George Bebis Richard Boyle
Bahram Parvin Darko Koracin
Nikos Paragios Syeda-Mahmood Tanveer
Tao Ju Zicheng Liu Sabine Coquillart
Carolina Cruz-Neira Torsten Müller
Tom Malzbender (Eds.)

Advances in Visual Computing

Third International Symposium, ISVC 2007
Lake Tahoe, NV, USA, November 26-28, 2007
Proceedings, Part I

Springer

Volume Editors

George Bebis, E-mail: bebis@cse.unr.edu

Richard Boyle, E-mail: Richard.Boyle@nasa.gov

Bahram Parvin, E-mail: parvin@hpcrd.lbl.gov

Darko Koracin, E-mail: darko@dri.edu

Nikos Paragios, E-mail: nikos.paragios@ecp.fr

Syeda-Mahmood Tanveer, E-mail: stf@almaden.ibm.com

Tao Ju, E-mail: taoju@cs.wustl.edu

Zicheng Liu, E-mail: zliu@microsoft.com

Sabine Coquillart, E-mail: Sabine.Coquillart@inria.fr

Carolina Cruz-Neira, E-mail: carolina@lite3d.com

Torsten Müller, E-mail: torsten@cs.sfu.ca

Tom Malzbender, E-mail: malzbend@hpl.hp.com

Library of Congress Control Number: 2007939401

CR Subject Classification (1998): I.4, I.5, I.2.10, I.3.3, I.3.5, I.3.7, I.2.6, F.2.2

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

ISSN 0302-9743
ISBN-10 3-540-76857-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-76857-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12192953 06/3180 5 4 3 2 1 0

Preface

It is with great pleasure that we welcome you to the Proceedings of the 3rd International Symposium on Visual Computing (ISVC 2007) held in Lake Tahoe, Nevada/California. ISVC offers a common umbrella for the four main areas of visual computing including vision, graphics, visualization, and virtual reality. Its goal is to provide a forum for researchers, scientists, engineers and practitioners throughout the world to present their latest research findings, ideas, developments, and applications in the broader area of visual computing.

This year, the program consisted of 14 oral sessions, 1 poster session, 6 special tracks, and 6 keynote presentations. Following a very successful ISVC 2006, the response to the call for papers was almost equally strong; we received over 270 submissions for the main symposium from which we accepted 77 papers for oral presentation and 42 papers for poster presentation. Special track papers were solicited separately through the Organizing and Program Committees of each track. A total of 32 papers were accepted for oral presentation and 5 papers for poster presentation in the special tracks.

All papers were reviewed with an emphasis on their potential to contribute to the state of the art in the field. Selection criteria included accuracy and originality of ideas, clarity and significance of results, and presentation quality. The review process was quite rigorous, involving two to three independent blind reviews followed by several days of discussion. During the discussion period we tried to correct anomalies and errors that might have existed in the initial reviews. Despite our efforts, we recognize that some papers worthy of inclusion may have not been included in the program. We offer our sincere apologies to authors whose contributions might have been overlooked.

We wish to thank everybody who submitted their work to ISVC 2007 for review. It was because of their contributions that we succeeded in having a technical program of high scientific quality. In particular, we would like to thank the ISVC 2007 Area Chairs, the organizing institutions (UNR, DRI, LBNL, and NASA Ames), the industrial sponsors (Intel, DigitalPersona, Equinox, Ford, Siemens, Hewlett Packard, MERL, UtopiaCompression), the International Program Committee, the special track organizers and their Program Committees, the keynote speakers, the reviewers, and especially the authors that contributed their work to the symposium. In particular, we would like to thank Siemens, Hewlett Packard, and MERL who kindly offered three “best paper awards” this year.

September 2007 ISVC 2007 Steering Committee and Area Chairs

Organization

ISVC 2007 Steering Committee

Bebis George, University of Nevada, Reno, USA
Boyle Richard, NASA Ames Research Center, USA
Parvin Bahram, Lawrence Berkeley National Laboratory, USA
Koracin Darko, Desert Research Institute, USA

ISVC 2007 Area Chairs

Computer Vision

Paragios Nikos, Ecole Centrale de Paris , France
Syeda-Mahmood Tanveer, IBM Almaden, USA

Computer Graphics

Ju Tao, Washington University, USA
Liu Zicheng, Microsoft Research, USA

Virtual Reality

Coquillart Sabine, INRIA, France
Cruz-Neira Carolina, Louisiana Immersive Technologies Enterprise, USA

Visualization

Müller Torsten, Simon Fraser University, Canada
Malzbender Tom, Hewlett Packard Labs, USA

Publicity

Li Wenjing, STI Medical Systems, USA

Local Arrangements

Veropoulos Kostas, Desert Research Institute, USA

Publications

Wang Junxian, UtopiaCompression, USA

ISVC 2007 Keynote Speakers

Mathieu Desbrun , California Institute of Technology, USA
Kwan-Liu Ma, University of Califirnia, Davis, USA
John Tsotsos, York University, Canada
Mubarak Shah, University of Central Florida, USA
Dimitris Metaxas, Rutgers University, USA
Fatih Porikli, MERL, USA

ISVC 2007 International Program Committee

Computer Vision (Area 1)

Abidi Besma, University of Tennessee, USA
Aggarwal J. K.,University of Texas, Austin, USA
Agouris Peggy, George Mason University, USA
Anagnostopoulos George, Florida Institute of Technology, USA
Argyros Antonis, University of Crete , Greece
Asari Vijayan, Old Dominion University, USA
Basu Anup, University of Alberta, Canada
Bebis George, University of Nevada at Reno, USA
Belyaev Alexander, Max-Planck-Institut fuer Informatik, Germany
Bhatia Sanjiv, University of Missouri-St. Louis, USA
Bioucas Jose, Instituto Superior Técnico, Lisbon, Portugal
Birchfield Stan, Clemson University, USA
Boon Goh Wooi, Nanyang Technological University, Singapore
Bourbakis Nikolaos, Wright State University, USA
Brimkov Valentin, State University of New York, USA
Cavallaro Andrea, Queen Mary, University of London, UK
Chellappa Rama, University of Maryland, USA
Chen Danny, University of Notre Dame, USA
Darbon Jerome, LRDE EPITA, France
Davis James, Ohio State University, USA
Debrunner Christian, Colorado School of Mines, USA
Duan Ye, University of Missouri-Columbia, USA
El-Gammal Ahmed, University of New Jersey, USA
Eng How Lung, Institute for Infocomm Research, Singapore
Erol Ali, Ocali Information Technology, Turkey
Fan Guoliang, Oklahoma State University, USA
Foresti GianLuca, University of Udine, Italy
Gandhi Tarak, University of California at San Diego, USA
Georgescu Bogdan, Siemens, USA
Hammoud Riad, Delphi Corporation, USA
Harville Michael, Hewlett Packard Labs, USA
He Xiangjian, University of Technology, Australia

Jacobs David, University of Maryland, USA
Kamberov George, Stevens Institute of Technology, USA
Kamberova Gerda, Hofstra University, USA
Kakadiaris Ioannis, University of Houston, USA
Kisacanin Branislav, Texas Instruments, USA
Klette Reinhard, Auckland University, New Zeland
Kollias Stefanos, National Technical University of Athens, Greece
Komodakis Nikos, Ecole Centrale de Paris, France
Kuno Yoshinori, Saitama University, Japan
Lee Seong-Whan, Korea University, Korea
Leung Valerie, Kingston University, UK
Li Wenjing, STI Medical Systems, USA
Liu Jianzhuang, The Chinese University of Hong Kong, Hong Kong
Ma Yunqian, Honyewell Labs, USA
Maeder Anthony, CSIRO ICT Centre, Australia
Maltoni Davide, University of Bologna, Italy
Maybank Steve, Birkbeck College, UK
Medioni Gerard, University of Southern California, USA
Metaxas Dimitris, Rutgers University, USA
Miller Ron, Ford Motor Company, USA
Mirmehdi Majid, Bristol University, UK
Monekosso Dorothy, Kingston University, UK
Mueller Klaus, SUNY Stony Brook, USA
Mulligan Jeff, NASA Ames Research Center, USA
Nait-Charif Hammadi, Bournemouth University, UK
Nefian Ara, Intel, USA
Nicolescu Mircea, University of Nevada, Reno, USA
Nixon Mark, University of Southampton, UK
Nolle Lars, The Nottingham Trent University, UK
Ntalianis Klimis, National Technical University of Athens, Greece
Pantic Maja, Imperial College, UK
Papadourakis George, Technological Education Institute, Greece
Papanikolopoulos Nikolaos, University of Minnesota, USA
Parvin Bharam, Lawerence Berkeley National Lab, USA
Pati Peeta Basa, Indian Institute of Science, India
Patras Ioannis, Queen Mary University, London, UK
Petrakis Euripides, Technical University of Crete, Greece
Peyronnet Sylvain, LRDE/EPITA, France
Pitas Ioannis, University of Thessaloniki, Greece
Porikli Fatih, MERL, USA
Prabhakar Salil, DigitalPersona Inc., USA
Qian Gang, Arizona State University, USA
Regazzoni Carlo, University of Genoa, Italy
Remagnino Paolo, Kingston University, UK
Ribeiro Eraldo, Florida Institute of Technology, USA

Ross Arun, West Virginia University, USA
Schaefer Gerald, Aston University, UK
Shi Pengcheng, The Hong Kong University of Science and Technology, Hong Kong
Salgian Andrea, The College of New Jersey, USA
Samir Tamer, Ingersoll Rand Security Technologies, USA
Sarti Augusto, DEI, Politecnico di Milano, Italy
Scalzo Fabien, University of Nevada, Reno, USA
Shah Mubarak, University of Central Florida, USA
Singh Rahul, San Francisco State University, USA
Skurikhin Alexei, Los Alamos National Laboratory, USA
Sturm Peter, INRIA Rhône-Alpes, France
Su Chung-Yen, National Taiwan Normal University, Taiwan
Sugihara Kokichi, University of Tokyo, Japan
Sun Zehang, eTreppid Technologies, USA
Teoh Eam Khwang, Nanyang Technological University, Singapore
Thiran Jean-Philippe, EPFL, Switzerland
Tobin Kenneth, Oak Ridge National Laboratory, USA
Triesch Jochen, Frankfurt Institute for Advanced Studies, Germany
Tsechpenakis Gabriel, University of Miami, USA
Tsotsos John, York University, Canada
Tubaro Stefano, DEI, Politecnico di Milano, Italy
Velastin Sergio, Kingston University London, UK
Veropoulos Kostas, Desert Research Institute, USA
Verri Alessandro, Universita' di Genova, Italy
Wang Song, University of South Carolina, USA
Wang Junxian, UtopiaCompression, USA
Wang Yunhong, Chinese Academy of Sciences, China
Webster Michael, University of Nevada, Reno, USA
Wolff Larry, Equinox Corporation, USA
Wong Kenneth, University of Hong Kong, Hong Kong
Xiang Tao, Queen Mary, University of London, UK
Xu Meihe, University of California at Los Angeles, USA
Yau Wei-Yun, Institute for Infocomm Research, Singapore
Yeasin Mohammed, Memphis University, USA
Yi Lijun, SUNY at Binghampton, USA
Yuan Chunrong, University of Tuebingen, Germany
Zhang Yan, Delphi Corporation, USA

Computer Graphics (Area 2)

Arns Laura, Purdue University, USA
Baciu George, Hong Kong PolyU, Hong Kong
Barneva Reneta, State University of New York, USA
Bartoli Vilanova Anna, Eindhoven University of Technology, Netherlands

Peters Jorg, University of Florida, USA
Qin Hong, State University of New York at Stony Brook, USA
Renner Gabor, Computer and Automation Research Institute, Hungary
Sapidis Nickolas, Aegean University, Greece
Sarfraz Muhammad, King Fahd University of Petroleum and Minerals, Saudi Arabia
Schaefer Scott, Texas A&M University, USA
Sequin Carlo, University of California-Berkeley, USA
Shamir Arik, The Interdisciplinary Center, Herzliya, Israel
Silva Claudio, University of Utah, USA
Snoeyink Jack, University of North Carolina at Chapel Hill, USA
Sourin Alexei, Nanyang Technological University, Singapore
Teschner Matthias, University of Freiburg, Germany
Umlauf Georg, University of Kaiserslautern, Germany
Vinacua Alvar, Universitat Politècnica de Catalunya, Spain
Wald Ingo, University of Utah, USA
Weinkauf Tino, ZIB Berlin, Germany
Wylie Brian, Sandia National Laboratory, USA
Ye Duan, University of Missouri-Columbia, USA
Yin Lijun, Binghamton University, USA
Yuan Xiaoru, University of Minnesota, USA

Virtual Reality (Area 3)

Alcañiz Mariano, Technical University of Valencia, Spain
Arns Laura, Purdue University, USA
Behringer Reinhold, Leeds Metropolitan University UK
Benesa Bedrich, Purdue University, USA
Bilalis Nicholas, Technical University of Crete, Greece
Blach Roland, Fraunhofer Institute for Industrial Engineering, Germany
Boyle Richard, NASA Ames Research Center, USA
Brega Jos Remo Ferreira, UNIVEM, PPGCC, Brazil
Brown Ross, Queensland University of Technology, Australia
Chen Jian, Brown University, USA
Cheng Irene, University of Alberta, Canada
Craig Alan, NCSA University of Illinois at Urbana-Champaign, USA
Cremer Jim, University of Iowa, USA
Crosa Pere Brunet, Universitat Politècnica de Catalunya, Spain
Encarnacao L. Miguel, Imedia Labs, USA
Figueroa Pablo, Universidad de los Andes, Colombia
Froehlich Bernd, University of Weimar, Germany
Geiger Christian, Duesseldorf University of Applied Sciences, Germany
Gupta Satyandra K., University of Maryland, USA
Haller Michael, FH Hagenberg, Austria
Hamza-Lup Felix, Armstrong Atlantic State University, USA

