.*, pages 37–40 (Electronic version 7 pages), 2001.
4. Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.
5. Bernard Chazelle and Janet Incerpi. Triangulation and shape-complexity. *ACM Trans. Graph.*, 3(2):135–152, 1984.
6. Siu-Wing Cheng, Otfried Cheong, Hazel Everett, and René van Oostrum. Hierarchical vertical decompositions, ray shooting, and circular arc queries in simple polygons. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 227–236, June 1999.
7. Richard Cole, Martin Farach-Colton, Ramesh Hariharan, Teresa Przytycka, and Mikkel Thorup. An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *SIAM J. Comput.*, 30(5):1385–1404 (electronic), 2000.
8. A. Fournier and D. Y. Montuno. Triangulating simple polygons and equivalent problems. *ACM Trans. Graph.*, 3(2):153–174, 1984.

Finding Coarse Grained Parallelism in Computational Geometry Algorithms

Volodymyr Beletskyy

Faculty of Computer Science, Technical University of Szczecin, Zolnierska 49 st.,
71-210 Szczecin, Poland
vbeletskyy@wi.ps.pl

Abstract. A technique, permitting automatic finding coarse grained parallelism in algorithms presented with arbitrary nested loops, is presented. The technique is based on finding affine space partition mappings. The main advantage of this technique is that it allows us to form constraints for finding mappings directly in a linear form while known techniques result in building non-linear constraints which should next be linearized. After finding affine space partition mappings, well-known code generation approaches can be applied to expose algorithm parallelism. It is shown how this technique can be applied for parallelizing computational geometry algorithms by means of two examples.

1 Introduction

Many of the applications of computational geometry require real-time speeds. To achieve the required performance, parallel computers and distributed systems can be applied.

Usually, parallel algorithms in computational geometry are developed manually [4], [8], [12]. Such a way requires carrying out research, and consequently high skills, much time, and high costs.

Another way is applying techniques that may automatically find parallelism in sequential algorithms. Most computation in real applications are contained in loops. Hence, most techniques of automatic detecting parallelism deal with loops.

In this paper, we consider automatic finding paallelism for a coarse grained multicomputer, which comprised of a set of p processors with local memory per processor and an arbitrary communication network (or shared memory).

Coarse-grained parallelism is employed by creating a thread on each of the processors, executing in parallel for a period of time with occasional synchronization.

This paper presents a technique permitting us to find communication-free parallelism for arbitrary nested loops and next illustrates how it can be applied for detecting coarse grained parallelism in computational geometry algorithms. A basic idea of the technique assumes that the operations of a loop are divided into partitions, such that dependent operations are placed in the same partition. A partitioning is described by an affine mapping for each loop instruction.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 913-922, 2003.
© Springer-Verlag Berlin Heidelberg 2003

An *m*-dimensional affine partition mapping for statement *s* in a loop is an *m*-dimensional affine expression $\overline{\Phi}_s = C_s \bar{i} + \bar{c}_s$, which maps an instance of statement *s*, indexed by its iteration vector $\bar{i}$, to an *m*-dimensional vector. After finding affine mappings, well-known techniques for generating parallel code can be applied, for example, [3],[5],[19].

2 Backgrounds

To detect parallelism in loops, we need to know dependences among the loop iterations.

We refer to a particular execution of a statement for a certain iteration of the loops, that surround this statement, as an operation.

Two operations are dependent if both access the same memory location and if at least one access is a write.

There are three types of data dependence: flow, anti, and output dependence. A statement *S2* is flow dependent on statement *S1* if a variable is first defined in statement *S1* and later used in statement *S2*. A statement *S2* is anti dependent on statement *S1* if a variable is first used in statement *S1* and later defined in statement *S2*. A statement *S2* is output dependent on *S1* if a variable is first defined in statement *S1* and later redefined in statement *S2*.

Our algorithm requires an exact representation of the dependences, and consequently an exact dependence analysis.

A dependence analysis is exact if for any affine dependence it detects a dependence if and only if one exists[18].

Our algorithm is based on the dependence analysis proposed by Pugh and Wonnacott [17]. That analysis permits us to extract exact dependence information for any single structured procedure in which the expressions in the subscripts, loop bounds, and conditionals are affine functions of the loop indices and loop-independent variables, and the loop steps are known constants. Dependences are presented with dependence relations. A dependence relation is a mapping from one iteration space to another, and is represented by a set of linear constraints on variables that stand for the values of the loop indices at the source and destination of the dependence and the values of the symbolic constants. A dependence relation is a tuple relation. An integer *k*-tuple is a point in Z^k. A tuple relation is a mapping from tuples to tuples.

The basic merits of the dependence analysis proposed by Pugh and Wonnacott are as follows: i) it is exact; ii) it is valid for both perfectly and imperfectly nested loops; iii) it permits value-based dependences to be calculated.

A dependence between iterations $\overline{I}$ and $\overline{J}$, which are the dependence source and destination, respectively, is value-based if: $\overline{I}$ is executed before $\overline{J}$; $\overline{I}$ and $\overline{J}$ refer to a memory location *M*, and at least one of these references is a write; the memory location *M* is not written between iteration $\overline{I}$ and iteration $\overline{J}$.

The dependence analysis by Pugh and Wonnacott is implemented in Petit, a research tool for doing dependence analysis and program transformations. To carry out dependence analysis manually, the Omega calculator can be applied [14].

3 Finding Communication-free Parallelism

Our technique finds communication-free parallelism in algorithms presented with arbitrary nested loops that meet the requirements of the dependence analysis by Pugh and Wonnacott.

In this section, we consider the following task. Given a loop originating a set of dependences presented with the dependence relations

$$D = \{\bar{I}_j -> \bar{J}_j, j = 1,2,..., q\},$$

find m_s-dimensional affine space partition mappings $\bar{\Phi}_s = C_s \bar{i} + \bar{c}_s$ for each statement *s* such that $\bar{\Phi}_{si}(\bar{I}_j) = \bar{\Phi}_{sk}(\bar{J}_j) = \bar{P}_m$, where *si, sk* are the statements which instances originate the source and destination of the dependence $\bar{I}_j -> \bar{J}_j$, C_s is a matrix of dimensions m_s x *n*, $\bar{c}_s$ is an m_s-dimensional vector representing a constant term, $\bar{P}_m$ is a vector representing the identifier of a processor to execute the sources and destinations arisen by the relation dependence $\bar{I}_j -> \bar{J}_j$.

Given the dependence relations, we should firstly form the reduced dependence graph[7], that is, a directed graph *G=(V, E),* where the vertices are the statements and the edges represent the dependence relations.

Definition 3.1 Strongly connected graph is a directed graph that has a path from each vertex to every other vertex.

Definition 3.2 Given a directed graph *G=(V, E)*, a subgraph *S=(V', E')* is a strongly connected component(*SCC*) if *S* is strongly connected, and for all vertices *u* such that $u \in V$ and $u \notin V'$ there is no vertex $v \in V'$ for which $(u, v) \in E$.

The graph in Figure 1 below has two strongly connected components given by {S1, S2} and {S3}, respectively.

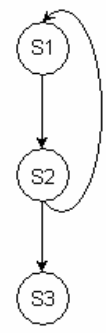


Figure 1. A directed graph

Let $\overline{I}_j$,$\overline{J}_j$ be represented in the following form

$$\overline{I}_j = A_{1j} * \overline{i1}_j + \overline{B1}_j, \quad \overline{J}_j = A_{2j} * \overline{i2}_j + \overline{B2}_j,$$

where $\overline{I}_j$,$\overline{J}_j$ are $m1_j$ and $m2_j$-dimensional vectors, respectively, $m1_j <= n$, $m2_j <= n$, n is the number of the loop nests, $\overline{B1}_j$, $\overline{B2}_j$ are $m1_j$ and $m2_j$-dimensional vectors, respectively, $\overline{i1}_j$ and $\overline{i2}_j$ are n-dimensional vectors, A_{1j} and A_{2j} are matrices of dimensions $m1_j$ x n and $m2_j$ x n, respectively.

Let us write matrices A_{1j} and A_{2j} in the following form

$$A_{1j} = [\overline{A}^1_{1j} \ \overline{A}^2_{1j} \ \ldots \ \overline{A}^n_{1j}], \quad A_{2j} = [\overline{A}^1_{2j} \ \overline{A}^2_{2j} \ \ldots \ \overline{A}^n_{2j}],$$

where $\overline{A}^i_{1j}$ and $\overline{A}^i_{2j}$, $i=1,2,\ldots,n$ represent the columns of A_{1j} and A_{2j}, respectively.

If a dependence $\overline{I}_j -> \overline{J}_j, j \in [1,q]$ is the self dependence, that is, it is originated with the same statement s, we seek an affine space partition mapping $\overline{\Phi}_s = C_s \overline{i} + \overline{c}_s$ such that the following condition is satisfied

$$C_s \overline{I}_j + \overline{c}_s = C_s \overline{J}_j + \overline{c}_s$$

or

$$C_s \overline{I}_j - C_s \overline{J}_j = 0,$$

which means that the same processor executes operations $\overline{I}_j$ and $\overline{J}_j$.

Let us rewrite the equation above as follows

$$C_s(\overline{A}^1_{1j} i1^1_j + \overline{A}^2_{1j} i1^2_j + \ldots + \overline{A}^n_{1j} i1^n_j + \overline{B1}_j) -$$

$$- C_s(\overline{A}^1_{2j} i2^1_j + \overline{A}^2_{2j} i2^2_j + \ldots + \overline{A}^n_{2j} i2^n_j + \overline{B2}_j) = 0,$$

where $i1^1_j, i1^2_j, \ldots, i1^n_j$ and $i2^1_j, i2^2_j, \ldots, i2^n_j$ are the coordinates of $\overline{i1}_j$ and $\overline{i2}_j$, respectively, and transform it to the form

$$\langle \overline{C}_s, \overline{A}^1_{1j} \rangle i1^1_j + \langle \overline{C}_s, \overline{A}^2_{1j} \rangle i1^2_j + \ldots + \langle \overline{C}_s, \overline{A}^n_{1j} \rangle i1^n_j + \langle \overline{C}_s, \overline{B1}_j \rangle - \quad (1)$$

$$- \langle \overline{C}_s, \overline{A}^1_{2j} \rangle i2^1_j - \langle \overline{C}_s, \overline{A}^2_{2j} \rangle i2^2_j - \ldots - \langle \overline{C}_s, \overline{A}^n_{2j} \rangle i2^n_j - \langle \overline{C}_s, \overline{B2}_j \rangle = 0,$$

where $\langle \overline{x}, \overline{y} \rangle$ denotes the inner product of vectors $\overline{x}$ and $\overline{y}$, $\overline{C}_s$ represents an arbitrary row of C_s.

If a dependence $\overline{I}_j -> \overline{J}_j, j \in [1,q]$ is originated with two different statements $s1$ and $s2$, we seek affine space partition mappings $\overline{\Phi}_{s1} = \overline{C}_{s1} \overline{i} + \overline{c}_{s1}$ and $\overline{\Phi}_{s2} = \overline{C}_{s2} \overline{i} + \overline{c}_{s2}$ such that the following condition is satisfied

$$C_{s1} \overline{I}_j + \overline{c}_{s1} = C_{s2} \overline{J}_j + \overline{c}_{s2}.$$

Let us rewrite the above condition as follows

$$C_{s1}(\overline{A}^1_{1j} i1^1_j + \overline{A}^2_{1j} i1^2_j + ... + \overline{A}^n_{1j} i1^n_j + \overline{B1}_j) + \overline{c}_{s1} =$$

$$C_{s2}(\overline{A}^1_{2j} i2^1_j + \overline{A}^2_{2j} i2^2_j + ... + \overline{A}^n_{2j} i2^n_j + \overline{B2}_j) + \overline{c}_{s2},$$

and transform it to the form

$$\langle \overline{C}_{s1}, \overline{A}^1_{1j} \rangle i1^1_j + \langle \overline{C}_{s1}, \overline{A}^2_{1j} \rangle i1^2_j + ... + \langle \overline{C}_{s1}, \overline{A}^n_{1j} \rangle i1^n_j + \langle \overline{C}_{s1}, \overline{B1}_j \rangle + c_{s1} -$$

$$- \langle \overline{C}_{s2}, \overline{A}^1_{2j} \rangle i2^1_j - \langle \overline{C}_{s2}, \overline{A}^2_{2j} \rangle i2^2_j - ... - \langle \overline{C}_{s2}, \overline{A}^n_{2j} \rangle i2^n_j - \langle \overline{C}_{s2}, \overline{B2}_j \rangle - c_{s2} = 0, \quad (2)$$

where $\overline{C}_{s1}, \overline{C}_{s2}$ represent an arbitrary row of C_{s1}, C_{s2}, respectively, c_{s1}, c_{s2} are unknown constant terms which are dependent on $\overline{C}_{s1}, \overline{C}_{s2}$.

Let us introduce an r_j-dimensional vector $\bar{i}_j$ with coordinates $i^1_j, i^2_j, ..., i^{r_j}_j$ which contains all unique coordinates(having different names) of $\overline{I}_j, \overline{J}_j$, $r_j <= m1_j + m2_j$ and rewrite equations (1) and (2) in the following form

$$\sum_{k=1}^{r_j} D^k_j \; i^k_j + d_j = 0, \quad (3)$$

where D^k_j and d_j are formed as follows:

i) for self dependences, D^k_j is the sum of all those $\langle \overline{C}_{s1}, \overline{A}^m_{1j} \rangle$ and $-\langle \overline{C}_{s1}, \overline{A}^p_{2j} \rangle$ for which the following condition holds $i1^m_j = i2^p_j = i^k_j$; $d_j = \langle \overline{C}_{s1}, \overline{B1}_j \rangle - \langle \overline{C}_{s1}, \overline{B2}_j \rangle$;

ii) for dependences originated with two different statements, D^k_j is the sum of all those $\langle \overline{C}_{s1}, \overline{A}^m_{1j} \rangle$ and $-\langle \overline{C}_{s2}, \overline{A}^p_{2j} \rangle$ for which the following condition holds $i1^m_j = i2^p_j = i^k_j$; $d_j = \langle \overline{C}_{s1}, \overline{B1}_j \rangle - \langle \overline{C}_{s2}, \overline{B2}_j \rangle + c_{s1} - c_{s2}$.

Algorithm 1. Find coarse grained parallelism in an algorithm presented with a loop.

1. Form a reduced dependence graph for a loop and find all *SCCs* in this graph.
2. For each *SCC*, form a set of the dependence relations. Each such a set includes only those dependence relations which define edges in a given *SCC*. Let us the number of the dependence relations of SSC_t equal to q_t.
3. For each *SCC* do
   3.1 From each dependence $\overline{I}_j -> \overline{J}_j$, $j = 1,2,..., q_t$, build the constraint in the form of (3).

3.2 Construct a system of linear equations of the form

$$D_j^k = 0,$$

$$d_j = 0, j = 1,2,...,q_t, \quad k = 1,2,...,r_j$$

which we rewrite as

$$A\bar{x} = 0,$$

where $\bar{x}$ is a vector representing all the unknown coordinates of $\bar{C}_s$ and constant terms c_s of the affine space partition mappings.

3.3 Eliminate all the unknowns c_s from $A\bar{x} = 0$ with the Gaussian Elimination algorithm. Let the reduced system be $A'\bar{x}' = 0$, where $\bar{x}'$ represents the unknown coordinates of $\bar{C}_s$.

3.4 Find the solution to $A'\bar{x}' = 0$ as a set of basis vectors spanning the null space of A'.

3.5 Find one row of the desired affine partition mapping from each basic vector found in step 4. The coordinates of $\bar{C}_s$ are formed directly by the basic vector; the constant terms c_s are found from the coordinates of $\bar{C}_s$ using $A\bar{x} = 0$.

3.6 For each *SCC,* using mappings $\bar{\Phi}_s = C_s\bar{i} + \bar{c}_s$, apply one of well-known techniques [3],[5],[19] for generating parallel code.

4. Generate code to execute the *SCCs* in topological order.

The algorithm described finds communication-free parallelism for each *SCC* in the reduced dependence graph of a loop.

4 Examples

Let us illustrate the technique presented by means of the two following examples from computational geometry.

Example 1. /* The loop from the code generating a comb polygon. This code is associated with "Computational Geometry in C" (Second Edition) by Joseph O'Rourke, Cambridge University Press, 1998 */

```
for ( i = 0; i < n/2; i++ ) {

      s1: P[2*i][X] = n-2 - (2*i);

      s2: P[2*i][Y] = 0;

      s3: P[2*i+1][X] = n-2 - (2*i + 1);

      s4: P[2*i+1][Y] = 10;

   }
```

This loop originates the following dependences found with Petit

output s1: P(2*i,X) → s2: P(2*i,Y)
{[i] -> [i] : X = Y && 0 <= i <n/2}
output s3: P(2*i+1,X) → s4: P(2*i+1,Y)
{[i] -> [i] : X = Y && 0 <= i < n/2}.

Analyzing these dependences, we can conclude that there exist the four *SCCs*, represented with *s1, s2, s3,* and *s4*, respectively, and there are no self-dependences in this loop. This means that, according to Algorithm 1, we can form the following code

```
parfor (p = 0; p < n/2; p++)
        s1: P[2*p][X] = n-2 - (2*p);
parfor (p = 0; p < n/2; p++)
        s2: P[2*p][Y] = 0;
parfor (p = 0; p < n/2; p++)
        s3: P[2*p+1][X] = n-2 - (2*p + 1);
parfor (p = 0; p < n/2; p++)
        s4: P[2*p+1][Y] = 10;
```

where `parfor` means that all loop iterations can be executed simultaneously, *p* is the identifier of a processor.

Example 2. /* This loop is extracted from the code fg.c : face graph of hull, and splay trees by Ken Clarkson, http:// cm.bell-labs.com/who/clarkson */

```
for (j=0;j<100;j++)
      for (i=0;i<20;i++) {
          s1:tot_good[i]+= fg_hist[i][j];
          s2:tot_bad[i]+= fg_hist_bad[i][j];
          s3:tot_far[i]+= fg_hist_far[i][j];
          }
```

This loop originates the following dependences found with Petit

anti s1: tot_good(i) → s1: tot_good(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
flow s1: tot_good(i) → s1: tot_good(i)
{[j,i] -> [j+1,i] : 0 <= j <= 98 && 0 <= i <= 19}
flow s1: tot_good(i) → s1: tot_good(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
output s1: tot_good(i) →s1: tot_good(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
anti s2: tot_bad(i) →s2: tot_bad(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
flow s2: tot_bad(i) →s2: tot_bad(i)
{[j,i] -> [j+1,i] : 0 <= j <= 98 && 0 <= i <= 19}
flow s2: tot_bad(i) → s2: tot_bad(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}

output s2: tot_bad(i) →s2: tot_bad(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
anti s3: tot_far(i) → s3: tot_far(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
flow s3: tot_far(i) → s3: tot_far(i)
{[j,i] -> [j+1,i] : 0 <= j <= 98 && 0 <= i <= 19}
flow s3: tot_far(i) → s3: tot_far(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}
output s3: tot_far(i) → s3: tot_far(i)
{[j,i] -> [j',i] : 0 <= j < j' <= 99 && 0 <= i <= 19}.

Analyzing these dependences, we conclude that there exist the following three *SCCs,* represented with statements *s1, s2,* and *s3*, respectively. There exist the self-dependences originated with each statement in the loop, that is, we cannot execute all loop iterations in parallel.

We seek space partition mappings for *s1, s2,* and *s3* of the forms $\Phi_{s1}=[C_{11} \quad C_{12} \quad]\bar{i}$, $\Phi_{s2}=[C_{21} \quad C_{22} \quad]\bar{i}$, and $\Phi_{s3}=[C_{31} \quad C_{32} \quad]\bar{i}$, respectively. According to our approach, we first form the following constraint for each *SCC*

$$C_{k1}*j'+C_{k2}*i = C_{k1}*j+C_{k2}*i \text{ , } k=1,2,3$$

$$C_{k1}*(j+1)+C_{k2}*i = C_{k1}*j+C_{k2}*i \text{ , } k=1,2,3$$

which we simplify to the following form

$$C_{k1}*(j'-j)=0, \quad k=1,2,3$$
$$C_{k1}=0 \text{ , } \quad k=1,2,3.$$

The resulting constraint is as follows

$$C_{k1}=0, \quad k=1,2,3 \text{ .}$$

The solution to this constraint is

$$C_{k1}=0, C_{k2}=1, \quad k=1,2,3 \text{ .}$$

Applying the Omega code generator (public available at ftp://ftp.cs.umd.edu/pub/omega) for the transformation of the source loop by means of the space partition mappings found, we have got the following parallel code

```
parfor (p=0;p<20;p++)
   for (j=0;j<100;j++)
        s1:tot_good[p]+= fg_hist[p][j];
parfor (p=0;p<20;p++)
   for (j=0;j<100;j++)
        s2:tot_bad[p]+= fg_hist_bad[p][j];
parfor (p=0;p<20;p++)
   for (j=0;j<100;j++)
        s3:tot_far[p]+= fg_hist_far[p][j];
```

where `for` and `parfor` denote serial and parallel loops, respectively. The outer loop gives space partitionings while the inner loop defines the statement instances executed serially by processor p.

5 Related Work

Unimodular loop transformations[2],[20], permitting the outer loop to be parallelized, find a communication-free partitions. But unimodular transformations do not allow such transformations as loop fission, fusion, scaling, reindexing, or reordering grasped with affine transformations.

Techniques presented in [1],[13] enable finding communication-free partitioning, supposing operations within each lop iteration are indivisible. This is the limitation of those techniques.

The affine partitioning framework, considered in many papers, for example, [6],[7],[9],[10],[11],[15],[16],[17], unifies a large number of previously proposed loop transformations. It is the most powerful framework for loop parallelization today allowing us to parallelize loops with arbitrary nested loops.

Work [17] is most closely related to ours. In contrast to that work and other known approaches, our technique permit us to form constraints for finding affine space partition mappings for arbitrary nested loops directly in a linear form without the necessity of applying the Farkas lemma[17] to linearize the constraint, and, hence, it is less time-consuming than that of work [17] and other known approaches.