Visualization (Area 4)

Encarnasao James Miguel, Imedia Labs, USA
Ertl Thomas, University of Stuttgart, Germany
Floriani Leila De, University of Maryland, USA
Fujishiro Issei, Tohoku University, Japan
Geiger Christian, Duesseldorf University of Applied Sciences, Germany
Grller Eduard, Vienna University of Technology, Austria
Goebel Randy, University of Alberta, Canada
Hadwiger Helmut Markus, VRVis Research Center, Austria
Hamza-Lup Felix, Armstrong Atlantic State University, USA
Julier Simon J., University College London, UK
Koracin Darko, Desert Research Institute, USA
Liere Robert van, CWI, Netherlands
Lim Ik Soo, University of Wales, UK
Ma Kwan-Liu, University of California-Davis, USA
Maeder Anthony, CSIRO ICT Centre, Australia
Malpica Jose, Alcala University, Spain
Masutani Yoshitaka, The University of Tokyo Hospital, Japan
Melanon Guy, INRIA Futurs and CNRS UMR 5506 LIRMM, France
Monroe Laura, Los Alamos National Lab, USA
Mueller Klaus, SUNY Stony Brook, USA
Paelke Volker, Leibniz Universität Hannover, Germany
Preim Bernhard, Otto-von-Guericke University, Germany
Rabin Robert, University of Wisconsin at Madison, USA
Rhyne Theresa-Marie, North Carolina State University, USA
Rolland Jannick, University of Central Florida, USA
Santhanam Anand, MD Anderson Cancer Center Orlando, USA
Scheuermann Gerik, University of Leipzig, Germany
Shen Han-Wei, Ohio State University, USA
Silva Claudio, University of Utah, USA
Snoeyink Jack, University of North Carolina at Chapel Hill, USA
Sourin Alexei, Nanyang Technological University, Singapore
Theisel Holger, Max-Planck Institut für Informatik, Germany
Thiele Olaf, University of Mannheim, Germany
Tory Melanie, University of Victoria, Canada
Umlauf Georg, University of Kaiserslautern, Germany
Viegas Fernanda, IBM, USA
Viola Ivan, University of Bergen, Norway
Wald Ingo, University of Utah, USA
Wylie Brian, Sandia National Laboratory, USA
Yeasin Mohammed, Memphis University, USA
Yuan Xiaoru, University of Minnesota, USA
Zachmann Gabriel, Clausthal University, Germany
Zhukov Leonid, Caltech, USA

ISVC 2007 Special Tracks

Intelligent Algorithms for Smart Monitoring of Complex Environments

Organizers

Paolo Remagnino, DIRC, Kingston University, UK
How-Lung Eng, IIR, Singapore
Guoliang Fan, Oklahoma State University, USA
Yunqian Ma, Honeywell Labs, USA
Dorothy Monekosso, DIRC, Kingston University, UK
Yau Wei Yun, IIR, Singapore

Object Recognition

Organizers

Andrea Salgian, The College of New Jersey, USA
Fabien Scalzo, University of Nevada, Reno, USA

Program Committee

Boris Epshtein, The Weizmann Institute of Science, Israel
Bastian Leibe, ETH Zurich, Switzerland
Bogdan Matei, Sarnoff Corporation, USA
Raphael Maree, Université de Liège, Belgium
Randal Nelson, University of Rochester, USA
Justus Piater, Université de Liège, Belgium
Nicu Sebe, University of Amsterdam, Netherlands
Bill Triggs, INRIA, France
Tinne Tuytelaars, Katholieke Universiteit Leuven, Belgium

Image Databases

Organizers

Sanjiv K. Bhatia, University of Missouri-St. Louis, USA
Ashok Samal, University of Missouri-St. Louis, USA
Bedrich Benes, Purdue University, USA
Sharlee Climer, Washington University in St. Louis, USA

Algorithms for the Understanding of Dynamics in Complex and Cluttered Scenes

Organizers

Paolo Remagnino, DIRC, Kingston University, UK
Fatih Porikli, MERL, USA
Larry Davis, University of Maryland, USA
Massimo Piccardi, University of Technology Sydney, Australia

Program Committee

Rita Cucchiara, University of Modena, Italy
Gian Luca Foresti, University of Udine, Italy
Yoshinori Kuno, Saitama University, Japan
Mohan Trivedi, University of California, San Diego, USA
Andrea Prati, University of Modena, Italy
Carlo Regazzoni, University of Genoa, Italy
Graeme Jones, Kingston University, UK
Steve Maybank, Birkbeck University of London, UK
Ram Nevatia, University of Southern California, USA
Sergio Velastin, Kingston University, USA
Monique Thonnat, INRIA, Sophia Antipolis, France
Tieniu Tan, National Lab of Pattern Recognition, China
James Ferryman, Reading University, UK
Andrea Cavallaro, Queen Mary, University of London, UK
Klaus Diepold, University of Technology in Munich, Germany

Medical Data Analysis

Organizers

Irene Cheng, University of Alberta, Canada
Guido Gortelazzo, University of Padova, Italy
Kostas Daniilidis, University of Pennsylvania, USA
Pablo Figueroa, Universidad de los Andes, Columbia
Tom Malzbender, Hewlett Packard Lab., USA
Mrinal Mandal, University of Alberta, USA
Lijun Yin, SUNY at Binghamton, USA
Karel Zuiderveld, Vital Images Inc., USA

Program Committee

Walter Bischof, University of Alberta, Canada
Anup Basu, University of Alberta, Canada
Paul Major, University of Alberta, Canada

Tarek El-Bialy, University of Alberta, Canada
Jana Carlos Flores, University of Alberta, Canada
Randy Goebel, University of Alberta, Canada
David Hatcher, DDI Central Corp., USA
Shoo Lee, iCARE, Capital Health, Canada
Jiambo Shi, University of Pennsylvania, USA
Garnette Sutherland, University of Calgary, Canada

Soft Computing in Image Processing and Computer Vision

Organizers

Gerald Schaefer, Nottingham Trent University, UK
Mike Nachtegael, Ghent University, Belgium
Lars Nolle, Nottingham Trent University, UK
Etienne Kerre, Ghent University, Belgium

Additional Reviewers

Leandro A. F. Ferndandes
Erik Murphy-Chutorian
Tarak Gandhi
Florian Mannuss
Daniel Patel
Mark Keck

Organizing Institutions and Sponsors

Table of Contents – Part I

Real-Time 3D Face Tracking with Mutual Information and Active Contours

Giorgio Panin and Alois Knoll

Technical University of Munich, Informatics Faculty,
Boltzmannstrasse 3, 85748 Munich, Germany
{panin,knoll}@in.tum.de

Abstract. We present a markerless real-time, model-based 3D face tracking methodology. The system combines two robust and complimentary op-timization-based strategies, namely active contours and mutual information template matching, in order to obtain real-time performances for full 6dof tracking. First, robust head contour estimation is realized by means of the Contracting Curve Density algorithm, effectively employing local color statistics separation for contour shape optimization. Afterwards, the 3D face template is robustly matched to the underlying image, through fast mutual information optimization. Off-line model building is done using a fast modeling procedure, providing a unique appearance model for each user. Re-initialization criteria are employed in order to obtain a complete and autonomous tracking system.

Keywords: Real-time Face Tracking, Nonlinear Optimization, 3D Template Matching, Mutual Information, Active Contours, Local Statistics Contour Matching, 3D Face Modeling.

1 Introduction

Real-time 3D face tracking is an important problem in computer vision. Several approaches have been proposed and developed with different model definitions, concerning shape, degrees of freedom, use of multiple appearance/shading templates, use of natural facial features, etc.; a careful choice of model complexity is a critical issue for real-time tracking, often forcing to resort to approximate solutions, in terms of the output provided to the end user or to subsequent processing modules.

The focus of this paper concerns fast and reliable $6dof$ face pose estimation and tracking in real-time, based on a hierarchical integration of two robust high-level visual modalities. In order to motivate our approach, we first consider here related state-of-the-art methodologies, from the available literature on the subject.

The system proposed in [1] employs a robust multi-layer fusion of different visual cues in the hierarchical framework named IFA [2], proceeding from coarse to accurate visual modalities, and providing the result from the top-level tracker as output; in this work, simple template and feature point models are used at the high levels.

Other approaches, using natural face features detection and tracking in a monocular setting, have been presented in [3][4]. In paticular, in [3] a 3D face model is fitted by matching features across subsequent frames, with an approach combining RANSAC [5] and Particle Filters [6] under frame-to-frame epipolar constraints. [4] combines on-line and off-line information by matching local features and optimizing a robust least-squares global cost function. Although this approach can provide a better stability, precision and speed, due to the use of local optimization techniques, the joint use of online and offline information may pose additional choices, concerning the overall cost function parameters, and the models required for tracking.

Template-based approaches also show to be well-suited for face tracking tasks, concerning both precision and robustness issues. In [7], shape and appearance parameters are optimized at the same time under a 2D piece-wise affine deformation model; although the 3D pose of the face is not directly provided by the estimation algorithm, it can be subsequently estimated from the set of 2D parameters, at the price of quite complex computations involving a Kalman-filter based methodology [8]. The approach [9], closer to ours, directly employs a full 3D shape and appearance template, with multiple appearance models and robust nonlinear least-squares optimization. The 3D shape of the face is used together with multiple texture models of the user under different light conditions; these informations are provided off-line.

Generally speaking, the need for multiple light/appearance models can constitute a major drawback of most template-based approaches, forcing the user to provide off-line several reference images for the textured model. This motivation leads to consider more general similarity functions for template matching, in order to provide robustness of the system while using a few, or even a single appearance model, during all of the tracking task.

Motivated by the previous considerations, in this paper we propose a novel model-based approach fusing two complimentary visual modalities, namely active contour head tracking and face template matching; both modules are based on fast, local optimization of robust cost functions, and at the same time require off-line a minimal set of modeling operations. The paper is organized as follows: Section 2 describes the framework for providing the textured template. Section 3 states the overall tracking algorithm. Section 4 deals with the solution to head contour fitting, employing a fast implementation of the CCD algorithm, while Section 5 describes robust template matching optimizing mutual information similarity index. Finally, Section 6 provides experimental results obtained from a complete implementation of the proposed system.

2 Face Template Modeling

Off-line, given two photos of the subject to be tracked in front and profile views (Fig. 1), we adapt a generic 3D head model by employing part of the model-fitting methodology given in [10], where a few pre-selected feature lines (e.g. eyes, lips, cheek contour) are marked onto the photos, and a least-squares fitting of the mesh is performed.

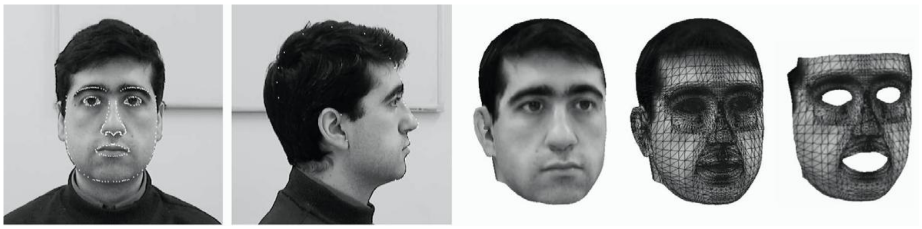


Fig. 1. Automatic 3D model adaptation

After shape adaptation, the same pictures provide the texture map, and the resulting model is shown in Fig. 1 on the left side. From the complete model, deformable parts (eyes and mouth) are removed, and a limited area covering front and part of the side view is extracted, obtaining the textured template shown on the right side.

Almost all of the described operations can be automatically performed, so that with the proposed method a new template is built by a non-expert user within very short time.

3 Overall Tracking Methodology

3.1 Geometry Representation

We express the rigid transformation between camera and head coordinate frames with the homogeneous (4×4) transformation matrix T

$$T = \begin{bmatrix} R & \theta_T \\ \mathbf{0} & 1 \end{bmatrix} \tag{1}$$

with R the rotation matrix, and $\theta_T = [X, Y, Z]^T$ the translation vector. 3D rotations are expressed in terms of XYZ Euler angles

$$R(\theta_R) = R_x(\alpha) R_y(\beta) R_z(\gamma); \theta_R = [\alpha, \beta, \gamma]^T \tag{2}$$

with R_x, R_y, R_z the respective rotation matrices. Overall head pose is then given by the 6-vector θ

$$\theta = [\theta_R, \theta_T] \tag{3}$$

and a body point in homogeneous coordinates ${}^b\bar{\mathbf{x}}$ transforms to camera coordinates ${}^c\bar{\mathbf{x}}$ according to

$${}^c\bar{\mathbf{x}} = T(\theta)^b\bar{\mathbf{x}} \tag{4}$$

Concerning the intrinsic camera parameters, we adopt a simple pinhole model with focal length f, so that points in camera space ${}^c\mathbf{x} = [x_{1c}, x_{2c}, x_{3c}]^T$ project to the screen $\mathbf{y} = [y_1, y_2]^T$ as

$$y_1 = f\frac{x_{1c}}{x_{3c}} + \frac{r_{y1}}{2};\ y_2 = -f\frac{x_{2c}}{x_{3c}} + \frac{r_{y2}}{2} \tag{5}$$

with r_{y1}, r_{y2} the horizontal and vertical image resolution.

By assuming camera calibration to be performed off-line, we can indicate the overall body-to-screen mapping with

$$\mathbf{y} = P(\mathbf{x}, \theta) \tag{6}$$

3.2 On-Line 3D Pose Estimation

In a template matching framework, local optimization of the similarity function alone could be used for estimating the full 6dof pose of the face, starting from a pre-defined hypothesis θ_0.

However, due to the 3D/2D projection (6), any similarity function shows a narrow convergence region, and non-quadratic behavior outside a limited area around the optimum. This results in a difficult optimization problem that may take many cost function evaluations, or get stuck in local optima.

Therefore, as stated in the introduction, in the present approach pose estimation has been split into two sub-tasks, by hierarchically estimating translation and rotation parameters θ_R, θ_T

1. First, translations θ_T are estimated via robust head contour tracking, using the CCD algorithm and an elliptical contour model. This module also provides a coarse estimate of the in-plane rotation angle γ, which can be used for initializing the next module
2. Afterwards, full rotation parameters θ_R are estimated by maximizing mutual information between the 3D textured template and the underlying image; translation parameters θ_T are kept fixed to the value given by the previous module, while rotations are initialized with the ones $\theta_R(t-1)$ obtained from the previous timestep.

Both modules employ suitable critera in order to detect tracking failures, and automatically re-initialize the system in case of loss. In order to re-initialize the contour tracker, a generic face detector [11] is employed; for the template matching module, rotations are re-initialized to zero (front view).

4 Head Contour Tracking Module

The first step of our methodology consists of robustly estimating the translation parameters θ_T. For this purpose, we first model the head external contour as a planar ellipse (Fig. 2) with proper axes length. The degrees of freedom of the contour pose directly correspond to 4 of the 6 pose parameters θ, namely 3D translation and in-plane rotation angle

$$\mathbf{\Phi} = (\gamma, \theta_T) \tag{7}$$

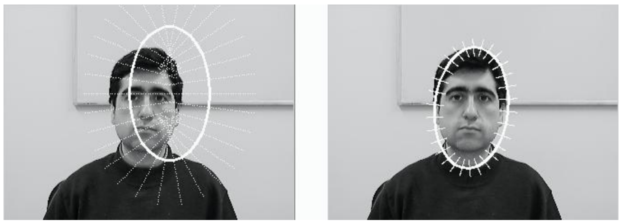


Fig. 2. Contour model fitting to the head boundary, before and after 12 steps of CCD optimization. Sample points used by the algorithm are shown.

Contour tracking is performed through the real-time version of the Contracting Curve Density (CCD) algorithm [12][13]. This method aims at fitting a parametric curve model to the image stream, by making use of local self-adapting separation criteria, and a two-step optimization procedure akin to Expectation Maximization.

Contour estimation at each frame makes also use of prior knowledge of pose parameters in a Bayesian framework, which in the re-initialization phase is obtained from a generic face detector [11], while in subsequent frames comes from the previous estimation result. A dynamic model, although not required, can further improve tracking performance through Kalman filtering [13]; however, for the present tracking task this improvement has shown not to be necessary.

In CCD, the following two-step procedure is iterated, refining the estimate and, at the same time, gaining certainty on the posterior density (that *contracts* to a small, unimodal pdf), until a satisfactory result is found.

4.1 Step 1: Compute Local Color Statistics Around the Head Contour

Current mean and covariance matrix $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ are set to the initial values $(\mathbf{\Phi}_0, \mathbf{\Sigma}^*_{\mathbf{\Phi}})$, with statistics modeled by multivariate Gaussian distributions.

A set of points along normal directions for each sample position k on both sides of the curve is taken (Fig. 2) up to a distance h related to the current parameter uncertainty $\mathbf{\Sigma}_{\mathbf{\Phi}}$, and normal segments to the base shape are uniformly sampled along into L sample points

$$\mathbf{x}_{kl} = \mathbf{x}_{k0} + d_l \mathbf{n}_k \tag{8}$$

with $d_l = d(\mathbf{x}_{kl}, \mathbf{x}_{k0})$ the perpendicular distance.

According to the length h, two weighting functions w_1, w_2 and fuzzy *assignment* values a_1, a_2 of each color pixel to the respective region, are computed for each contour side

$$a_1(d) := \frac{1}{2}\left[erf\left(\frac{d}{\sqrt{2}\sigma}\right)+1\right];\ \ a_2 := 1-a_1 \tag{9}$$
$$w_{1/2}(d) := \left(\frac{a_{1/2}(d)-0.5}{0.5}\right)^6\left[e^{-d^2/2\hat{\sigma}^2}-e^{-\gamma_2}\right]^+$$

where all of the constants inside are properly selected in advance.

In order to collect local color statistics, base sample points $\mathbf{x}_{kl}$ are projected according to the current hypothesis $\mathbf{\Phi} = \mathbf{m}_{\mathbf{\Phi}}$ (Fig. 2) by applying (6)

$$\mathbf{y}_{kl} = P(\mathbf{x}_{kl}, [0, 0, \mathbf{\Phi}]) \tag{10}$$

and local, weighted RGB statistics up to the second order are collected

$$\nu_{ks}^{(0)} = \sum_{l=1}^{L} w_{kls};\ \nu_{ks}^{(1)} = \sum_{l=1}^{L} w_{kls}\mathbf{I}_{kl};\ \nu_{ks}^{(2)} = \sum_{l=1}^{L} w_{kls}\mathbf{I}_{kl}\mathbf{I}_{kl}^T \tag{11}$$

with $s = 1, 2$ the curve side, and $\mathbf{I}_{kl}$ the observed pixel colors at image locations $\mathbf{y}_{kl}$.

Color statistics of order $o = 0, 1, 2$ along the contour are blurred with exponential filtering both in space and time

$$\nu_{ks}^{(o)} = \sum_{k_1=1}^{K} b(k-k_1)\,\nu_{k_1 s}^{(o)} \tag{12}$$
$$\widetilde{\nu}_{ks}^{(o)}(t) = \tau\nu_{ks}^{(o)} + (1-\tau)\,\widetilde{\nu}_{ks}^{(o)}(t-1)$$

and the resulting quantities are finally normalized, giving a set of expected color values and covariance matrices for each sample position and contour side

$$\bar{\mathbf{I}}_k^{(s)} = \frac{\widetilde{\nu}_{ks}^{(1)}(t)}{\widetilde{\nu}_{ks}^{(0)}(t)};\qquad \bar{\mathbf{\Sigma}}_k^{(s)} = \frac{\widetilde{\nu}_{ks}^{(2)}(t)}{\widetilde{\nu}_{ks}^{(0)}(t)} - \bar{\mathbf{I}}_k^{(s)}\bar{\mathbf{I}}_k^{(s)T} \tag{13}$$

Normalized local statistics are finally indicated with $S := \left(\bar{\mathbf{I}}_k^{(s)}, \bar{\mathbf{\Sigma}}_k^{(s)}\right)$.

4.2 Step 2: Optimize Contour Shape

In the second step of CCD, estimated statistics (13) are used in order to evaluate the contour fit function, its gradient and the Hessian matrix for parameter update. We first write the MAP cost function as

$$E = E_1(\mathbf{\Phi}, \mathbf{m}_{\mathbf{\Phi}}^*, \mathbf{\Sigma}_{\mathbf{\Phi}}^*) + E_2(\mathbf{\Phi}, S) \tag{14}$$

where

$$E_1 := \frac{1}{2}\left(\mathbf{\Phi} - \mathbf{m}_{\mathbf{\Phi}}^*\right)^T \left(\mathbf{\Sigma}_{\mathbf{\Phi}}^*\right)^{-1} \left(\mathbf{\Phi} - \mathbf{m}_{\mathbf{\Phi}}^*\right) + Z\left(\mathbf{\Sigma}_{\mathbf{\Phi}}^*\right) \tag{15}$$
$$E_2 := \sum_{k,l}\left[\frac{1}{2}\left(\mathbf{I}_{kl} - \hat{\mathbf{I}}_{kl}\right)^T \hat{\mathbf{\Sigma}}_{kl}^{-1}\left(\mathbf{I}_{kl} - \hat{\mathbf{I}}_{kl}\right) + Z\left(\hat{\mathbf{\Sigma}}_{kl}\right)\right]$$

are, respectively, the prior log-probability of the pose, and the log-likelihood of observed colors $\mathbf{I}_{kl}$ w.r.t. the expected statistics $\left(\hat{\mathbf{I}}_{kl}, \hat{\mathbf{\Sigma}}_{kl}\right)$, with normalizing terms Z.

Expected colors $\hat{\mathbf{I}}_{kl}$ and local covariances $\hat{\mathbf{\Sigma}}_{kl}$ are given by the fuzzy assignment (9) to each contour side

$$\hat{\mathbf{I}}_{kl} = a_1(d_l)\,\bar{\mathbf{I}}_k^{(1)} + (1 - a_1(d_l))\,\bar{\mathbf{I}}_k^{(2)} \tag{16}$$
$$\hat{\mathbf{\Sigma}}_{kl} = a_1(d_l)\,\bar{\mathbf{\Sigma}}_k^{(1)} + (1 - a_1(d_l))\,\bar{\mathbf{\Sigma}}_k^{(2)}$$

where only the dependence of $\hat{\mathbf{I}}_{kl}$ on $\mathbf{\Phi}$ is considered when computing the derivatives.

In order to implement the Gauss-Newton optimization step, the gradient of E w.r.t. $\mathbf{\Phi}$ and the Gauss-Newton matrix are computed

$$\nabla_{\mathbf{\Phi}} E_2 = -\sum_{k,l}\left(\nabla_{\mathbf{\Phi}}\hat{\mathbf{I}}_{kl}\right)^T \hat{\mathbf{\Sigma}}_{kl}^{-1}\left(\mathbf{I}_{kl} - \hat{\mathbf{I}}_{kl}\right) + \left(\mathbf{\Sigma}_{\mathbf{\Phi}}^*\right)^{-1}\left(\mathbf{\Phi} - \mathbf{m}_{\mathbf{\Phi}}^*\right) \tag{17}$$
$$H_{\mathbf{\Phi}} E_2 = \sum_{k,l}\left(\nabla_{\mathbf{\Phi}}\hat{\mathbf{I}}_{kl}\right)^T \hat{\mathbf{\Sigma}}_{kl}^{-1}\nabla_{\mathbf{\Phi}}\hat{\mathbf{I}}_{kl} + \left(\mathbf{\Sigma}_{\mathbf{\Phi}}^*\right)^{-1}$$

with

$$\nabla_{\mathbf{\Phi}}\hat{\mathbf{I}}_{kl} := \left(\bar{\mathbf{I}}_k^{(1)} - \bar{\mathbf{I}}_k^{(2)}\right)\left(\nabla_{\mathbf{\Phi}} a_1(d_l)\right)^T \tag{18}$$

and the parameter mean and covariance update is given by

$$\mathbf{m}_{\mathbf{\Phi}}^{new} = \mathbf{m}_{\mathbf{\Phi}} - \left(H_{\mathbf{\Phi}} E\right)^{-1}\nabla_{\mathbf{\Phi}} E \tag{19}$$
$$\mathbf{\Sigma}_{\mathbf{\Phi}}^{new} = c\mathbf{\Sigma}_{\mathbf{\Phi}} + (1 - c)\left(H_{\mathbf{\Phi}} E\right)^{-1}$$

where $c = 0.25$ is an exponential decay coefficient.

The whole procedure (Steps 1 and 2) is iterated with updated parameters until convergence. In this module, detection for tracking loss is performed by looking at the final value of E (14), indicating the overall color statistics separation between the two regions (inside and outside) along the contour line. If the separation index decreases below a suitable threshold, the CCD tracker is re-initialized.

5 Face Template Matching Module

With the given appearance model, the template matching module aims at finding the orientation θ_R maximizing *consistency* between projected template points and underlying image pixels, as shown in Fig. 3.

Using mutual information as consistency measure allows to accommodate a more general model of variation between expected and observed appearance, in presence of different lighting, shading, and other nonlinear effects such as partial occlusions. As a result, by using mutual information we can keep a constant appearance model working through a wide range of rotations, at the same time gaining robustness with respect to occlusions and image noise.

For this task, we indicate with $M(\mathbf{x})$ the gray-scale intensity of points $\mathbf{x}$ over the textured surface model of Fig. 1.

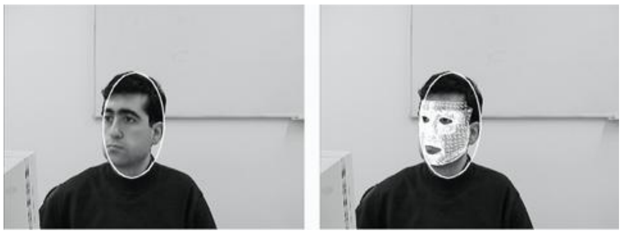


Fig. 3. Template matching result after mutual information optimization

5.1 Histogram-Based Estimator

For real-time purposes, we use histogram-based mutual information [14]. Histograms have large computational advantages over other methods like as kernel density estimators [15], which generally exhibit an $O(N^2)$ complexity and make use of costly real-valued functions (e.g. exponential kernels). For this reason, histogram-based entropy estimators are widely used in medical image registration problems, where the volume of data is usually very large; on the other hand, in a real-time application sample sizes have to be kept relatively small, and therefore the finite-sample *bias* of the estimator cannot be anymore neglected.