6 Conclusion

This paper describes a technique to detect coarse grained parallelism in algorithms presented with arbitrary nested loops which comprise most computations in many computational geometry applications. This technique can be easily implemented by means of academic public available software, for example, Omega project (ftp://ftp.cs.umd.edu/pub/omega) which includes all necessary tools: finding the dependence relations, solving linear systems of equalities, and code generation.

Other public available software (see, for example http: //www.irisa.fr/cosi/ ALPHA/welcome.html) can be also used to generate parallel code in accordance with our approach.

Applying the technique presented may permit us to shorten the time of building parallel algorithms as well as to transform existing sequential computational geometry applications into parallel ones.

References

[1] Amarasinghe, S.P., Lam, M.S.: Communication optimization and code generation for distributed memory machines. In: Proceedings of the SIGPLAN'93. (1993) 126-138

[2] Banerjee, U.: Unimodular transformations of double loops. In: Proceedings of the Third Workshop on Languages and Compilers for Parallel Computing. (1990) 192-219
[3] Boulet, P., Darte, A., Silber, G.A., Vivien, F.: Loop paralellelization algorithms: from parallelism extraction to code generation. Technical report. (1997)
[4] Chan, A., Dehne, F., and Rau-Chaplin, A.: Coarse grained parallel next element search. In.: Proceedings of the 11-th International Parallel Processing Symposium. (1997)320-325
[5] Collard, J.F., Feautrier, P., Risset, T.: Construction of do loops from systems of afffne constraints. Technical Report 93-s1, LIP, Lyon. (1993)
[6] Darte, A., Risset, T., Robert, Y.: Loop nest scheduling and transformations. In Dongarra, J., Tourancheau, B., eds.: Enviroments and tools for parallel science computing. North Holland. (1993)
[7] Darte, A., Silber, G., Vivien, F.: Combining retiming and scheduling techniques for loop parallelization and loop tiling. Technical Report 96-34, Laboratoire de l'Informatique du Parallelisme. (1996)
[8] Dehne,F., Fabri., A.: Scalable parallel computational geometry for coarse grained multicomputers. International Journal of Computational Geometry and Applications. 3(1996) 379-400
[9] Feautrier, P.: Some efficient solutions to the affine scheduling problem, part i, one dimensional time. International Journal of Parallel Programming 21 (1992) 313-348
[10] Feautrier, P.: Some efficient solutions to the affine scheduling problem, part ii, multidimensional time. International Journal of Parallel Programming 21 (1992) 389-420
[11] Feautrier, P.: Toward automatic distribution. Journal of Parallel Processing Letters 4 (1994) 233-244
[12] Goodrich, M.T.: Parallel algorithms in geometry. In: J. E. Goodman and J. O'Rourke, editors, Handbook of Discrete and Computational Geometry, chapter 36, CRC Press LLC. (1997)669-682
[13] Huang, C., Sadayappan, P.: Communication-free hyperplane partitioning of nested loops. Journal of Parallel and Distributed Computing 19 (1993) 90-102
[14] Kelly, W., Maslov, V., Pugh, W., Rosser, E., Shpeisman, T., Wonnacott, D.: The omega library interface guide. Technical Report CS-TR-3445, University of Maryland. (1995)
[15] Lim, W., Lam, M.S.: Communication-free parallelization via affine transformations. In: Proceedings of the Seventh Workshop on Languages and Compilers for Parallel Computing. (1994) 92-106
[16] Lim, W., Cheong, G.I., Lam, M.S.: An affine partitioning algorithm to maximize parallelism and minimize communication. In: Proceedings of the 13th ACM SIGARCH International Conference on Supercomputing. (1999)
[17] Lim, W., Lam, M.S.: Maximizing parallelism and minimizing synchronization with affine transforms. In: Conference Record of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. (1997)
[18] Pugh, W., D.Wonnacott: An exact method for analysis of value-based array data dependences. In: Workshop on Languages and Compilers for Parallel Computing. (1993)
[19] Quillere, F., Rajopadhye, S., Wilde, D.: Generation of efficient nested loops from polyhedra. International Journal of Parallel Programming 28 (2000)
[20] Wolf, M.E.: Improving locality and parallelism in nested loops. Ph.D. Dissertation CSL-TR-92-538, Stanford University, Dept. Computer Science. (1992)

On the Reliability of Triangle Intersection in 3D

Steven Robbins and Sue Whitesides

School of Computer Science, McGill University
3480 University Street, Montréal, QC, H3A 2A7, Canada.
{stever,sue}@cs.mcgill.ca

Abstract. Implementors of triangle/triangle intersection tests often opt to forego exact calculations for speed reasons. It is widely known that such code will fail for certain inputs, but it is not evident from the literature that published intersection tests implemented using floating-point arithmetic are not stable. We show how such a test can fail on a triangle pair that is widely separated in space. We find that an exact intersection test can be implemented with a modest speed penalty.

1 Introduction

Surfaces are often modeled using polyhedra composed of many triangles. When two such surfaces are checked for intersection, a preprocessing step using bounding volumes (arranged hierarchically [7] or not [19]) is used to quickly eliminate many triangle pairs from further consideration. For the pairs that remain, a fast method of checking for intersection is required. In geometric computing, speed comes at the cost of reliability.

Reliability of computational geometry algorithms has been studied for many years, see e.g. [18] for an overview. Algorithms are classified as *exact* if the result is always correct for the input given, *robust* if the result is always correct for some perturbation of the input, and *non-robust* otherwise. A robust algorithm for which the perturbation is bounded is called *stable*. Most work on robustness has examined classical computational geometry algorithms such as computing the Delaunay triangulation or the arrangement of a set of lines, for which the combinatorial output must satisfy nontrivial consistency properties. Triangle intersection, having a single bit of output, is trivially robust. The interesting question is whether an algorithm is stable or exact.

Exact computation can be achieved by using extended precision arithmetic to compute each quantity used for a branching test in the algorithm [17,11]. Extended precision arithmetic comes at a cost in execution time. In some contexts, e.g. graphics or virtual reality systems, speed is paramount and an inexact, but stable, triangle intersection test is perfectly acceptable. We show, however, that published triangle intersection codes of Held [8] and of of Möller [13] are not stable. Such code can be made stable using a general arbitrary precision arithmetic package such as LEDA's reals [4] or the CORE library [9], but the resulting speed penalty is prohibitive for many applications. Instead, we use the orientation tests of Shewchuk [16] to implement an exact triangle intersection test, with a modest penalty in speed.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 923–930, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Inexact Intersections

It is instructive to first examine testing for line segment intersection in the plane.

2.1 Segment Intersection

It is well known (e.g. O'Rourke [14]) that this problem can be phrased in terms of the 2D orientation predicate, which is given by the sign of a degree-2 polynomial in the coordinate values of three points. The predicate value indicates on which side of the directed line through two points lies the third point. An implementation of this predicate in floating point arithmetic can fail when the value is very small, which corresponds to the three points being nearly collinear. Two non-intersecting segments are shown in the left panel of Figure 1. If the orientation test for point C with respect to line AB fails, the segment intersection test incorrectly reports that the segments do intersect. In effect, the intersection algorithm behaves as if it were given the perturbed input pictured on the right panel of Figure 1.

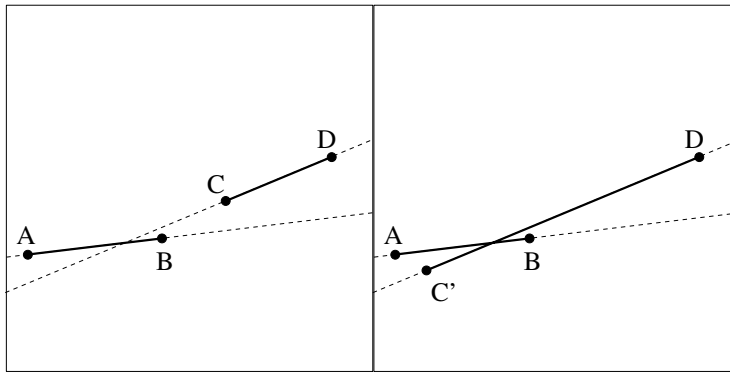


Fig. 1. Problematic case for segment intersection. Point C lies close enough to the line through AB that the floating point computation of the orientation predicate fails, whereas the orientation for point D is correctly computed. The test result is as if the input on the right were given instead. Note that the perturbation from C to C' may be arbitrarily large.

This is a particularly egregious error, since the segments may be arbitrarily far apart. Note also that this problem is exacerbated by the common practice of using an "epsilon" for numeric comparisons, i.e. a floating point number x is deemed to be equal to floating point value y when $|x - y| < \epsilon$. The segments in this example will then be deemed intersecting if the orientation value $< \epsilon$ even when the approximate computation correctly returns a positive number.

Several algorithms for computing the sign of the orientation predicate exactly using existing floating point hardware have been implemented and used for segment intersection testing [15,16,6].

2.2 Triangle Intersection

Testing whether two 3D triangles intersect can be reduced to computing the 3D orientation predicate, given as the sign of a degree-3 polynomial in the coordinate values of four points [1,14]. The predicate value indicates on which side of the oriented plane determined by three points lies the fourth point.

Let the input triangles be T_1 and T_2, each specified using coordinates of its three vertices. Let the planes supporting the triangles be denoted by π_1 and π_2, respectively. Testing whether triangle T_2 intersects π_1 is a matter of testing each vertex of T_2 against π_1 using the orientation predicate. If all vertices lie strictly to one side of the plane, T_2 does not intersect π_1; otherwise they do intersect.

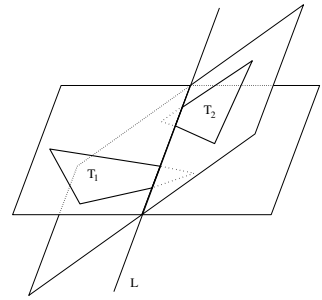


Fig. 2. Triangle intersection. Line L is the intersection between the planes supporting each triangle.

The code of Möller [13] operates as follows. Test whether triangle T_1 intersects π_2, then test whether triangle T_2 intersects π_1. If both are true, construct the line $L = \pi_1 \cap \pi_2$, compute intersection of L with each triangle, then test whether the two segments (on L) intersect. This code, being implemented with floating point, suffers from the analogous flaw as the segment-segment intersection previously described. Two nearly-coplanar triangles may be arbitrarily far apart, yet declared intersecting if the orientation test fails for one vertex. We have observed this situation in practice for triangles generated by the tesselation of an ellipsoid, and for triangles generated with random coordinate values.

The ERIT code [8] operates by first testing whether π_1 intersects T_2. If so, then the segment $S_2 = T_2 \cap \pi_1$ is constructed and tested for intersection with triangle T_1. The segment/triangle intersection is computed in 2D after projection to a suitable plane. Consider an input (pictured on the left of Figure 3) in which (after projection) triangle T_1 (ABC in figure) is skinny and segment S_2 (PQ in

figure) is nearly collinear with lines supporting edges AB and BC of T_1. Suppose that the location of Q with respect to the edge lines is correctly decided, but the approximate 2D orientation tests place P below lines AB and BC. The intersection test is computing as if it were given the perturbed situation pictured in the right panel of Figure 3 and incorrectly declares that S_2 intersects triangle T_1, hence the two triangles intersect.