An analysis of statistical properties (bias, variance) of the histogram entropy estimator for continuous densities has been given in [14]; in particular, the bias contribution due to finite sample size N (termed *N-bias*) can be approximated by a second-order Taylor expansion of the density estimator around the "true" value p_{ij}, and the resulting correction for the joint entropy $\widehat{H}(x_1, x_2)$ is given by

$$\widehat{H}(x_1, x_2) = -\sum_{i,j} \left(\frac{h_{ij}}{N} \log \frac{h_{ij}}{N} \right) + \frac{C^2 - 1}{2N} + \log(\Delta x_1 \Delta x_2) \qquad (20)$$

where h_{ij} is the joint (template-image) histogram, with C^2 cells of size $(\Delta x_1, \Delta x_2)$.

In order to get a good statistical behaviour, a suitable number of histogram cells is required, for which we adopt a simple rule of thumb, by dividing each dimension into $I = J = \sqrt[3]{N}$ bins.

It is worth to note here that both bias and variance of $\widehat{H}(\mathbf{x})$, like for any statistical estimator, grow up very fast with space dimension, so that the sample size N required to achieve the same performance in higher dimension (e.g. using a color template) would be too high for a real-time implementation, and at the same time not necessary. By using grayscale intensities, a bi-variate histogram allows a minimum sample size of $N = 1000$ sample points to be reliably employed.

5.2 Mutual Information Optimization

In our framework, a set of N points $\mathbf{x}_k$ is randomly drawn with uniform probability from the visible area of the template M at pose $\theta_0 = \left[\theta_R^0, \theta_T^0\right]$; the set of visible points is pre-computed, by removing back and self-occluded surface points[1].

Because of random sampling, the cost function will have a random component (noisy optimization); this indeed helps in avoiding local optima that for a small size N may be present.

For furtherly improving convergence and speed properties, the optimization is performed in a multi-resolution cascade, by Gaussian filtering both the model texture map and the underlying image; moreover, the sample set N is also kept lower for low resolution images, while increasing it for higher resolutions.

At each pose hypothesis θ_R, the random sample $\mathbf{x}_k$ is projected onto the image I according to (6), to pixel coordinates $\mathbf{y}_k = P(\mathbf{x}_k, \theta)$, and the corresponding intensity pairs $(m_1, v_1), ..., (m_N, v_N)$, with

$$m_k = M(\mathbf{x}_k); v_k = I(\mathbf{y}_k) \tag{21}$$

are collected into the joint histogram $h_{ij} = h(m, v)$.

Afterwards, marginal histograms $h(m)$ and $h(v)$ are computed by summation over rows and columns

$$h(m) \triangleq h_{i,\bullet} = \textstyle\sum_{j=1}^{C} h_{ij};\ h(v) = h_{\bullet,j} = \sum_{i=1}^{C} h_{ij} \tag{22}$$

and bias-corrected mutual information is finally given by

$$\widehat{I}_{m,v}\left(x_1, ..., x_N, \theta_T^0, \theta_R\right) = \sum_{i,j} \frac{h_{ij}}{N} \log \frac{h_{ij} N}{h_{i,\bullet} h_{\bullet,j}} - \frac{(C-1)^2}{2N} \tag{23}$$

In order to optimize (23) w.r.t. θ_R, we apply here the standard Nelder-Mead *simplex* algorithm [16]. This algorithm is well-suited for low dimensional problems, and at the same time performs well in presence of a noisy cost function.

[1] Since the previous pose θ_R^0 is close to the current one, the visible point set can be computed just once, before the optimization loop.

After optimization, tracking loss is detected by setting a lower threshold on the output mutual information value. By optimizing only over rotational degrees of freedom θ_R, this module yields stable and reliable results within less than 100 function evaluations, ensuring real-time performances on common platforms.

6 Experimental Results

Our face modeling and tracking system has been successfully applied to on-line tracking experiments. We show here results from a video sequence of 1550 frames, using the model of Fig. 1; in absence of ground-truth for this experiment, accuracy of the result can be visually evaluated by looking at the matching between superimposed template and image pixels (Fig. 4).

Throughout all of the sequence, tracking has been performed in real-time at a frame rate of 10-15 fps on a common Desktop PC with 3GHz CPU, using a standard FireWire camera with an image resolution of 640x480; due to the nature of both estimation modules, the input images are directly used without the need for any preprocessing operation (noise filtering, features enhancement etc.); results have been on-line recorded during the tracking task.

Fig. 4 shows a selection of frames, with the results of both contour tracking and template matching, and the frontal part of the textured model superimposed at estimated 3D pose. A single appearance template has been sufficient for successfully tracking the user throughout almost the whole sequence, apart from two cases of too wide lateral rotations; after loosing the track, the system

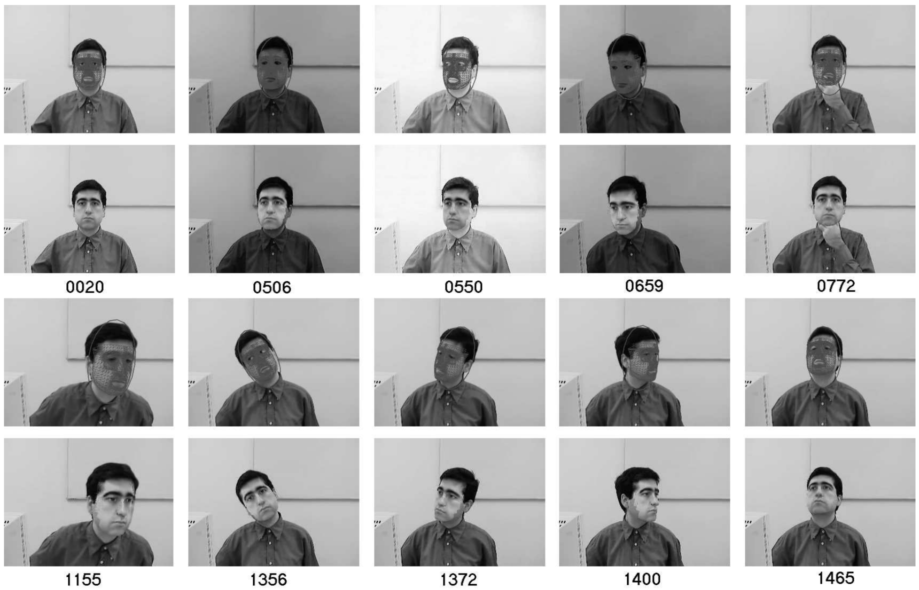


Fig. 4. Face tracking in presence of different light conditions, partial occlusion and wide rotation angles

automatically recovered within few frames, without the need for any manual intervention.

In particular, the top rows of Fig. 4 show robustness of the system under strong light changes and partial occlusion; thanks to the use of mutual information as robust matching index, the tracker did not loose accuracy of pose estimation despite dark lighting (frames 506, 659), light switching (frame 550), and occlusions (frame 772). The same observation concerns contour tracking that, by employing locally self-adaptive color separation criteria, is able to overcome abrupt changes in color statistics both inside and outside the enclosed region.

The frames displayed in the bottom rows show tracking performance under wider face rotations; wide in-depth rotations (frame 1400) usually pose more difficult problems with respect to in-plane rotations (frame 1356) for face tracking systems, in particular due to the shading difference with respect to the frontal pose, and the smaller visible template area. Despite these difficulties, mutual information template matching performs well also in these cases, where different shadings have the effect of reducing the maximum similarity value, but not its location in pose space.

7 Conclusions

We presented a real-time 3D face tracking system combining two complimentary and robust model-based visual modalities; possible applications for the system include hands-free Human-Computer Interfaces, 3D gaze tracking, real-time video augmentation, and many others as well.

Planned developments, in the direction of increasing speed, robustness and versatility, include the following ones: augmenting the system with further visual modalities such as color and motion cues; a parallel (or pipeline) implementation scheme for an efficient integration of all modules; exploration of suitable data fusion methodologies for this system, incorporating dynamic models and multiple hypotheses for multi-target tracking in presence of crowded situations; introduction of deformable models on top of the rigid template, in order to detect lips expression and eye gaze.

References

1. Toyama, K.: Look, ma — no hands!' hands-free cursor control with real-time 3d face tracking (1998)
2. Toyama, K., Hager, G.: Incremental focus of attention for robust visual tracking (1996)
3. Lu, L., Dai, X.T., Hager, G.: A particle filter without dynamics for robust 3d face tracking. In: CVPRW 2004: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2004), vol. 5, p. 70. IEEE Computer Society, Washington, DC, USA (2004)
4. Vacchetti, L., Lepetit, V., Fua, P.: Stable real-time 3d tracking using online and offline information. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 1391–1391 (2004)

5. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 381–395 (1981)
6. Isard, M., Blake, A.: Condensation – conditional density propagation for visual tracking. International Journal of Computer Vision 29, 5–28 (1998)
7. Matthews, I., Baker, S.: Active appearance models revisited. Technical Report CMU-RI-TR-03-02, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2003)
8. Xiao, J., Baker, S., Matthews, I., Kanade, T.: Real-time combined 2d+3d active appearance models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 535–542 (2004)
9. Cascia, M., Sclaroff, S., Athitsos, V.: Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models (1999)
10. Park, I.K., Zhang, H., Vezhnevets, V., Choh, H.K.: Image-based photorealistic 3-d face modeling. In: FGR, pp. 49–56 (2004)
11. Viola, P.A., Jones, M.J.: Robust real-time face detection. In: ICCV, p. 747 (2001)
12. Hanek, R., Beetz, M.: The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. Int. J. Comput. Vision 59, 233–258 (2004)
13. Panin, G., Ladikos, A., Knoll, A.: An efficient and robust real-time contour tracking system. In: ICVS, p. 44 (2006)
14. Moddemeijer, R.: On estimation of entropy and mutual information of continuous distributions. Signal Processing 16, 233–246 (1989)
15. Viola, P., William, M., Wells, I.: Alignment by maximization of mutual information. Int. J. Comput. Vision 24, 137–154 (1997)
16. Nelder, J., Mead, R.: A simplex method for function minimization. Computer Journal 7, 308–313 (1965)

Robust Infants Face Tracking Using Active Appearance Models: A Mixed-State CONDENSATION Approach

Luigi Bagnato[1], Matteo Sorci[1], Gianluca Antonini[1], Giuseppe Baruffa[3], Andrea Maier[2], Peter Leathwood[2], and Jean-Philippe Thiran[1]

[1] Ecole Polytechnique Federale de Lausanne, Signal Processing Institute
Ecublens, 1015 Lausanne, Switzerland
{Luigi.Bagnato,Matteo.Sorci,Gianluca.Antonini,JP.Thiran}@epfl.ch
[2] Nestlé Research Center (CRN), Food-Consumer Interactions Department, Lausanne, Switzerland
{Andrea.Maier,Peter.Leathwood}@rdls.nestle.com
[3] University of Perugia,Dept.of Electronic and Information Engineering
06125 Perugia, Italy
baruffa@diei.unipg.it

Abstract. In this paper a new extension of the CONDENSATION algorithm, with application to infants face tracking, will be introduced. In this work we address the problem of tracking a face and its features in baby video sequences. A mixed state particle filtering scheme is proposed, where the distribution of observations is derived from an active appearance model. The mixed state approach combines several dynamic models in order to account for different occlusion situations. Experiments on real video show that the proposed approach augments the tracker robustness to occlusions while maintaining the computational time competitive.

1 Introduction

The tracking of the face motion in a video sequence represents a challenging task in computer vision, because of the variability of facial appearance in real scenes, most notably due to changes in head pose, expressions, lighting or occlusions. This is especially challenging when an infant face is the tracking target. This task requires, by definition, the use of a model that describes the expected structure of the face. The Active Appearance Model (AAM) [1] is one of such techniques, which elegantly combines shape and texture models in a statistical framework, providing as output a mask of face landmarks. These combined models account for all sources of variability in face images. This feature makes them suitable for face tracking and enables the tracking of both global motion and inner features. Previous work on visual tracking can be divided in two groups: deterministic tracking and stochastic tracking. Deterministic approaches [2] usually reduce to an optimization problem, i.e. minimizing an appropriate cost function, while stochastic tracking approaches often reduce to the estimation of the state for a

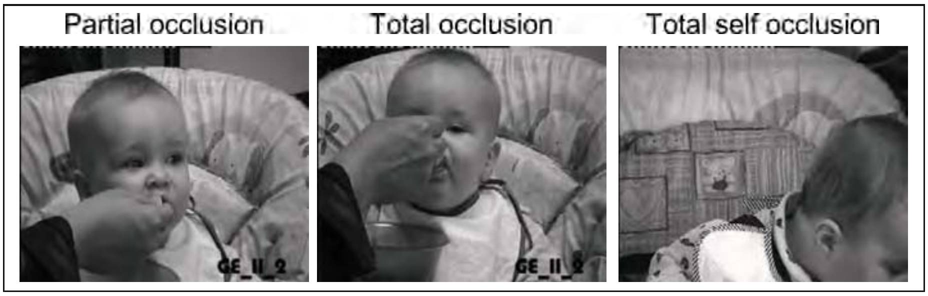


Fig. 1. Example images from input video sequences

time series state space model. Stochastic tracking improves robustness over its deterministic counterpart, thanks to its capability to escape from local minimum since the search directions are for the most part random. Early approaches used Kalman filter (or its variants [3]) to provide solutions, while, recently, sequential Monte Carlo algorithms [4] have gained prevalence in the tracking literature, especially due to the CONDENSATION algorithm [5].