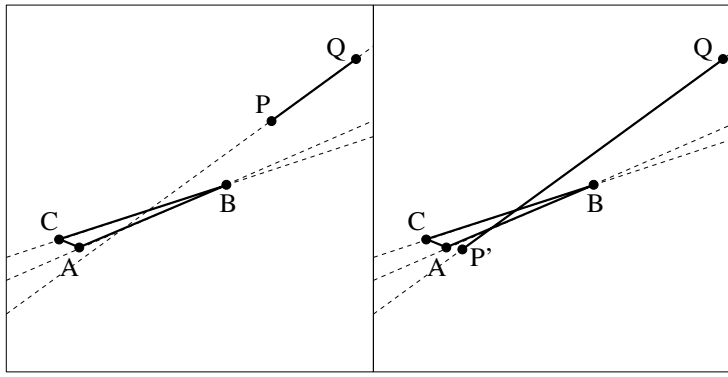


Fig. 3. Potential problem for ERIT method.

In short, both methods can fail on a widely-separated triangle pair. Using a bounding-box preprocessing step will alleviate but not eliminate this problem.

3 Exact Intersection

Triangle intersection can be made exact using an exact orientation test. Using extended precision arithmetic unconditionally to compute the exact value of the orientation test polynomial is possible, but slow. Practical implementations of the orientation test use a floating point filter to handle most cases, reverting to the full computation only when the value is close to zero [2,3,16]. The idea is that a large fraction of inputs can be decided using floating point arithmetic. The easy cases are filtered out quickly, leaving only a few hard cases for exact computation. For example, in a mesh generation application, Aftosmis et al. [1] report on a helicopter data set for which only 68 out of 1.37 million orientation calculations failed to be resolved by the floating point filter.

One route to an exact computation, advocated by Yap and Dubé [17], is to take an existing intersection test and replace the fixed precision calculations with (filtered) arbitrary precision arithmetic so that correct branching decisions are guaranteed. This is made very easy to do with the CORE arithmetic library [5] or LEDA [12]. Both packages enable exact computation of arbitrary arithmetic expressions, though the generality comes at a high cost in running time. Such

packages store an expression graph, used to perform the exact computations in cases where the floating-point filter fails, so they tend to consume a lot of time and memory building the graph [10] even when not needed. We replaced the arithmetic in Möller's code using CORE, for example, and found that it was about 2000 times slower.

Not willing to pay such a high price, we build our intersection test around the highly optimized implementation of the 3D orientation test by Shewchuk [16]. This code implements a single primitive test, so it can be made much faster than the equivalent computation using CORE or LEDA, e.g. by unrolling loops by hand and allocating memory in local variables.

Since the 3D orientation predicate uses the sign of a polynomial, we could have used instead the exact sign of sum algorithm (ESSA) of Ratschek and Rokne (see [6]). ESSA requires that each term of the sum be exactly represented in a machine floating-point number. The 2D orientation predicate is degree two in the coordinate values, so Gavrilova and Rokne were able to use single-precision coordinate values and a double-precision implementation of ESSA. In 3D, the orientation test is degree three in the coordinates, so we cannot represent each term of the sum in a double-precision floating-point word. We would have to limit the inputs to 17 bits of precision when using IEEE double-precision numbers (53-bit significand) for ESSA.

Shewchuk's 3D orientation code works with inputs of either single- or double-precision floating-point numbers. We use his code to implement an exact triangle intersection test as follows. Our code first checks whether triangle T_1 intersects plane π_2. If not, then we have established that the triangles do not intersect and we're done. Otherwise, exactly two edges of T_1 intersect π_2.[1] Each of these edges is checked for intersection with T_2, again using applications of the 3D orientation test. If intersection is not yet established, the same procedure is applied with the roles of T_1 and T_2 interchanged.

3.1 Timing Experiments

Our intersection code is expected to be slower than inexact tests. To get a handle on the performance penalty incurred, we timed triangle intersection tests using our code and using two published codes: TRI_TRI_INTERSECT() of Möller [13], and the TRITRI3D() routine from ERIT 1.1 [8]. Möller's code was updated in 2001 with a newer, slightly faster version that avoids division. Comparisons such as $a/b < c/d$ are re-written as $ad < bc$. This entails evaluating polynomials of larger degree in the coordinates, however. We found that the computation would occasionally overflow the exponent of single precision numbers, so we report times using the original code.

We used the codes as-is. ERIT computes using double precision arithmetic, while Möller's test uses single precision. This potentially gives the latter an advantage in speed, but the limited precision also renders it more fragile. We

[1] In the general case. Our code also handles the degenerate cases where one vertex or one edge of T_1 lies in plane π_2.

tested both single precision and double precision in our exact implementation. On some systems, the times were about the same, while others showed single precision to be up to 30% faster. However, the filter is expected to be more effective when using higher precision, so the times reported here use double precision arithmetic.

The speed of these codes is expected to vary across different hardware, compiler, and input data. We report on two platforms. The first is a 360MHz MIPS R12000-based IRIX 6.5 system with 256MB memory, compiled using the MIPSpro 7.30 compiler with flags `-O2 -n32 -mips4`. The second system is a 1GHz Pentium III machine with 1GB memory, running linux; the compiler is GNU GCC version 2.95 with optimization flags `-O2 -march=i686`. We use the CLOCK() system call to measure the time, which reports times in units of 10 milliseconds. We repeat the entire dataset a number of times so that the total execution time is at least 10 seconds, rendering the error due to clock granularity negligible.

The input data matters because Shewchuk's routines are adaptive, with four levels of filtering. We measure first the best case using coordinates randomly chosen in the interval $[0, 1000]$. All determinant signs were resolved at the first level of filtering, giving the lower bound on the speed penalty. We obtain the upper bound by modifying Shewchuk's code so that it never exits early, thereby obtaining a range for the speed penalty.

The results are tabulated Table 1 in terms of the raw time for a test (averaged over many repetitions of 10000 randomly-chosen triangle pairs) and also a ratio of the time divided by the time for the fastest method (Möller's in our test) giving a slowdown factor. We expect the vast majority of instances to be near the lower end of the range. This depends on the application, and data characteristics, of course. Aftosmis et al. [1] also used Shewchuk's orientation code for their triangle intersection test, and found that only 0.005% of the tests failed the first filter. Even if 1% of the triangle pairs fail the filter, our results indicate a slowdown ratio of 3.7 at most. The exact computation is expected, therefore, to be on the order of 3-4 times slower in practice. It is worth keeping in mind, however, that this slowdown is only for the triangle intersection code. Real-world applications will only spend part of their running time in the triangle intersection routine, so the overall impact on the application is not expected to be as great.

4 Conclusions

Intersection tests based on approximate arithmetic can fail not only on triangle pairs that are in "grazing" contact, but also on pairs that are widely separated. The common practice of using "epsilon" for numeric comparison exacerbates the problem. Practitioners need to be aware of these issues when implementing primitive geometrical tests.

A straightforward exact implementation can have a relatively small speed penalty, as our experiments have demonstrated.

Table 1. Performance of triangle intersection test on pseudorandom data. Time is given in microseconds per test while the slowdown factor (SlowF) is the time divided by the time for method Möller.

	MIPS R12000		Pentium III	
Method	Time	SlowF	Time	SlowF
Möller	0.63	1.0	0.80	1.0
ERIT	0.84	1.3	1.2	1.5
Exact (lower)	2.1	3.3	2.6	3.3
Exact (upper)	30	47.6	34	42.5

References

1. M.J. Aftosmis, M.J. Berger, and J.E. Melton. Robust and efficient cartesian mesh generation for component-based geometry. In *35th AIAA Aerospace Sciences Meeting*, Reno NV, USA, January 1997. AIAA97-0196.
2. Hervé Brönnimann, Christoph Burnikel, and Sylvain Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 165–174, 1998.
3. Christoph Burnikel, Stefan Funke, and Michael Seel. Exact geometric predicates using cascaded computation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 175–183, 1998.
4. Christoph Burnikel, Kurt Mehlhorn, and Stefan Schirra. The LEDA class real number. Technical Report MPI-I-96-1-001, Max-Planck-Institut für Informatik, January 1996.
5. CORE. The core library, v1.5. Web Page, 2002. http://www.cs.nyu.edu/exact/core/.
6. Marina Gavrilova and Jon Rokne. Reliable line segment intersection testing. *Computer-Aided Design*, 32:737–745, 2000.
7. S. Gottschalk, Ming C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *ACM Siggraph*, 1996.
8. Martin Held. ERIT: A collection of efficient and reliable intersection tests. *Journal of Graphics Tools*, 2(4):25–44, 1997.
9. V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A core library for robust numeric and geometric computation. In *Proc. 15th Annual ACM Symp. Comput. Geom.*, pages 351–359, 1999.
10. Michael Karasick, Derek Lieber, and Lee R. Nackman. Efficient delaunay triangulation using rational arithmetic. *ACM Transactions on Graphics*, 10(1):71–91, January 1991.
11. Chen Li and Chee Yap. Recent progress in exact geometric computation. http://cs.nyu.edu/exact/doc/dimacs.ps.gz, June 2001.
12. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
13. Tomas Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2):25–30, 1997.
14. Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, second edition, 1998.

15. T. Ottmann, G. Theimt, and C. Ullrich. Numerical stability of geometric algorithms. In *Proceedings of the third annual symposium on Computational geometry*, pages 119–125. ACM Press, 1987.
16. Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete and Computational Geometry*, 18(3):305–363, October 1997. Source code available at http://www.cs.cmu.edu/~quake/robust.html.
17. Chee Yap and Thomas Dubé. The exact computation paradigm. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean Geometry*. World Scientific, second edition, 1995.
18. Chee K. Yap. Robust geometric computation. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, 1997.
19. Afra Zomorodian and Herbert Edelsbrunner. Fast software for box intersections. *International Journal of Computational Geometry and Applications*, 12:143–172, 2002.

A Parametrically-Optimized Morphing Scheme of Polygonal Meshes

Jie Shen

Department of Computer and Information Science,
University of Michigan – Dearborn, Dearborn, MI 48128, USA
shen@umich.edu

Abstract. In this paper we propose a new parametric morphing scheme for polygonal meshes. The approach consists of three main ingredients: 1) surface partition of polygonal meshes into different patches; 2) a new parametric morphing scheme of constant surface normal flow and 3) parametric shape optimization. The uniqueness of our morphing scheme is that the shape evolving process is controlled by an optimization and by parameterization of surface patches. Numerical experiment indicates the effectiveness of the proposed approach in engineering applications.

1 Introduction

Morphing or metamorphosis has been extensively used in computer graphics to conduct shape interpolation between key frames [1,4-10]. In engineering disciplines, morphing can be used in two scenarios:

(1) A base model is given. Perturbation over this base model is applied in order to generate a better shape design.
(2) Two or more base models are provided. The shape interpolation among these models is applied to get a new and better shape design.

This paper focuses on the first case. As the CPU rate of modern computers has been constantly and dramatically increased, polygonal meshes in engineering analyses could contain as many as 1 million vertices. How to effectively morph this kind of large-scale polygonal meshes is still a problem that has not been well solved. We propose a parametrically-optimized morphing scheme. The main idea of our approach is to conduct a surface partition on the basis of geometric characteristics first, and then to parametrically morph each selected surface patch under the guidance of a constrained optimization.

The organization of this paper is as follows. In Section 2, our surface partition algorithm is briefly introduced. Parametric morphing of constant surface normal flow is given in Section 3, while parametric optimization is explained in Section 4. Numerical experiment is in Section 5, followed by conclusions in Section 6.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 931-940, 2003.
© Springer-Verlag Berlin Heidelberg 2003

2 Surface Partition

Surface partition of unstructured meshes is important to many problems in engineering and science. The partitioning problem may arise from the requirement of computation on multiprocessor architectures. If a surface mesh is involved in a computation, mapping such a surface onto a multiprocessor machine generally requires partitioning the surface mesh into a number of subregions and assigning these subregions to different processors.

In addition, the surface partition may come from the requirement of surface manipulations or morphing based on geometric characteristics of polygonal meshes. In contrast to the case of parallel computation, the objective of the decomposition herein is not to generate equally-sized subregions with minimized boundaries. Instead, the decomposition is controlled by the geometric characteristics such as discontinuity of surface normal and curvature. In this paper, we focus solely on this type of decomposition, and limit our attention on polygonal meshes with discontinuity of surface normal or curvature. As to a mesh without any discontinuity of surface normal or curvature, there is no need to break it down into several subregions from the perspective of shape morphing.

Since the surface partition may be dependent upon several factors such as surface normal and curvature, in order to make things simple, the entire task is divided into three main steps in a hierarchical manner as follows:

Step 1: surface partition by G^1 discontinuity
Step 2: identification of flat regions
Step 3: surface partition by discontinuity of curvatures

Step 2 is conducted on the result of Step 1, while Step 3 is carried out on the result of Step 2. To find out G^1 discontinuity, we use the angle formed by the surface normal of adjacent elements as an index that is called normal angle change in this paper, and to compare this index with a predetermined threshold. Whenever the index is greater than the threshold, we consider that the edge between these two adjacent elements is a sharp edge or feature edge. In order to find each surface partition enclosed by this kind of sharp edges, a breath-first search is proposed to traverse over the polygonal mesh as follows:

(1) set up element neighbor list for each surface element
(2) calculate normal angle change between adjacent elements
(3) perform a breath-first search
  (3.1) initiate from an arbitrary surface element
  (3.2) propagate over the surface until a discontinuity line is encountered, which is identified by a condition: normal angle change > a user-specified angular threshold.
  (3.3) go back to (3.1) and repeat this breath-first search over unprocessed regions until all surface elements are covered by a partition.

In order to simplify things, we propose an idea of identifying flat regions as early as possible such that the task of surface partition by curvature in Step 3 could be reduced. To identify a flat region, we also use the angle formed by the surface normal of adjacent elements. If this angle is smaller than a very small angular threshold, we consider that these two elements form a small portion of a flat region. Similar to

the idea of finding sharp edges, we use another pass of breath-first search over the polygonal mesh as follows:

(1) loop over each surface patch generated by Step 1
  (1.1) perform a breath-first search
    (1.1.1) initiate from a surface element that has, w.r.t. each neighboring element, a normal angle change < a user-specified angular threshold for flat planes
    (1.1.2) propagate over the surface until a boundary line of a flat plane is encountered, which is identified by a condition: normal angle change > a user-specified angular threshold for flat planes.
    (1.1.3) go back to (1.1.1) and repeat this breath-first search over unprocessed regions until all surface elements are processed.
  (1.2) group the remaining elements that do not belong to flat regions into one or more different surface patches by means of their connectivity.

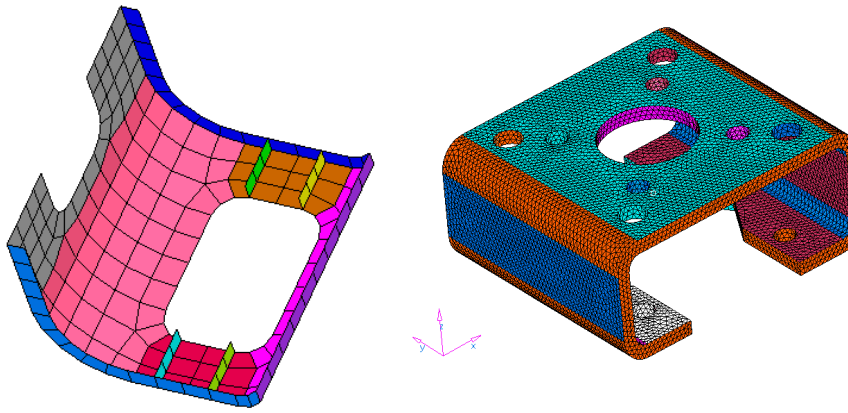
In order to conduct Step 3, estimation of surface curvatures is required. Since a polygonal mesh is an approximation to an underlying surface, surface curvatures can be estimated in either nodal or element formats, depending upon the requirement of how the approximated surfaces will be used in related computation. An effective scheme for estimating discrete surface curvatures of polygonal meshes is proposed by the authors [11] and is supported by a number of lemmas and propositions.

With the surface partition produced by Steps 1 and 2, a third pass of partition is conducted on the basis of curvature discontinuity. This pass is extremely important to curved surfaces, especially when we want to separate features like fillets from others. Our basic strategy is to let the breath-first search find out one subregion with low curvature and group the remaining elements into one or more subregions. The search is controlled to start from an element with all its nodal curvatures smaller than the nodal curvature threshold. The definition of nodal curvatures can be found in reference [11]. In the search propagation, if normal curvature w.r.t. an element edge is smaller than the nodal curvature threshold, the propagation continues. Otherwise, it terminates at that element edge.

Overall, the following procedures are used to conduct surface partition by discontinuity of curvatures:

(1) loop over each surface patch generated by Steps 1 and 2, which is not a flat region
  (1.1) calculate nodal curvatures
  (1.2) calculate average nodal curvature
  (1.3) perform breath-first searches
    (1.3.1) initiate only from an element with a curvature < average curvature
    (1.3.2) propagate over the surface until a termination condition is satisfied: curvature > curvature threshold.
    (1.3.3) If there are still some elements unprocessed, go back to (1.3.1) to initiate another breath-first search.
  (1.4) group the remaining elements, which do not belong to regions formed by breath-first searches, into one or more different surface patches by means of their connectivity.

Figure 1 shows the results of our surface partition algorithm applied on different polygonal meshes.



(a) (b)

Fig. 1. Surface partition of polygonal meshes with different patches being represented in different colors

3 Parametric Morphing of Constant Surface Normal Flow

The basic idea of parametric morphing is a process called as constant surface normal flow in which the perturbation of each partitioned surface patch is carried out by a constant magnitude (order 0) of movement of all nodes along the direction of their surface normal, as shown in Figure 2. If the surface patch is planar, the parametric morphing of order 0 is a rigid movement. If the surface patch is curved, then the morphing is non-rigid. This perturbation method is supported by the following lemma and proposition.

Lemma 1: Let $\mathbf{x} = \mathbf{x}(u,v)$ be a regular parameterized surface patch[1]. Assume that the constant surface normal flow is used to generate another surface patch:

$$\mathbf{y}(u,v) = \mathbf{x}(u,v) + a\hat{N}(u,v), \tag{1}$$

where a is a constant that denotes the length of perturbation vectors. $\hat{N}(u,v)$ is a unit surface normal. Then at regular points, the Gaussian and mean curvatures of $\mathbf{y}$ are respectively

[1] A regular surface refers to a surface $S \subset R^3$ that is locally diffeomorphic to R^2. In other words, for each point $P \in S$, there exists a neighborhood V of P in *S*, an open set $U \subset R^2$, and a map $\mathbf{x} : U \to V$, which is a diffeomorphism [2].

$$K^{(y)}(u,v) = \frac{K^{(x)}(u,v)}{1-2H^{(x)}(u,v)a + K^{(x)}a^2},$$
$$H^{(y)}(u,v) = \frac{H^{(x)}(u,v) - K^{(x)}(u,v)a}{1-2H^{(x)}(u,v)a + K^{(x)}a^2} \tag{2}$$

where $K^{(x)}(u,v)$ and $H^{(x)}(u,v)$ are Gaussian and mean curvatures of **x**. $K^{(y)}(u,v)$ and $H^{(y)}(u,v)$ are Gaussian and mean curvatures of **y**.

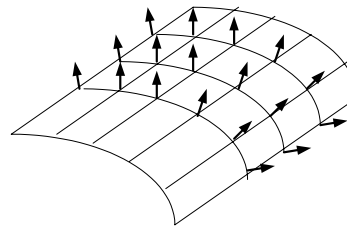


Fig. 2. Constant surface normal flow

Proof: By differentiating $\mathbf{y}(u,v)$ in Equation (1) with respect to *u* and *v*, we get

$$\mathbf{y}_u = \mathbf{x}_u + aN_u, \quad \mathbf{y}_v = \mathbf{x}_v + aN_v. \tag{3}$$

Then the cross product of the above two vectors is

$$\mathbf{y}_u \otimes \mathbf{y}_v = \mathbf{x}_u \otimes \mathbf{x}_v + a\mathbf{x}_u \otimes N_v + aN_u \otimes \mathbf{y}_v + a^2 N_u \otimes N_v, \tag{4}$$

where $\otimes$ refers to vector cross product. Since

$$S(\mathbf{x}_u) \otimes \mathbf{x}_v + \mathbf{x}_u \otimes S(\mathbf{x}_v) = 2H(u,v)(\mathbf{x}_u \otimes \mathbf{x}_v)$$
$$\text{and} \qquad S(\mathbf{x}_u) \otimes S(\mathbf{x}_v) = K(u,v)(\mathbf{x}_u \otimes \mathbf{x}_v), \tag{5}$$

where S is shape operator or Weingarten map [3] such that

$$S(\mathbf{x}_u) = -N_u, \qquad S(\mathbf{x}_v) = -N_v, \tag{6}$$

hence,

$$\mathbf{y}_u \otimes \mathbf{y}_v = (1-2H^{(x)}(u,v)a + K^{(x)}(u,v)a^2)(\mathbf{x}_u \otimes \mathbf{x}_v). \tag{7}$$

Now by using Equations (3), (5), (6) and (7), we can write the curvatures of **y** as follows:

$$K^{(y)}(u,v)=\frac{N_u\otimes N_v}{\mathbf{y}_u\otimes\mathbf{y}_v}=\frac{K^{(x)}(u,v)}{1-2H^{(x)}(u,v)a+K^{(x)}(u,v)a^2},$$

$$\begin{aligned}H^{(y)}(u,v)&=\frac{-N_u\otimes y_v-y_u\otimes N_v}{2(\mathbf{y}_u\otimes\mathbf{y}_v)}\\&=\frac{-N_u\otimes\mathbf{x}_v-aN_u\otimes N_v-\mathbf{x}_u\otimes N_v-aN_u\otimes N_v}{2(\mathbf{y}_u\otimes\mathbf{y}_v)}.\\&=\frac{H^{(x)}(u,v)-K^{(x)}(u,v)a}{1-2H^{(x)}(u,v)a+K^{(x)}(u,v)a^2}\end{aligned}\tag{8}$$

□

On the basis of Lemma 1, the following proposition is ready to be derived.