Our work is based on a direct combination of an AAM with a particle filter as first introduced by Hamlaoui [6]. In this approach the authors combine an AAM with a temporal dynamics guided by the AAM search algorithm and use a filtering scheme based on CONDENSATION. Although their stochastic approach allows to augment robustness, they rely too much on the deterministic AAM search and the resulting algorithm performs poorly in case of heavy occlusions. Our contribution consists in a customized version of the mixed-state algorithm to face the particular problem of infants face tracking.

In Figure 1, some example images show inherent difficulties when dealing with real video sequences of infants. There are two major elements, which add complexity to the tracking task: infants move continuously and in an unpredictable way, producing face self-occlusions as most undesirable effect; external objects (a hand in Figure 1) may occlude the target face either partially or totally. In this paper we propose a technique for a robust tracker of a single infant face in a video sequence with the following properties: high responsiveness to sudden movements, robustness to partial occlusions, short recovery period after distraction due to a total occlusion. Our approach, based on the CONDENSATION algorithm, integrates, in a Bayesian mixed-state framework, multiple dynamic models, allowing to cope with the limitations of previous approaches.

The rest of the paper is organised as follows: in Section 2 we briefly review the Active Appearance Model and we introduce the AAM-based CONDENSATION framework. Section 3 describes in detail our proposed mixed-state approach, while Section 4 is dedicated to the experimental results. Conclusions and future works are then reported in Section 5.

2 Background

2.1 Face Active Appearance Model

The AAM is a statistical method for matching a combined model of shape and texture to unseen faces. The combination of a model of shape variations with a model of texture variations generates a statistical appearance model. Principal Component Analysis (PCA) is applied to build the statistical shape and texture models:

$$\mathbf{s} = \bar{s} + \Phi_s b_s \quad and \quad \mathbf{g} = \bar{g} + \Phi_t b_t \tag{1}$$

where $\bar{s}$ and $\bar{g}$ are the mean shape and texture, Φ_s and Φ_t are the eigenvectors of shape and texture covariance matrices. The unification of the presented shape and texture models into one complete appearance model is obtained by concatenating the vectors b_s and b_t and learning the correlations between them by means of a further PCA:

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{Q_s c} \quad and \quad \mathbf{g} = \bar{\mathbf{g}} + \mathbf{Q_t c} \tag{2}$$

where Q_s and Q_t are the matrices describing the principal modes of the combined variations in the training set and **c** is the appearance parameters vector. By varying the appearance parameters **c**, new instances of shape and texture can be generated. The matching of the appearance model to a target face can be treated as an optimization problem, minimizing the difference between the synthesized model image and the target face [1].

2.2 AAM-Based CONDENSATION

The CONDENSATION is a Monte Carlo-type technique to recursively approximate the posterior state density. Approximation is done by means of the empirical distribution of a system of particles. The particles explore the state space following independent realizations from a state dynamic model, and are redistributed according to their consistency with the observations, the consistency being measured by a likelihood function (observation model). For an introduction to the subject the reader is referred to the Isard and Blake paper [5]. Using Active Appearance Models, we can represent the shape and texture of a face in terms of a vector **c**. To complete the description of the face, the four pose parameters are also needed, namely $\mathbf{p} = (\alpha, \vartheta, t_x, t_y)$, representing scale, orientation and position, respectively. The state vector **x**, which contains the parameters used to infer about the object (the face) is thus composed by the concatenation of the vector **c** of combined parameters and vector **p** of pose.

Observation model. The Observation Model is based on the difference between the sampled pixel grey level patch at the hypothesized position in the current image and the one generated by the face model. The likelihood function $p(\mathbf{y}_k|\mathbf{x}_k)$ denotes the probability that a hypothesized state $\mathbf{x}_k = (\mathbf{c}_k, \mathbf{p}_k)$ gives rise to the observed data. Since the observed data consist of pixel greylevel values, it is straightforward to look for a function with the following form:

$$p(\mathbf{y}_k|\mathbf{x}_k) = p(\mathbf{y}_k|\mathbf{c}_k, \mathbf{p}_k) = C \exp\left(-d[\mathbf{g}_{model}(\mathbf{c}_k), \mathbf{g}_{image}(\mathbf{c}_k, \mathbf{p}_k)]\right) \tag{3}$$

where $\mathbf{g}_{image}(\mathbf{c}_k, \mathbf{p}_k)$ is the image patch sampled at the hypothesized pose and shape, $\mathbf{g}_{model}(\mathbf{c}_k)$ is the model texture representing the hypothesized appearance of the face, and C is a normalizing constant. The texture distance $d[;]$ is an error measure, summed over all L pixels of both textures.

State transition model. The state transition model characterizes the dynamics between frames. The state evolves according to

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{S}_k\mathbf{u}; \tag{4}$$

in this equation, $\mathbf{f}(\mathbf{x}_{k-1})$ represents a deterministic function of the previous state vector $\mathbf{x}_{k-1}$, $\mathbf{S}_k$ is the process noise covariance and $\mathbf{u}$ is a vector of normally distributed random variables.

Whereas the choice of gaussian noise is expected when accurate model uncertainty cannot be provided, choosing an appropriate function $\mathbf{f}(.)$ is not an easy task, and depends on the particular situation.

3 Proposed Scheme: A Mixed State CONDENSATION

In the case of this particular application the presence of heavy occlusions and the unpredictable infant movements make the choice of a single model inadequate to describe the dynamics. In this spirit we opted for a mixed-state framework with model switching.

3.1 Pose-CONDENSATION and ICONDENSATION

The first two retained models consist in a fixed constant-velocity model with fixed noise variance

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{S}_k\mathbf{u}; \tag{5}$$

and an adaptive dynamic model, guided by a deterministic AAM search:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta\mathbf{x}_{k-1,k} + \mathbf{S}_k\mathbf{u}, \tag{6}$$

where $\Delta\mathbf{x}_{k-1,k} = (\Delta\mathbf{c}_{k-1,k}, \Delta\mathbf{p}_{k-1,k})$ is the predicted shift in pose and appearance parameters, obtained by applying the AAM search to previous estimated state vector $(\mathbf{x}_{k-1})$ with respect to frame at time k. An AAM-CONDENSATION approach, using either (5) or (6) as dynamic models, has three main drawbacks: AAMs describe faces with high-dimensional vectors; the deterministic AAM search is highly sensitive to large occlusions, so robustness is achieved only by increasing the noise variance; accuracy is only accomplished at the cost of unacceptable time performance. In order to solve the problem of dimensionality while assuring a good robustness to occlusions, Davoine [7] proposed the use of the CONDENSATION to track the four pose parameters (in the following this algorithm is referred as Pose-CONDENSATION). In this case (5) is a convenient form for the dynamic model with the state vector represented by the pose vector. Obviously, the accuracy of the algorithm, intended as the ability to generate a photo realistic synthetic replica of the face, cannot be guaranteed since the appearance parameters, $\mathbf{c}$, are not tracked. In order to track the entire state vector $\mathbf{x} = (\mathbf{c}, \mathbf{p})$ and keep the computational time acceptable, a solution could be to use an importance function, as described in the ICONDENSATION framework [8], to constrain the search in a neighbourhood of the previous state estimate. In this case a proper dynamics is the one described in (6). This approach performs well, in terms of speed and accuracy, only in presence of soft partial occlusions: the deterministic search is inherently not robust and the main feature of CONDENSATION, i.e. maintaining multiple hypothesis, is constrained in a limited regionof the state space by the importance function. The two described approaches are complementary: the first one sacrifices accuracy for robustness, while the second one allows fast and accurate tracking, but only in occlusions free situations. The integration of both dynamics into the same tracker would then allow for a wider range of motion to be supported without losing the advantages of an accurate prediction. In summary, we can say that in case of limited occlusions, the deterministic AAM search should be reasonably trusted, leaving to the importance sampling technique the task to improve the accuracy of the detection. In the second scenario, when strong occlusions occur, a less accurate tracker but more robust to occlusions should be preferred.

3.2 The Third Model

In the proposed solution, a third motion model from data averaging has been included and defined by:

$$\mathbf{x}_k = \bar{\mathbf{x}}_k + \mathbf{S}_k \mathbf{u} \tag{7a}$$

$$p(\mathbf{x}_k) = N(\bar{\mathbf{x}}_{k-1}, \mathbf{S}_k), \tag{7b}$$

where $\bar{\mathbf{x}}_\mathbf{k}$ is an estimate of the (fixed) *mean vector* $\bar{\mathbf{x}}$ and $\mathbf{S}_k$ is the covariance matrix of the gaussian distribution at time k. The aim of this model is to describe a kind of a priori knowledge on the face motion of the considered sequences. In each video sequence the infants are sitting, thus their movements are somehow constrained around a region in the scene. We can then assume that the face lies in a neighbourhood of such region, with probability decreasing with the distance. This third motion model (in the following referred as Reinitialization) can be used to include some probability of tracking reinitialization, particularly useful after distraction due, for example, to total occlusions.

3.3 The Mixed-State Algorithm

Our solution, then, consists in merging the three approaches by means of an automatic model switching procedure ([9]). The extended state is defined as

$\mathbf{X} = (\mathbf{x}, \theta),\ \theta \in 1, .., N$,where θ is a discrete variable labelling the current model, while N represents the total number of models. The process density can then be decomposed as follows:

$$p(\mathbf{X}_k|\mathbf{X}_{k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}, \theta_k)P(\theta_k|\theta_{k-1}, \mathbf{x}_{k-1}) \tag{8}$$

where $P(\theta_k|\theta_{k-1}, \mathbf{x}_{k-1}) : P(\theta_k = j|\theta_{k-1} = i, \mathbf{x}_{k-1}) = T_{ij}(\mathbf{x}_{k-1})$ and the T_{ij} are *state transition probabilities*. The continuous motion models for each transition are given by the *sub-process densities* $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \theta_k)$. During tracking, each discrete state transition with non zero probability contributes some samples to the state distribution and the model that predicts more accurately than the others will dominate. The possible values of θ_k are: R (Reinitialization), I (ICONDENSATION), P (Pose-CONDENSATION). The particle is propagated forward in time according to the dynamics implied by the motion model, and transitions between models happen according to the transition matrix $\mathbf{T}$. We propose the following simple form for the transition matrix where the parameters α and δ control the robustness of the tracker:

$$\mathbf{T} = \begin{pmatrix} T_{RR} & T_{RI} & T_{RP} \\ T_{IR} & T_{II} & T_{IP} \\ T_{PR} & T_{PI} & T_{PP} \end{pmatrix} = \begin{pmatrix} \alpha & 1-\alpha-\delta & \delta \\ \alpha & 1-\alpha-\delta & \delta \\ \alpha & 1-\alpha-\delta & \delta \end{pmatrix} \tag{9}$$

– α is a *reinitialization* parameter, since, at each time step, a number of particles proportional to α is generated from model (7).
– The meaning of δ changes, instead, between the first and the second representation: in the first case it represents an *adaption speed* parameter, that controls how rapidly the probability flows from the model I to the model P. Thus, we can say that it trades off adaptation rate and steady state behavior. In the second case, we give it the meaning of *robustness* parameter, controlling how promptly the tracker switches into pose tracking.

Summarizing, at each time step k, the proposed algorithm chooses a particle from the previous sample set, proportionally to its weight. The particle is then propagated through one of the three dynamic models, in accordance with the current motion label. If, for example, a particle is chosen with label I, then with probability T_{II} a particle is drawn from the importance function $q(\hat{\mathbf{x}}_{k-1})$, with probability T_{IR} it is drawn from (7), and with probability T_{IP} it is propagated through (5) (where $x \equiv p$). Finally, the particle is weighted in accordance to observation, and the multiplicative factor f/q is applied if it was generated with Importance Sampling.

4 Experimental Results

The implementation of the tracker is based on AAM-API, a C++ implementation of Active Appearance Model. In order to build our AAM representation of face, we have manually landmarked a set of 222 images. To test the Mixed State CONDENSATION tracker we used 50 video sequences. For all of them a visual

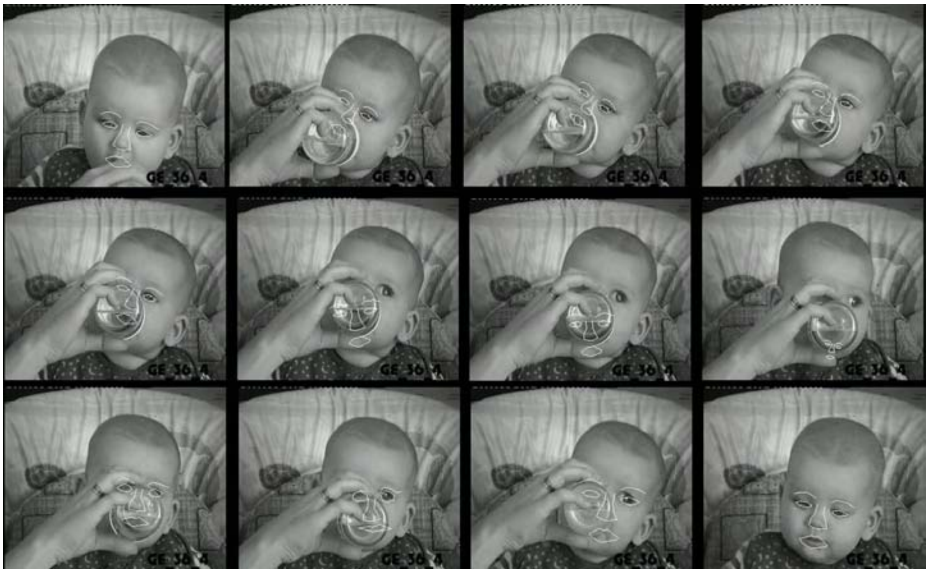


Fig. 2. Tracking results for consecutive frames

analysis has been done, resulting in good overall performance of the proposed tracker. Figure 2 shows consecutive frames of an example sequence characterized by a total occlusion. Given that the videos used in the experiments are under corporate proprietary rights, only a small percent of them can be published. In Figure 3, 4 and 5 we report the results applying the three methods to three different sequences. In the image grid, each column represents a tracked frame from the sequence, while each row is related to one of the 3 compared approaches. Below the image grid, a plot shows the tracking error for the entire sequences.