Proposition 2: If the curvatures of a surface patch are continuous, i.e., the surface patch is smooth except on its boundary, then the perturbation scheme of constant surface normal flow does not destroy the smoothness of the surface patch after the shape morphing, as long as a is controlled in a proper range.

Proof: Since the curvatures of the surface patch represented by $\mathbf{x}(u,v)$ in Equation (1) is initially continuous, at each regular point (u_0,v_0) of $\mathbf{x}$ we have

$$\lim_{\substack{u\to u_0\\v\to v_0}}K^{(x)}(u,v)=K^{(x)}(u_0,v_0),\quad\lim_{\substack{u\to u_0\\v\to v_0}}H^{(x)}(u,v)=H^{(x)}(u_0,v_0).\tag{9}$$

By Equations (8) and (9), we can write

$$\begin{aligned}\lim_{\substack{u\to u_0\\v\to v_0}}K^{(y)}(u,v)&=\frac{\lim\limits_{\substack{u\to u_0\\v\to v_0}}K^{(x)}(u,v)}{1-2a\lim\limits_{\substack{u\to u_0\\v\to v_0}}H^{(x)}(u,v)+a^2\lim\limits_{\substack{u\to u_0\\v\to v_0}}K^{(x)}(u,v)}=K^{(y)}(u_0,v_0),\\\lim_{\substack{u\to u_0\\v\to v_0}}H^{(y)}(u,v)&=\frac{\lim\limits_{\substack{u\to u_0\\v\to v_0}}H^{(x)}(u,v)-a\lim\limits_{\substack{u\to u_0\\v\to v_0}}K^{(x)}(u,v)}{1-2a\lim\limits_{\substack{u\to u_0\\v\to v_0}}H^{(x)}(u,v)+a^2\lim\limits_{\substack{u\to u_0\\v\to v_0}}K^{(x)}(u,v)}=H^{(y)}(u_0,v_0).\end{aligned}\tag{10}$$

Thus, the curvatures of surface patch **y** are continuous as long as the denominators in Equations (10) are not zero, and the smoothness of the surface patch after the shape optimization is not changed. For the sake of convenience, we rewrite the denominator in Equation (10) as $1-2Ha+Ka^2$, where a should be positive. If $H=0$ and $K=0$, then the denominator cannot be zero. Thus, for a plane, the constant surface normal flow maintains the continuity over the surface patch. If $H=0$ and

$K \neq 0$, surface continuity is kept as long as $a < \sqrt{1/K}$. Similarly if $H \neq 0$ and $K = 0$, surface continuity is not destroyed as long as $a < 1/(2H)$. Next let's look at the general case in which $H \neq 0$ and $K \neq 0$. Since $H^2 \geq K$, if $K > 0$ and $H < 0$, then the surface continuity is maintained for any a (> 0). Otherwise, a should less than $\dfrac{H - \sqrt{H^2 - K}}{K}$ to avoid that the denominator becomes zero. □

4 Parametric Shape Optimization of Polygonal Meshes

The parametric shape optimization can be expressed by:

$$\text{Minimize } f(\mathbf{y}) = \frac{1}{2}\mathbf{u}(\mathbf{y})^T \mathbf{K}(\mathbf{y})\mathbf{u}(\mathbf{y}) = \frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}(\mathbf{y})^T \boldsymbol{\sigma}(\mathbf{y}) d\Omega, \tag{11a}$$

$$\text{subject to} \quad \frac{V(\mathbf{y})}{V_0} = 1, \tag{11b}$$

$$\mathbf{y}^L \leq \mathbf{y} \leq \mathbf{y}^u, \tag{11c}$$

where the objective function $f(\mathbf{y})$ is called a compliance that is half of strain energy over a design domain Ω. If there are multiple load cases, a weighted compliance is used on the basis of linear combination of compliance in each load case with a specific weighting factor. $\mathbf{u}$ and $\mathbf{K}$ are displacement vector and stiffness matrix, respectively. $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ are strain and stress tensors, respectively. Ω refers to the domain occupied by the structure. $V(\mathbf{y})$ and V_0 are the volume of the structure at the current iteration and before the first iteration of the optimization process, respectively. Equation (11b) refers to a constant volume constraint on the polygonal mesh, i.e., we would like to morph a 3-D polygonal mesh without changing its enclosed volume, which reflects how much material is needed to produce such a shape model. The perturbation vector approach is used in this paper such that the components of design variable vector $\mathbf{y}$ are related to structure shape by the following formula:

$$\mathbf{x} = \mathbf{x}^0 + \sum_{i=1}^{m} \mathbf{P}^i y^i, \tag{12}$$

where $\mathbf{x}^0$ and $\mathbf{x}$ refer to the geometric configure of the structure before and after the shape optimization, respectively. $\mathbf{P}^i$ and y^i are *i*-th perturbation vector and *i*-th design variable, respectively. $\mathbf{P}^i$ is defined by the constant surface normal flow of *i*-th surface patch. m denotes the number of design variables in vector $\mathbf{y}$.

Note that Equation (11a) defines only one category of optimization problems. Our parametric morphing scheme is not tightly tied to this category, and might be used in some other problems such as minimization of frequency response, uniform stress or minimization of stress. Such extension will be part of future work.

In this paper, the parametrically-optimized morphing is carried out by using the optimizers available in OptiStruct^{TM}, a dual method and a primal feasible directions method that are both based on the convex linearization of the design space.

5 Numerical Experiments

The proposed approach is implemented in VC++ and tested on a Pentium III HP PC. Control arm is a typical structure encountered in the structure design of automobiles, as shown in Figure 3. It contains one constant-cross-section beam (at the rear) and two tapered beams (along the sides). The non-design elements are shown in dark grey color and the design elements are in light grey.

Figure 4 illustrates the surface partition results after removing all non-design elements. Among all 38 partitioned patches, users have a choice to choose certain number of patches as design variables for an optimized morphing. We do not want automate this step, because it will provide considerable amount of flexibility to try out different combination of shape design variables.

In our test, 11 design variables are chosen, each of which corresponds to a partitioned surface patch. The visible design surface patches are marked in Figure 4. Since there is a V-shaped discontinuity at the top surface of beam 3 in Figure 4, two surface patches are generated to cover that surface. The comparison between the original and optimized geometric configurations is illustrated in Figure 5. The weighted compliance (i.e., the objective function) of the entire structure is reduced by 12.95 %. Note that the optimization in this example is a minimization problem with a constraint of constant volume for the polygonal mesh during the optimization process. The constant volume means the same amount of material to be used for both original and optimized shape models.

Unlike the conventional morphing methods [1,4-10] that require both beginning and ending shapes to be predefined, our scheme requires only the beginning shape and relies on a constrained optimization to obtain an optimal ending shape.

6 Conclusions

In this paper a new morphing scheme is proposed for polygonal meshes. The unique features of our approach include 1) surface partition of large-scale polygonal meshes into a few number of surface patches and 2) parametric morphing over each selected surface patch under the guidance of a constrained optimization. Our approach is not be suited for entertainment industry, but is useful for shape design in many engineering disciplines such as automotive and aerospace.

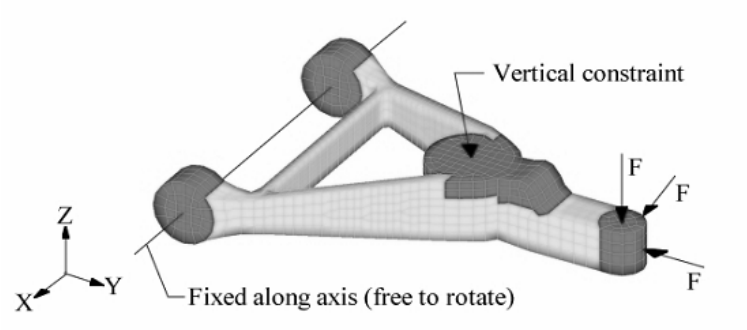


Fig. 3. Polygonal mesh of a control arm

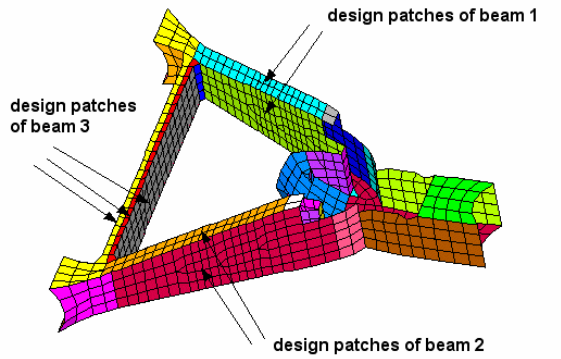
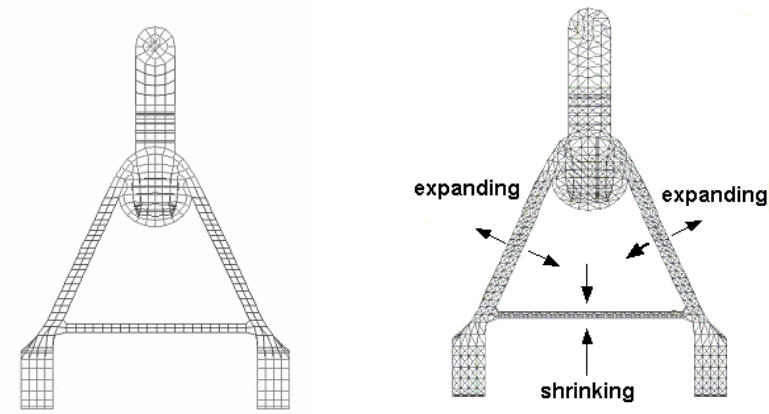


Fig. 4. Surface partition of the polygonal mesh



(a) (b)

Fig. 5. Original versus optimized geometric shape. (a) original; (b) optimized

Acknowledgements

This work is supported in part by University of Michigan OVPR and Rackham research grants.

References

1. Catmull E.: The Problems of Computer-assisted Animation. Computer Graphics 12(1978): 348-353
2. Do Carmo MP.: Differential Geometry of Curves and Surfaces. Prentice Hall (1976)
3. Gray A.: Modern Differential Geometry of Curves and Surfaces with Mathematica. CRC Press, (1997)
4. Gregory A., State A, Lin M., Manocha D. and Livingston M.: Interactive surface decomposition for polyhedral morphing. The Visual Computer 15 (1999) 453-470.
5. Kanai T, Suzuki H. and Kimura F.: 3D geometric metamorphosis based on harmonic map. The Visual Computer 14 (1998) 166-176.
6. Kent, J. R., Parent, R. E., and Carlson, W. E.: Establishing correspondences by topo-logical merging: a new approach to 3-D shape transformation. Proceedings of Graphics Interface '91 (1991) 271-278.
7. Korfiatis I. and Paker Y.: Three-dimensional object metamorphosis through energy minimization. Comput & Graphics 22 (1998) 195-202.
8. Lazarus F. and Verroust A.: Metamorphosis of cylinder-like object. International Journal of Visualization and Computer Animation 8 (1997) 131-146.
9. Lee W.F., Dobkin D., Sweldens W. and Schroder P.: Multiresolution mesh morphing. ACM SIGGRAPH Computer Graphics Proc (1999) 343-350.
10. Sederberg T.W. and Greenwood E.: A Physically Based Approach to 2-D Shape Blending. ACM SIGGRAPH Computer Graphics Proc 26 (1992) 25-34.
11. Shen, J. and Yoon, D.: A hierarchical approach to surface partition of polygonal meshes. Proceedings of International Conference on Scientific and Engineering Computation (ICSEC) (2002) 659-662.

Author Index

Abbey, K., I-358
Abdel Razek, M., I-170
Abou-Chakra, H., I-493
Abreu Soares, E. de, II-589
Adamidis, P.A., I-472
Aggarwal, J.K., III-650
Aggarwal, V., III-168
Agu, E., III-336
Ahn, B.J., II-135
Ahn, G., II-320
Ahn, J., III-508
Ahn, M.K., II-338
Ahn, S., II-1, II-664
Alarcon, V.J., I-539
Albert, F., III-569
Albuquerque, E.L., II-774
Allen, K.P., II-290
Alves, C.E.R., II-249
Arnal, A., III-366
Atiqullah, M.M., I-263
Aubanel, E., III-60

Bae, E.-H., II-210
Bae, S.-H., III-690, III-701
Balah, M., III-670
Bang, Y.-C., II-135
Barenco Abbas, C.J., III-91
Barhen, J., II-70
Barsky, B.A., III-246, III-256
Bartels, R., I-67
Barton, A.J., II-259
Basermann, A., III-139
Bassen, G., I-654
Beletskyy, V., III-913
Bénallal, M., I-99
Ben-Artzi, M., I-809
Benjamim Werneck, V.M., II-589
Bertazzon, S., III-23
Bespamyatnikh, S., III-807
Bhavsar, V.C., I-724, III-60
Bhowmick, S., I-828
Bianco, M., I-893
Bischof, C.H., II-78
Biswas, R., II-279
Blais, J.A.R., I-105
Blessing R., E., II-684
Boada, I., III-468, III-836
Bohbot, J., I-483
Boland, R.P., III-903
Bounaïm, A., II-705
Bradley, M.S.A., I-493
Brent, R.P., I-1
Brezany, P., I-446
Brezanyova, J., I-446
Brezhneva, O., I-903
Bridle, I., I-493
Brimicombe, A.J., III-1
Brito F. Oliveira, A., II-589
Bruno-Castañeda, C., III-84
Bu, J., I-59, I-114
Bücker, H.M., II-78
Burge, L.M., III-33
Byun, H.-S., III-731

Caballero, J., I-799
Caballero-Gil, P., III-84
Cáceres, E.N., II-249
Camp, O., III-91
Campos, A.E., I-967
Carthy, J., I-358
Cha, J.S., III-326
Chae, K., II-210
Chae, Y., II-892
Chakraborty, A., II-715
Chang, B., II-320
Chang, C.-L., II-859
Chang, H., II-656
Chang, H.J., II-383
Chapelle, P., I-493
Cheetham, J., II-300
Chen, C., I-59, I-114
Chen, W., II-705
Cheng, M.-C., II-881
Chi, E., II-901
Ching, W.-K., I-912
Cho, J.-B., II-552
Cho, M., I-79
Cho, S., I-437
Cho, S.-D., III-539
Cho, S.E., II-455

Cho, S.H., II-60, II-517
Cho, S.K., III-711
Cho, Y., III-448
Choi, B.-T., I-957, III-488
Choi, C., I-742
Choi, D.H., II-230
Choi, D.K., II-144
Choi, H.-I., I-633, II-310
Choi, J., I-79
Choi, J.S., I-227
Choi, K.-S., I-957
Choi, M.S., II-328
Choi, S.-Y., II-674
Choi, S.C., I-227
Choi, T.-H., III-539
Choi, Y.-S., II-31
Choi, Y.S., II-391, II-455
Choo, H., II-144, II-220
Choudhari, M., II-859
Christakis, N., I-493
Chung, C.H., II-383, II-411, II-599, III-630
Chung, J.W., II-1, II-664
Chung, T.M., II-320
Chung, W.H., III-711
Clark, D.E.R., III-894
Coleshill, J.E., I-51
Coll, N., III-836
Conesa, J., III-549
Contero, M., III-549, III-569
Corsava, S., II-374
Crespo, E.M., I-263
Croisille, J.-P., I-809
Cross, M., I-493

Dai, G., II-695
Dailyudenko, V.F., I-921
Daněk, J., I-587
Daripa, P., II-764
Das, S.K., II-279
Dash, R.K., II-764
Davenport, S., I-523
Dazzi, R.L.S., I-674
Dehne, F., II-249, II-300
Denk, F., I-587
Denzinger, J., III-148
Dervieux, A., II-85
Desbarats, P., III-498
Ding, J., I-623
Ding, R., I-338
Dixon, M.F., I-818, III-226
Dong, H., III-376
Drummond, A., I-358
Dunnion, J., I-358
Dupuy, S., I-31
Durán, A., III-114
Dutoit, D., I-368
Dutta, S., II-782

Eberl, H.J., I-503
Eberle, S., I-217
Effantin, B., I-245
Egges, A., I-31
Emiris, I.Z., III-397
Enright, W.H., II-725
Ercan, M.F., I-752, I-987
Estrela Abad, M.P., III-148

Fang, V., III-189
Fatta, G. Di, I-417
Feixas, M., III-275
Feng, L., I-462
Fernández, M., III-458
Ferworn, A., I-51
Fishelov, D., I-809
Flauzac, O., III-69
Forsyth, P.A., III-158
Forth, S.A., II-115
Foufou, S., III-387
Fox, P., III-579
Fracari Branco, V.G., III-148
Fragoso D., O.G., I-704
Frasson, C., I-170
Fugère, J., III-69
Fung, Y.F., I-752, I-987

Galeriu, C., II-755
Gallegati, M., I-770
Gänser, H.-P., I-927
Gao, J., III-817
Garber, I., I-654
Garcia, E., III-559
Garcia, T., II-239
García-Villalba, L.J., III-79, III-91, III-129
Garg, A., III-876
Gaur, D.R., I-254
Gavrilova, M.L., III-751, III-827
Ge, B., I-663
Geng, W., III-620
Getov, V., II-374

Giering, R., II-95
Giulioni, G., I-770
Glimm, J., I-558, II-782
Gobbert, M.K., II-290
Göhner, P., I-217
Gola, M., I-180
Gomes, A.J.P., III-418, III-478
Gomis, J.M., III-569
González S., G., I-124, I-160
González, L., I-937
Gopalakrishnan, S., II-715, II-745
Gottlieb, J.J., II-725
Grove, J.W., II-782
Gueorguieva, S., III-498
Gupta, R., III-817
Gustedt, J., II-269
Guster, D., I-568

Ha, J.C., II-338
Ha, K.-J., I-875
Hagan, D., I-818
Hahm, J.-H., II-135
Hahn, K.-S., I-733
Hamdi, S., II-725
Han, S., I-144, I-761
Han, S.-Y., I-348, II-171
Harvey, D.J., II-279
Hascoët, L., II-85
Haworth, D.C., I-643
He, H., III-199
Heimbach, P., II-105
Hendriks, H., I-388
Henry, K.A., III-33
Heo, Y.-J., I-883
Hernández, J.C., III-123
Hernández Castro, C.J., III-79
Hernández Castro, J.C., III-79
Hernández-Goya, C., III-84
Hesse, M., III-751
Hill Jr., F.S., III-336
Hitt, D.L., I-513, II-837
Hlaváček, I., I-587
Holm, S., II-705
Hong, C.P., I-855
Hong, D., II-445
Hong, H.T., II-498
Hong, J., I-761
Hong, K.-S., II-639
Hong, S., I-144, II-892, II-909
Hong, S.-K., II-924
Hong, S.J., III-508
Horn, D.R., III-246, III-256
Hu, J., III-579
Huang, S., III-60
Huettmann, F., III-43
Huh, E.-N., II-463
Hungershöfer, J., I-927
Hurtado, F., III-766
Hwang, I., II-445
Hwang, J., II-411
Hwang, S., I-733
Hwang, T., II-482

Iglesias, A., III-660
Ihm, I., I-41
Im, E., I-733

Jaekel, U., III-139
Janciak, I., I-446
Jang, H.-J., I-348
Jang, S.-W., I-633
Jang, W.-S., II-135
Jeon, J.-C., I-947
Jeong, C.-S., II-562, II-674
Jeong, K., I-733
Jiang, L., II-859
Jiawan, Z., III-266
Jiménez, E., III-216
Jin, H., I-199
Jizhou, S., III-266
Jo, T., I-456
Joen, B.-k., II-646
Joh, M., I-41
Joo, B.-G., II-171, II-646
Jorge, J., III-549
Joukov, O.D., I-848
Jun, S.-I., II-552
Jun, W., II-892, II-909, II-924
Jung, B., III-721
Jung, H.K., II-41
Jung, M.-S., II-542, II-552
Jung, S.-H., II-472

Kagan, L., II-811
Kaltenbach, M., I-170
Kaminski, T., II-95
Kanda, T., III-776
Kang, C., II-445
Kang, C.-W., II-41
Kang, H., II-163

Kang, H.G., II-455
Kang, M., II-445
Kang, S., II-482
Kang, S.-J., III-599
Karunamuni, R., I-623
Kasprzak, A., I-180
Kazmi, S.H.K., III-199
Kechroud, R., II-847
Khaliq, A.Q.M., III-199
Khan, M.A., III-589
Kheddouci, H., III-387
Kichiji, N., I-770
Kijima, R., III-701
Kim, B., II-482, II-532
Kim, B.K., III-680
Kim, C.-H., III-599, III-640
Kim, C.H., I-855, III-528
Kim, D., II-391, III-847
Kim, D.-S., III-448, III-680, III-847
Kim, D.-W., II-542
Kim, D.Y., II-230
Kim, E.S., I-330
Kim, E.Y., I-312
Kim, G.-Y., I-633, II-310
Kim, H., III-448
Kim, H.-C., II-391
Kim, H.-J., II-60
Kim, H.-K., II-490
Kim, H.C., II-1
Kim, H.G., II-230
Kim, I., II-482
Kim, J.-G., I-427
Kim, J.G., II-50
Kim, J.J., I-855
Kim, J.O., II-383, II-411, II-599, III-630
Kim, K., I-144, I-761, II-163, II-220
Kim, K.-C., II-646
Kim, K.J., II-50
Kim, M., I-733, II-356, II-365, II-916
Kim, M.J., II-656
Kim, N.-Y., I-865
Kim, P., I-79, II-401
Kim, P.S., I-190, I-209
Kim, P.T., I-388
Kim, R.Y.-C., II-646
Kim, S., I-742, I-998, II-664
Kim, S.-J., III-599, III-640
Kim, S.-K., III-640
Kim, S.-Y., III-488
Kim, W., II-310, II-401
Kim, W.-S., III-690, III-701
Kim, Y., I-302, I-957
Kim, Y.K., I-209
Kingery, W., I-539
Kirk, H., I-558
Klahr, D., I-483
Klein, S.A., III-246, III-256
Knorr, W., II-95
Ko, I.-J., II-310
Ko, S.-J., I-302, I-957
Koh, J.G., II-437, II-455
Kong, H., II-401
Kotsireas, I.S., III-397
Kouh, H.-J., I-684
Koulisianis, M.D., III-178
Krajecki, M., III-69
Krishnamurti, R., I-254
Krot, A.M., I-398, I-407
Ku, K.-M., I-875
Kuri-Morales, A.F., I-378
Kwak, D.J., II-328
Kwak, J., II-191
Kwiesielewicz, M., III-208
Kwon, E.-S., III-690
Kwon, S., I-855