The results show that, although the Pose-CONDENSATION tracker is quite robust to occlusions, the tracking error is still very high, since the inner motion of the face is not tracked. The behaviour of the ICONDENSATION is exactly the opposite: it tracks well as long as the target is not occluded, but it is not able to recover the target face once distracted. The mixed-state CONDENSATION offers a good trade-off: it automatically switches model, choosing the best for each situation. It reveals high robustness and the best accuracy, from our analysis. Concerning the α and δ parameters of the transition matrix $\mathbf{T}$, their values have been empirically chosen in order to adapt the algorithm to our task. Table 1 reports the time performance of the different algorithms for 3 representative sequences. The results are obtained with a P4 1.8 GHz processor, equipped with 512MB of RAM. A further algorithm has been used for benchmarking: AAM Search. This is a simple algorithm in which the AAM Search is applied frame-by-frame: it cannot be used for tracking in practical situations, since it is neither accurate nor robust, however it gives a kind of reference for time performance. From Table 1, it is clear that, despite the increased complexity of the tracker, the

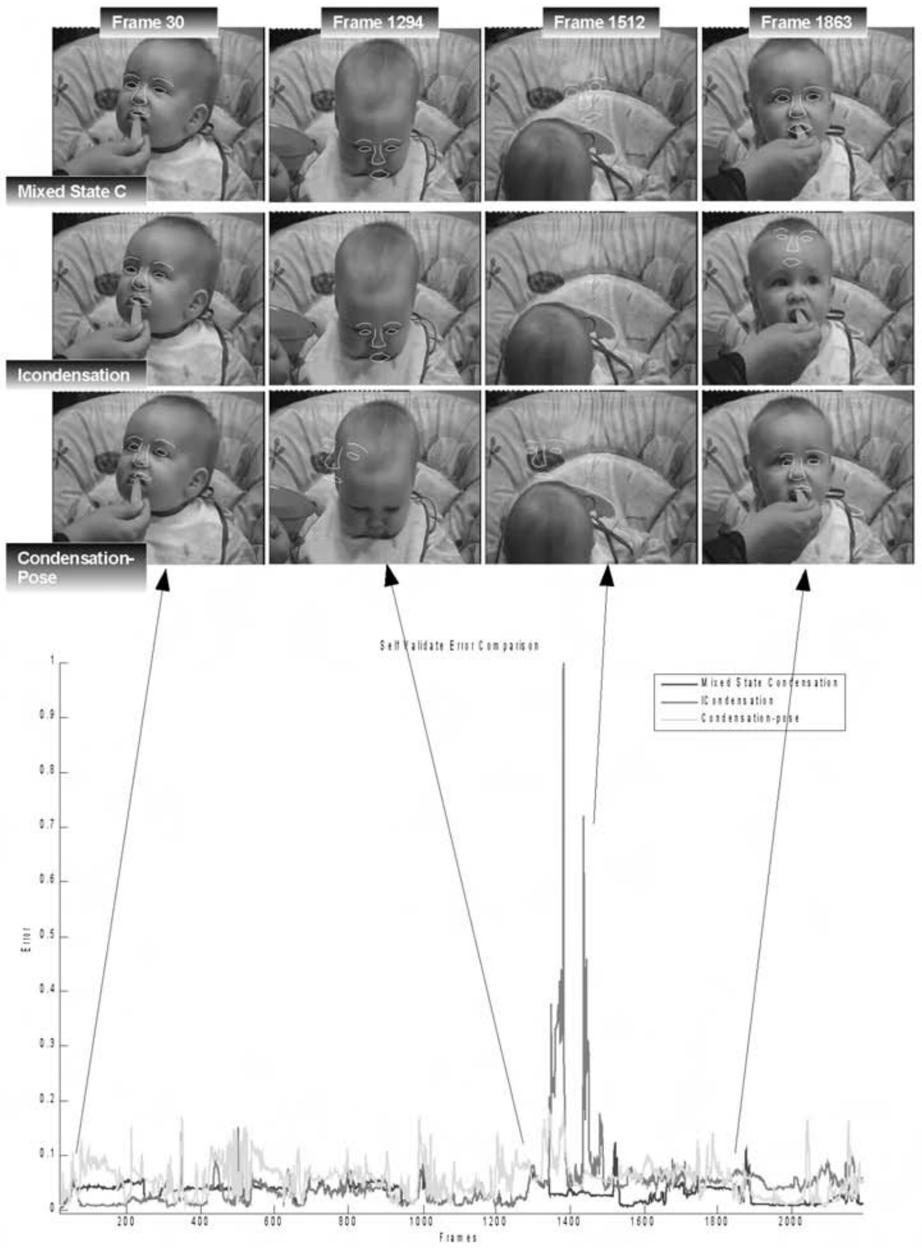


Fig. 3. Tracking results and tracking errors (Lorentzian norm) for Sequence 1

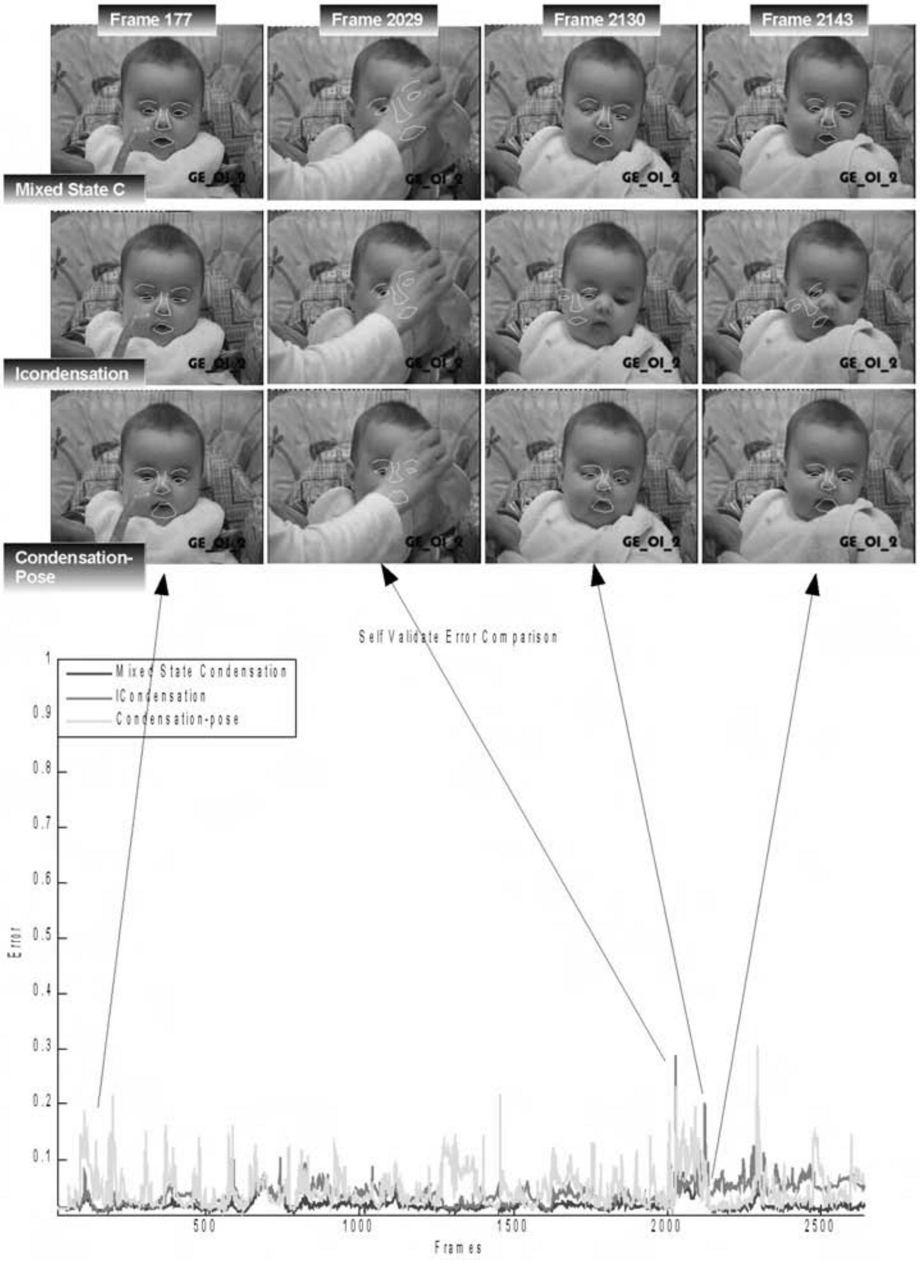


Fig. 4. Tracking results and tracking errors (Lorentzian norm) for Sequence 2

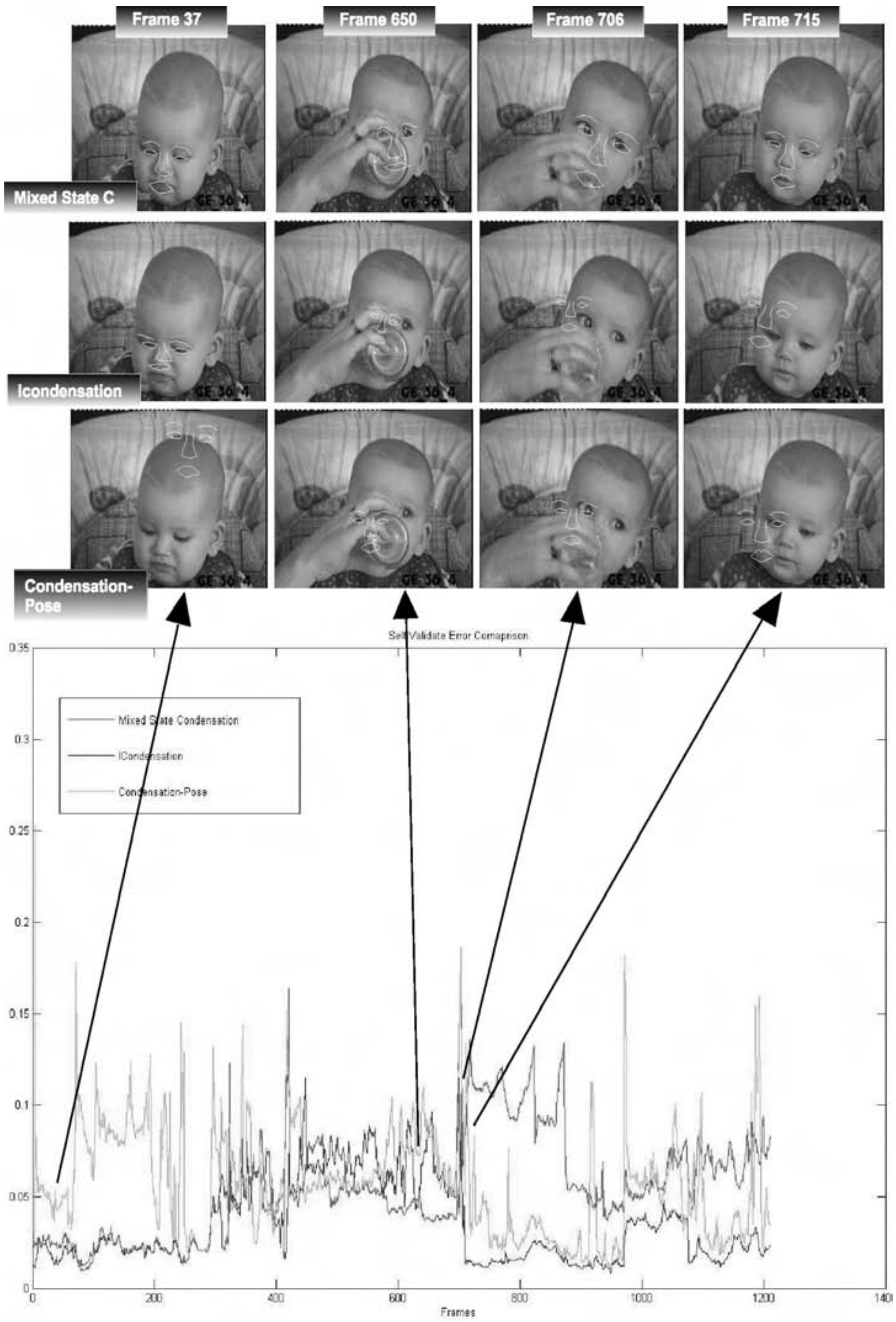


Fig. 5. Tracking results and tracking errors (Lorentzian norm) for Sequence 3

Table 1. Time performance comparison (in frames per second)

	IC	**C-pose**	**Mixed State C.**	**AAM Search**
Sequence 1	2.62	2.58	2.53	3.18
Sequence 2	3.13	2.57	2.59	2.5
Sequence 3	3.19	2.48	2.5	2.63

mixed-state CONDENSATION has performance comparable with those of the other algorithms.