Lai, W., III-376
Lanquetin, S., III-387
Lasserre, J.B., I-273, III-866
Lee, B., I-41
Lee, B.-G., II-463
Lee, B.-w., II-525
Lee, B.R., III-630
Lee, D.C., II-11, II-50, II-60
Lee, G., II-924
Lee, H.C., III-448, III-680
Lee, H.J., II-338
Lee, H.K., II-356, III-317
Lee, J., II-631
Lee, J.-i., II-220
Lee, J.-Y., I-41
Lee, J.H., I-190
Lee, J.J., II-624
Lee, K.H., I-714, I-1035, III-731, III-741
Lee, K.Y., II-624, II-631
Lee, N.-Y., II-310
Lee, S., I-733, II-191, II-391, II-419
Lee, S.-E., I-348
Lee, S.-H., II-31
Lee, S.H., II-599

Lee, T.-J., II-125
Lee, T.-Y., III-285, III-518
Lee, V.C.S., III-189
Lee, W., II-411
Lee, W.-H., I-883, II-348
Lee, Y., II-437, II-631
Levinski, K., III-428
Lew Yan Voon, L.C., II-735, II-755
Li, C., I-291
Liang, R., I-59, I-114, I-462
Lin, C.-H., III-518
Lin, P.-H., III-285
Linke, J., III-43
Liu, C., II-859
Liu, G., I-322
Liu, J.-C., I-998
Lluch, A., III-366
Lo Re, G., I-417
Lobão, D.C., II-871
Lončarić, J., II-801
Love, E., III-609
Luan, S., II-695
Luengo, F., III-660

Mărginean, F.A., I-614
Martínez, R., III-346
Matus, P., II-791
Mauriz, P.W., II-774
McDonald, K., I-558
McGarry, M., II-837
McInnes, L., I-828
Mé, L., III-91
Mehdizadeh, O.Z., II-819
Mei, S., III-12
Melnik, R.V.N., II-755, II-791
Meng, I-H., I-601
Meunier, J., I-99
Meza, F., I-967
Min, L.F., III-428
Minaev, S., II-811
Minervina, H.B., I-407
Misra, S.C., I-724
Mitra, P., III-787
Mogilenskikh, D.V., III-407
Monterde, J., III-366
Moon, S.J., II-328, II-338
Moon, Y., II-401
Morgado, F., III-418
Morillo, P., III-458
Mukhopadhyay, A., III-787, III-797
Mun, Y., II-356, II-365, II-463

Na, W.S., II-11
Naumann, U., II-105
Navarro, L., I-160
Navazo, I., III-468
Naya, F., III-549
Nazaré M.A. Hahne, M., I-674
Nedoma, J., I-587, I-977
Neumann, U., III-579
Neveu, M., III-387
Ng, M.M., I-912
Nguyen, D., III-33
Norris, B., I-828
Nugues, P., I-31, I-368
Nur, M.A., I-67

Ocone, R., III-894
Ødegård, Å., II-705
Oğuz, C., I-987
Oh, B.S., III-528
Oh, C.H., II-411
Oh, K., II-401
Oh, S., II-191
Oh, W., I-558
Orduña, J.M., III-458
Ortiz-Posadas, M.R., I-378, I-674
Otero, C., III-438
Oun, C.D., II-507

Paek, H.W., II-599
Pan, Z., I-59, I-114
Pang, J.A., III-246, III-256
Papatheodorou, T.S., III-178
Paraschivoiu, M., II-819
Park, C., II-901, II-916
Park, G.-L., II-155
Park, H., II-571
Park, H.-H., III-856
Park, H.-J., II-607
Park, J., II-365, II-463, III-650
Park, J.-C., II-181
Park, J.-K., II-144
Park, J.-M., II-210
Park, J.H., II-656
Park, J.K., I-227
Park, J.Y., III-448, III-680
Park, K., I-998
Park, K.-j., II-21
Park, K.-Y., II-171

Park, K.h., I-427
Park, S., I-41, III-650
Park, S.-C., II-472
Park, S.H., I-79, I-312
Park, Y.S., III-317, III-326
Patel, M.K., I-493
Pazos R., R.A., I-124, I-134
Pee, J.I., II-498
Pendharkar, P.C., II-581
Peraire, J., I-1035
Percher, J.-M., III-91
Pérez O., J., I-134, I-160, I-704
Perminov, V., I-549
Pico, J. del, III-559
Pierro, M. Di, I-1045
Pinciu, V., III-886
Pitre, S., II-300
Plassmann, P.E., I-291, I-643
Pomplun, M., I-633
Ponce, E., III-114, III-123
Pooley, D.M., III-158
Povitsky, A., II-826, II-871
Privman, V., II-881
Pryce, J.D., II-115
Puttini, R.S., III-91
Pyeon, H.-J., II-210
Pyun, J.-Y., I-302, I-957

Qi, X.T., I-987
Quan, Y., I-1006

Raghavan, P., I-828, I-840
Rahi, A., I-153
Rajabi, M.A., I-105
Rakoto-Ravalontsalama, N., I-533
Rao, S.V., III-797
Rasch, A., II-78
Rau-Chaplin, A., II-300
Rayner, P., II-95
Redouane, A., I-694
Reister, D.B., II-70
Resch, M.M., I-472
Rheem, J.Y., II-624
Ribagorda, A., III-123
Rigau, J., III-275
Rim, H., II-901
Rishe, N.D., I-848
Ro, S., II-445
Robbins, S., III-923
Rocha, J., I-799
Rodger, J.A., II-581
Rodriguez O., G., I-134
Romero, D., I-134
Rossokhata, N., I-1016
Roy Mahapatra, D., II-715, II-745
Ruskin, H.J., I-577
Rusnak, M., I-446
Rusu, A., III-876
Ruz, C., I-967
Rybak, I., II-791
Ryoo, I.T., II-11
Ryoo, S.-T., III-539
Ryou, H.-b., II-21

Saad, Y., II-847
Sadarangani, K.B., I-799
Saikin, S., II-881
Salman, A., III-609
Salman, R., III-609
Samavati, F.F., I-67
Samulyak, R., I-558
Santaolaya S., R., I-134, I-704
Sarfraz, M., III-589, III-670
Sbert, M., III-275, III-295, III-306, III-346
Schamberger, S., III-49
Schiesser, W.E., II-725
Scholze, M., II-95
Schramm, É., III-356
Schreck, P., III-356
Schumacher, J., III-139
Seara, C., III-766
Seed, G.M., III-894
Seguel, J., I-1025
Selim, H.M., I-539
Sellarès, J.A., III-836
Semé, D., II-239
Seo, D.-i., II-31
Seron, F.J., III-559
Serrão Lanzillotti, H., II-589
Serrão Lanzillotti, R., II-589
Seshaiyer, P., I-523
Sethia, S., III-766
Sharp, D.H., II-782
Shen, J., III-931
Shen, M., II-881
Sheppard, J., I-358
Shim, Y.-C., II-181
Shin, H., III-711
Shin, I.-H., II-155
Sierra Cámara, J.M., III-79

Sierra, J.M., III-123
Silva Ávila S. da, II-589
Silva, F.G.M., III-478
Sivashinsky, G., II-811
Slim, C., III-236
Slusanschi, E., II-78
Sohn, S.-W., II-31
Son, S., III-741
Song, J., I-761
Song, J.S., II-201, II-220
Song, S.I., II-498
Song, S.W., II-249
Sosa S., V.J., I-124, I-134, I-160
Soulaimani, A., II-847
Sourin, A., III-428
Sourina, O., II-427
Sousa Jr., R. de, III-91
Souza Portella, E., II-589
Srinivasan, A., III-168
Stehlík, J., I-587
Subramanian, G.H., II-581
Sugihara, K., III-776, III-847
Suh, W., II-571
Sultanov, R.A., I-568
Szirmay-Kalos, L., III-346

Tadjouddine, M., II-115
Tae, K.S., II-614
Taillon, P.J., II-300
Tan, C.J.K., I-818, III-226
Tan, G., I-199
Teranishi, K., I-840
Thomenius, K., II-705
Tjoa, A.M., I-446
Tkachova, P.P., I-398
Togores, R., III-438
Torcy, P. de, I-368
Torres, A., I-483
Toussaint, G., III-762
Trabelsi, A., III-236
Tret'yakov, A., I-903
Tsynkov, S., II-801
Tuzun, U., I-493
Tveito, A., II-705

Uden, E. van, III-208
Urrutia, J., III-903
Urso, A., I-417
Uthariaraj, R., II-684

Valdés, J.J., II-259
Valor, M., III-569
Vasyltsov, I., I-282
Vatistas, G., II-826
Vavřík, P., I-587
Vázquez, M., II-85
Vázquez, P.-P., III-295, III-306
Veljkovic, I., I-643
Vetzal, K.R., III-158
Villa, J.L., I-533
Vincze, S., I-238
Voss, D.A., III-199

Wang, L., II-791
Wang, R., I-358, I-577
Wang, W.-H., III-518
Weigang, L., III-148
Whitesides, S., III-923
Wierum, J.-M., I-927
Willatzen, M., II-735, II-755
Wilson, D.C., I-358
Winitzki, S., I-780, I-790
Won, D., II-191
Woo, C., I-733
Woo, Y., II-482
Woo, Y.-J., II-562
Wozniak, M., I-89
Wu, S., I-199
Wyvill, B., I-67

Xuan, Z.C., I-714, I-1035

Yang, C., I-840
Yang, J., I-1006
Yang, L., I-11, I-21, I-153
Yang, S., II-656
Yang, T.-H., III-285
Yang, W.-P., I-601
Yang, X.Y., III-894
Yang, Y., III-376
Yang, Z., I-322
Ye, C., I-1006
Yim, W.Y., II-599
Yoe, H., II-437, II-455
Yoo, H., II-41
Yoo, H.S., I-330
Yoo, J.-P., II-220
Yoo, J.S., II-498
Yoo, K.-S., II-348
Yoo, K.-Y., I-865, I-875, I-883, I-947

Yoo, S.K., III-711
Yoo, W.-H., I-684
Yoon, K.H., III-317, III-326, III-539
You, S., III-579
Youn, H.Y., I-227
Young, K.S., II-507
Yu, D., III-376
Yu, G., III-620
Yu, M., III-246, III-256
Yu, S.J., II-201
Yuen, W.-O., I-912
Zambrano S., L., I-704
Zeng, H., I-539
Zhang, Y., II-782
Zheng, T., II-826
Zhigang, S., III-266
Zhou, X., I-513
Zhu, J., I-539
Zimmermann, P., I-1
Zolver, M., I-483
Zunce, W., III-266