5 Conclusion

In this work we presented a stochastic framework for robust face tracking using complex models of face appearance. When compared to other approaches found in literature [6], the presented tracker not only succeeds in crucial cases of occlusion, but experiments show that it outperforms in accuracy the compared methods and finds an equilibrate trade-off between robustness and use of time resources. The resulting algorithm is an adapted mixed state CONDENSATION combined with AAM. The stochastic CONDENSATION search compensates for AAM limits in handling occlusions, and due to an appropriate choice of motion models, allows an efficient reinitialization of tracking, when an exhaustive search in the image is impractical. Furthermore, the approach is general enough to be applied to other face tracking problems: an advantage of the probabilistic approach is that it is modular, in the sense that application-specific dynamic models or observation models can be seamlessly included.

References

1. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. Pattern Analysis and Machine Intelligence, IEEE Transactions on 23, 681–685 (2001)
2. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: CVPR, vol. 2, pp. 142–149 (2000)
3. Anderson, B.D.O., Moore, J.B.: Optimal Filtering. Dover Publications (2005)
4. Doucet, A., Freitas, N.D., Gordon, N.: Sequential Monte Carlo Methods in Practice (Statistics for Engineering and Information Science), 1st edn. Springer, Heidelberg (2005)
5. Blake, A., Isard, M.: The CONDENSATION algorithm - conditional density propagation and applications to visual tracking. In: Mozer, M., Jordan, M.I., Petsche, T. (eds.) NIPS, pp. 361–367. MIT Press, Cambridge (1996)
6. Hamlaoui, S., Davoine, F.: Facial action tracking using an aam-based condensation approach. In: IEEE ICASSP, Philadelphia, U.S.A (2005)
7. Davoine, F., Dornaika, F.: Head and facial animation tracking using appearanceadaptive models and particle filters. Springer, Heidelberg (2005)
8. Isard, M., Blake, A.: ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. In: Burkhardt, H.-J., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1406, pp. 893–908. Springer, Heidelberg (1998)
9. Isard, M., Blake, A.: A mixed-state CONDENSATION tracker with automatic model-switching. In: ICCV, pp. 107–112 (1998)

Gradient-Based Hand Tracking Using Silhouette Data

Paris Kaimakis and Joan Lasenby

Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK
{pk228,jl}@eng.cam.ac.uk

Abstract. Optical motion capture can be classified as an inference problem: given the data produced by a set of cameras, the aim is to extract the hidden state, which in this case encodes the posture of the subject's body. Problems with motion capture arise due to the multi-modal nature of the likelihood distribution, the extremely large dimensionality of its state-space, and the narrow region of support of local modes. There are also problems with the size of the data and the difficulty with which useful visual cues can be extracted from it, as well as how informative these cues might be. Several algorithms exist that use stochastic methods to extract the hidden state, but although highly parallelisable in theory, such methods produce a heavy computational overhead even with the power of today's computers. In this paper we assume a set of pre-calibrated cameras and only extract the subject's silhouette as a visual cue. In order to describe the 2D silhouette data we define a 2D model consisting of conic fields. The resulting likelihood distribution is differentiable w.r.t. the state, meaning that its global maximum can be located fast using gradient ascent search, given manual initialisation at the first frame. In this paper we explain the construction of the model for tracking a human hand; we describe the formulation of the derivatives needed, and present initial results on both real and simulated data.

1 Introduction

In the last few years there has been an ever growing demand for fast and reliable motion capture systems. There are countless applications, the most significant ones being surveillance for crime prevention and public safety, human-computer interaction (HCI) including interaction with game consoles, animation and many more. The only commercially available systems to date are marker-based e.g. [1] and [2], whose intrusive nature makes them inappropriate for surveillance or HCI applications.

One of the main problems related to motion capture is the multi-modality of the likelihood distribution. Several systems exist that use Monte Carlo Sampling (MCS) techniques to reliably handle several modes [3], [4], [5], [6], but although highly parallelisable in theory, the extremely large dimensionality of the likelihood and the narrow nature of its local modes imply that an enormous number of particles needs to be maintained throughout tracking. Despite their robustness and their parallelisation potential, such methods produce a heavy computational overhead even with the power of today's computers, and tracking may only be applied in post-process. One can deal with the 'curse of dimensionality' by considering prior information about the motion of the subject to be tracked [7], [8]. Although this results in a significant increase in speed of performance, such methods require lengthy training sessions and are limited in tracking a set of pre-specified movements. Finally there also are systems that rely on the use of higher-level features such as passive markers [1], active markers [2], voxel data [9], correlation data [10], depth maps [11] etc. to define a likelihood distribution with fewer local modes, but special equipment needs to be used and data acquisition becomes expensive and cumbersome.

A significant amount of literature already exists on model-based human hand tracking [5], [6], [8], [11], [12], [13]. Stenger et al. present impressive results using edge data in [12], but their tracked gestures involve movement of very few fingers and the state-space for the likelihood only consists of 7 dimensions. In [13] they introduce Hierarchical Filtering where a pre-computed hierarchical tree (whose nodes represent samples in the state-space) is compiled before tracking begins, for the purpose of saving on-line processing time. The precision of this system depends on the resolution of the samples stored on the leaf nodes of the tree, and the extracted gestures can be shaky. Bray et al. introduce the Smart Particle Filter [11], which uses a gradient-based algorithm on multiple samples to prevent convergence on a local maximum. Their tracker is facilitated by the use of 3D depth maps generated by a structured light sensor. Although tracking in 3D is now a lot easier and their results are impressive, assumption of the light sensor makes an HCI system based on their algorithm less appealing.

The demand for interactive systems for use at home suggests that HCI technology has to be fast and cheap. Hence, the visual features used for tracking need to be simple enough for the system to extract the data in real time. Furthermore, the tracking method needs to be a gradient-based one, this way ensuring that the subject's pose is estimated as fast as possible. A very limited number of systems follows these guidelines [14] but to date no such system is convincingly accurate and robust. In this paper we present a system to track a human hand in 26 degrees of freedom with no need for a training session, and making no pre-assumptions on the gesture to be tracked. We use only silhouette data, which is the simplest and easiest-to-extract low-level feature available. Silhouettes can be extracted in real-time by the cheapest camera equipment [15] and can be processed with minimal computational overhead. We define a differentiable likelihood distribution, meaning that tracking can take place much faster using gradient ascent, given adequate manual state initialisation at the first frame.

2 Formulation

The state vector $\mathbf{x}$ denotes any possible posture that the subject may attain at any given time t_k. Let $\mathcal{P}(\mathbf{x})$ be a description of the process that produces the observation $\mathbf{Z}_k$, under some arbitrary posture for the subject. Assuming that modelling of this process is realistic, we have

$$\mathbf{Z}_k = \mathcal{P}(\chi) + \mathbf{v} \tag{1}$$

where the observation $\mathbf{Z}_k$ is the silhouette data extracted from the k^{th} frameset captured by the cameras, $\mathbf{v}$ is a zero-mean random process which describes the discrepancy between the model and the observation, and χ is the true state of the subject that also brings the model closest to the observation. $\mathcal{P}(\mathbf{x})$, $\mathbf{Z}_k$ and $\mathbf{v}$ are all vectors of size P, where P is the total number of pixels in our set of cameras. Our task is to find the best estimate for χ; we choose the maximum likelihood (ML) estimate, which also *minimises* the negative log-likelihood, i.e.,

$$\hat{\mathbf{x}}^{ML} = \arg\min_{\mathbf{x}} \left(\mathbf{Z}_k - \mathcal{P}(\mathbf{x})\right)^T \left(\mathbf{Z}_k - \mathcal{P}(\mathbf{x})\right) \ . \tag{2}$$

To converge to $\hat{\mathbf{x}}^{ML}$ we use the Gauss-Newton method for Non-Linear Least Squares. For each iteration i a new approximation for $\hat{\mathbf{x}}^{ML}$ is given by,

$$\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i + \left[\mathbf{J}^T_{\mathcal{P}}(\hat{\mathbf{x}}_i)\mathbf{J}_{\mathcal{P}}(\hat{\mathbf{x}}_i)\right]^{-1} \mathbf{J}^T_{\mathcal{P}}(\hat{\mathbf{x}}_i)\left(\mathbf{Z}_k - \mathcal{P}(\hat{\mathbf{x}}_i)\right) \tag{3}$$

where $\mathbf{J}_{\mathcal{P}}(\mathbf{x}) \equiv \left[\nabla\mathcal{P}(\mathbf{x})\right]^T$ is the Jacobian of the model's silhouette.

3 The Model

3.1 State

The full state vector $\mathbf{x}$ holds position and orientation information for every bone in the subject's skeleton. The first three coefficients of $\mathbf{x}$ constitute $\mathbf{x}_{\text{sp}}$, the spatial position of the skeleton's root (which for our case represents the wrist). The rest of the coefficients in $\mathbf{x}$ combine in groups of three, and each such group forms $\mathbf{x}_b$, a vector which contains orientation information for bone b w.r.t. its 'parent' bone, i.e. the bone to which b is attached to. In other words,

$$\mathbf{x} = \left[\mathbf{x}^T_{\text{sp}} \quad \mathbf{x}^T_1 \quad \mathbf{x}^T_2 \quad \ldots \quad \mathbf{x}^T_b \quad \ldots \quad \mathbf{x}^T_B\right]^T \tag{4}$$

where B is the total number of bones, $\mathbf{x}_{\text{sp}} = [x \ \ y \ \ z]^T$, and $\mathbf{x}_b \in \mathbb{R}^3$. Using $\mathbf{x}_b$ we can define

$$\mathbf{q}_b = [q_0 \ \ q_x \ \ q_y \ \ q_z]^T_b = \left[\cos|\mathbf{x}_b| \quad \frac{\sin|\mathbf{x}_b|}{|\mathbf{x}_b|}\mathbf{x}^T_b\right]^T \ . \tag{5}$$

The parameters stored in $\mathbf{q}_b$ are the coefficients of a unit quaternion. A rotation matrix giving us the rotation of bone b w.r.t. its parent b_{p} can be formed from the quaternion coefficients as follows:

$$\mathbf{R}^{\text{B}}_b = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 2(q_xq_y + q_0q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_yq_z - q_0q_x) \\ 2(q_xq_z - q_0q_y) & 2(q_yq_z + q_0q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{6}$$

where the superscript B (note different from B) emphasises that this rotation relates to a bone. The rotation of the subject's root bone $\mathbf{R}^{\text{B}}_1(\mathbf{x})$ is defined w.r.t. the 3D origin.

3.2 Skeleton Model

The 3D structure and functionality of the subject are modelled by a knowledge base of the subject's underlying skeleton $\mathcal{S}$, which is defined as a tree structure and can concisely be represented by $\mathbf{c}_b$, the chain of bones from the root bone up to bone b. For each bone we also define a binary vector $\mathbf{f}_b \in \mathbb{R}^3$, whose i^{th} entry if non-zero, indicates freedom of bone b to rotate about its i^{th} principal axis of rotation. Finally we define a length s_b for each bone. Put in a more formal notation, the definition of the underlying Skeleton model parameters includes definitions for $\{\mathbf{c}_b,\ \mathbf{f}_b,\ s_b\}_{b=1}^{B}$. There are a total of $B = 25$ bones in our current implementation of the Skeleton model, including carpal bones, metacarpals and phalanges. The mobility of the wrist joint is represented by $\mathbf{f}_1 = [1\ \ 1\ \ 1]^T$, while all carpals and metacarpals have $\mathbf{f}_b = [0\ \ 0\ \ 0]^T$, proximal phalanges have $\mathbf{f}_b = [1\ \ 0\ \ 1]^T$, and intermediate/distal phalanges have $\mathbf{f}_b = [1\ \ 0\ \ 0]^T$.

Clearly, the underlying skeleton's configuration in 3D space is always dictated by the state, and it is this configuration that we are interested in extracting. Any topology that the skeleton may attain at any time can be expressed as $\mathcal{S}(\mathbf{x})$. This can be described as the 3D space position $\mathbf{t}_{0,b}^{\mathrm{B}}$ and orientation $\mathbf{R}_{0,b}^{\mathrm{B}}$ of every bone b in the skeleton w.r.t. the 3D origin:

$$\mathcal{S}(\mathbf{x}) = \left\{\mathbf{t}_{0,b}^{\mathrm{B}}(\mathbf{x}),\ \mathbf{R}_{0,b}^{\mathrm{B}}(\mathbf{x})\right\}_{b=1}^{B} \tag{7}$$

where the superscripts specify that these translations and rotations refer to the Skeleton model's bones. Each bone's rotation w.r.t. the 3D origin is given by

$$\mathbf{R}_{0,b}^{\mathrm{B}}(\mathbf{x}) = \prod_{m\in\mathbf{c}_b} \mathbf{R}_{m}^{\mathrm{B}}(\mathbf{x}) \tag{8}$$

where $\mathbf{R}_{m}^{\mathrm{B}}(\mathbf{x})$ is the state-defined rotation matrix for bone m w.r.t. its parent bone, as defined in (6). The ordering of rotations $\mathbf{R}_{m}^{\mathrm{B}}(\mathbf{x})$ is also important and the Π-notation used above preserves their ascending order w.r.t. their bone index m. The bone's centre is given by

$$\mathbf{t}_{0,b}^{\mathrm{B}}(\mathbf{x}) = \mathbf{x}_{\mathrm{sp}}(\mathbf{x}) + \sum_{m\in\mathbf{c}_{b_{\mathrm{p}}}} s_m \mathbf{R}_{0,m}^{\mathrm{B}}(\mathbf{x})\mathbf{e}_2 + \frac{1}{2}s_b\mathbf{R}_{0,b}^{\mathrm{B}}(\mathbf{x})\mathbf{e}_2 \tag{9}$$

where $\mathbf{e}_2 = [0\ \ 1\ \ 0]^T$ is the default ($\mathbf{x} = \mathbf{0}$) orientation of the bones, and b_{p} is the parent-bone of b, i.e. the penultimate entry in $\mathbf{c}_b$. A visualisation of the Skeleton model is shown in Fig. 1(a).

3.3 Corpulence Model

A visual description of the 3D nature of the subject can only be completed when its soft tissue and skin, here collectively referred to as *corpulence* $\mathcal{C}$, is taken into account. This involves modelling the visible soft tissue that surrounds the underlying skeleton. To model the corpulence, each bone b is assigned a total of E_b ellipsoids. Each ellipsoid ϵ_b is associated with 3 radii along its principal axes, a translation $\mathbf{t}_{\epsilon_b}^{\mathrm{E}}$ and a rotation $\mathbf{R}_{\epsilon_b}^{\mathrm{E}}$ w.r.t. the centre of bone b. Note that the superscript E indicates that these translations and rotations refer to the ellipsoids of the Corpulence model, not the bones of the Skeleton model. Formally, the Corpulence model parameters are described by $\left\{r_{i,\epsilon_b},\ \mathbf{t}_{\epsilon_b}^{\mathrm{E}},\ \mathbf{R}_{\epsilon_b}^{\mathrm{E}}\right\}_{\epsilon_b=1}^{E_b}$ for each individual bone b. Most of the bones in our human hand model are dressed with $E_b = 2$ ellipsoids, of which one usually represents a joint between bone b and its parent b_{p}. For those ellipsoids not representing a joint, the major axis radius is half of the bone's length, i.e. $r_{y,\epsilon_b} = \frac{1}{2}s_b$. This arrangement gave a realistic representation of the 3D nature of a real human hand.

Now, apart from being a 3D entity, $\mathcal{C}$ also has to be differentiable. For this reason we use quadric fields [16], represented by symmetric 4×4 matrices $\mathbf{Q}$. Quadrics have been used extensively in the literature, mainly because their contours produce primitives visually suitable for modelling limbs [5], [6], [7], [10], [12], [13]. When $\mathbf{Q}$ is of full rank, the quadric field's zero-contour coincides with the ellipsoids defined above. For the case of each ellipsoid ϵ_b associated to bone b, we can therefore define a quadric centred at the 3D origin:

$$\mathbf{Q}_{\epsilon_b} = \begin{pmatrix} \mathbf{D}_{\epsilon_b} & \mathbf{0} \\ \mathbf{0}^T & -1 \end{pmatrix} \tag{10}$$

where $\mathbf{D}_{\epsilon_b}$ is a 3×3 diagonal matrix whose i^{th} entry along the diagonal is $1/r^2_{i,\epsilon_b}$. The Euclidean transformation to move it from the origin to its state-dictated position is given in homogeneous coordinates by

$$\mathbf{T}_{\epsilon_b}(\mathbf{x}) = \begin{pmatrix} \mathbf{R}^{\mathrm{B}}_{0,b}(\mathbf{x})\mathbf{R}^{\mathrm{E}}_{\epsilon_b} & \mathbf{t}^{\mathrm{B}}_{0,b}(\mathbf{x}) + \mathbf{R}^{\mathrm{B}}_{0,b}(\mathbf{x})\mathbf{t}^{\mathrm{E}}_{\epsilon_b} \\ \mathbf{0}^T & 1 \end{pmatrix} . \tag{11}$$

The transformed quadric is

$$\mathbf{Q}'_{\epsilon_b}(\mathbf{x}) = \mathbf{T}^{-T}_{\epsilon_b}(\mathbf{x})\mathbf{Q}_{\epsilon_b}\mathbf{T}^{-1}_{\epsilon_b}(\mathbf{x}) \ , \tag{12}$$

and every such quadric will produce a smooth 3D field, defined for every point $\mathbf{M}$ in space as

$$\mathcal{C}_{\epsilon_b}(\mathbf{x}, \mathbf{M}) = \begin{cases} \mathbf{M}^T\mathbf{Q}'_{\epsilon_b}(\mathbf{x})\mathbf{M} & \text{if} \quad \mathbf{M}^T\mathbf{Q}'_{\epsilon_b}(\mathbf{x})\mathbf{M} \le 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $\mathbf{M} = [X \quad Y \quad Z \quad 1]^T$. This will create a field spatially confined within the limits imposed by the surface of ellipsoid ϵ_b. The full Corpulence model consists of one such field due to every ellipsoid ϵ_b present in every bone b of the Skeleton model:

$$\mathcal{C}(\mathbf{x}, \mathbf{M}) = \min_{\epsilon_b} \left(\left\{ \{\mathcal{C}_{\epsilon_b}(\mathbf{x}, \mathbf{M})\}_{\epsilon_b=1}^{E_b} \right\}_{b=1}^{B} \right) \tag{14}$$

where the $\min$ operator chooses between several quadric fields ϵ_b, if more than one are non-zero at any 3D position $\mathbf{M}$. A visualisation of the Corpulence model is shown in Fig. 1(b).

3.4 Projection Model

Under the assumption of pin-hole cameras we use perspective projection with our 3D Corpulence model $\mathcal{C}(\mathbf{x})$ to define corresponding conic fields on the image plane onto which each of its quadric fields will project. These conic fields are defined through their conic envelopes $\mathbf{C}'^{*}_{\epsilon_b}$:

$$\mathbf{C}'^{*}_{\epsilon_b}(\mathbf{x}) = \mathbf{P}\mathbf{Q}'^{*}_{\epsilon_b}(\mathbf{x})\mathbf{P}^T \tag{15}$$

where $\mathbf{P}$ is the 3×4 camera projection matrix that holds both intrinsic and extrinsic calibration parameters for each of the cameras, and $\mathbf{Q}'^{*}_{\epsilon_b}(\mathbf{x})$ is the envelope of the state-transformed quadric field associated with ellipsoid ϵ_b. For our model, all the quadrics are of full rank, and the quadric envelope is given by

$$\mathbf{Q}'^{*}_{\epsilon_b} = \mathbf{Q}'^{-1}_{\epsilon_b} \ . \tag{16}$$

The resulting conic envelope $\mathbf{C}'^{*}_{\epsilon_b}$ is a 3×3 symmetric matrix of full rank, and the projected conic is therefore given by

$$\mathbf{C}'_{\epsilon_b} = (\mathbf{C}'^{*}_{\epsilon_b})^{-1} \ . \tag{17}$$

It should be noted here that all conics are 3×3 symmetric matrices, and that full-ranked quadrics project to full-ranked conics, i.e. that ellipsoids always project to ellipses. The 2D conic field that forms the projection of each 3D quadric field onto the image plane becomes

$$\mathcal{P}_{\epsilon_b}(\mathbf{x}, \mathbf{m}) = \begin{cases} \mathbf{m}^T \mathbf{C}'_{\epsilon_b}(\mathbf{x}) \mathbf{m} & \text{if} \quad \mathbf{m}^T \mathbf{C}'_{\epsilon_b}(\mathbf{x}) \mathbf{m} \leq 0 \\ \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

for every pixel $\mathbf{m} = [u \quad v \quad 1]^T$. In general there will be one such conic field due to every ellipsoid ϵ_b defined by the Corpulence model. The full Projection model, which forms the basis of our comparison with the visual cue extracted from the data, is therefore given by

$$\mathcal{P}(\mathbf{x}, \mathbf{m}) = \min_{\epsilon_b} \left(\left\{ \{\mathcal{P}_{\epsilon_b}(\mathbf{x}, \mathbf{m})\}_{\epsilon_b=1}^{E_b} \right\}_{b=1}^{B} \right) \tag{19}$$

where the $\min$ operator chooses between several conic fields ϵ_b, of different bones b, if more than one are non-zero at any pixel $\mathbf{m}$. Finally, $\mathcal{P}(\mathbf{x})$ is a consideration of all pixels $\mathbf{m}$ present in the cameras, making it a P-sized vector:

$$\mathcal{P}(\mathbf{x}) = \begin{bmatrix} \mathcal{P}(\mathbf{x}, \mathbf{m}_1) \ \ldots \ \mathcal{P}(\mathbf{x}, \mathbf{m}_p) \ \ldots \ \mathcal{P}(\mathbf{x}, \mathbf{m}_P) \end{bmatrix}^T \tag{20}$$

where P is the total number of pixels present in the cameras. The projection of our human hand model, viewed as an image, can be seen in Fig. 1(c). The relationship between the Skeleton, Corpulence and Projection models is shown in Fig. 1(d).

4 Derivatives

Differentiating (20) w.r.t. $\mathbf{x}$ gives $\mathbf{J}_{\mathcal{P}}(\mathbf{x})$, a $P \times S$ matrix, where S is the number of state parameters allowed to change through tracking. The (p, s) component of $\mathbf{J}_{\mathcal{P}}$ is

$$[\mathbf{J}_{\mathcal{P}}(\mathbf{x})]_{p,s} = \frac{\partial \mathcal{P}(\mathbf{x}, \mathbf{m}_p)}{\partial x_s} \ . \tag{21}$$

Using (19),

$$\frac{\partial \mathcal{P}(\mathbf{x}, \mathbf{m}_p)}{\partial x_s} = \frac{\partial \mathcal{P}_{\varepsilon_b}(\mathbf{x}, \mathbf{m}_p)}{\partial x_s} \tag{22}$$

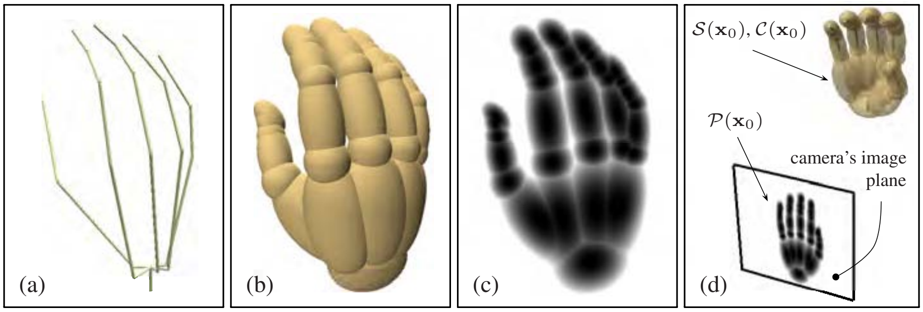

(a)
(b)
(c)
(d)
$\mathcal{S}(\mathbf{x}_0), \mathcal{C}(\mathbf{x}_0)$
$\mathcal{P}(\mathbf{x}_0)$
camera's image plane

Fig. 1. Skeleton, Corpulence and Projection models for the human hand. For all these models the state for the hand's rest-position $\mathbf{x}_0$ was used. **(a)** Skeleton model $\mathcal{S}(\mathbf{x}_0)$. Each bone b is drawn as a cylinder with position $\mathbf{t}^{\mathrm{B}}_{0,b}$ and orientation $\mathbf{R}^{\mathrm{B}}_{0,b}$ w.r.t. the 3D origin. Lighting conditions have been used for illustration of the three-dimensionality of $\mathcal{S}(\mathbf{x}_0)$. **(b)** Corpulence model $\mathcal{C}(\mathbf{x}_0)$. Each bone b of the Skeleton model is now dressed with E_b quadric fields. Note that for visualisation purposes only the contour $\mathcal{C}(\mathbf{x}_0) = 0$ is displayed. Lighting conditions have also been used here. **(c)** Projection model $\mathcal{P}(\mathbf{x}_0)$. The corpulence is now projected to a 2D image, as seen from the viewpoint of the reader. Each quadric field $\mathbf{Q}'_{\epsilon_b}$ from the Skeleton model projects down to a conic field $\mathbf{C}'_{\epsilon_b}$. **(d)** Illustration of Skeleton, Corpulence (drawn semi-transparent here) and Projection models as viewed from an arbitrary viewpoint.

where ε_b refers to the conic that minimised (19), i.e.

$$\varepsilon_b = \arg\min_{\epsilon_b} \left(\left\{ \{\mathcal{P}_{\epsilon_b}(\mathbf{x}, \mathbf{m})\}_{\epsilon_b=1}^{E_b} \right\}_{b=1}^{B} \right) \quad . \tag{23}$$

From (18),

$$\frac{\partial \mathcal{P}_{\varepsilon_b}(\mathbf{x}, \mathbf{m}_p)}{\partial x_s} = \begin{cases} \mathbf{m}_p^T \frac{\partial \mathbf{C}'_{\varepsilon_b}}{\partial x_s} \mathbf{m}_p & \text{if} \quad \mathbf{m}_p^T \mathbf{C}'_{\varepsilon_b}(\mathbf{x}) \mathbf{m}_p \leq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The gradient of the transformed conic can be calculated by differentiating (17):

$$\frac{\partial \mathbf{C}'_{\varepsilon_b}}{\partial x_s} = -\mathbf{C}'_{\varepsilon_b} \frac{\partial \left(\mathbf{C}'^{*}_{\varepsilon_b}\right)}{\partial x_s} \mathbf{C}'_{\varepsilon_b} \quad , \tag{25}$$

and from (15) we have

$$\frac{\partial(\mathbf{C}'^{*}_{\varepsilon_b})}{\partial x_s} = \mathbf{P} \frac{\partial(\mathbf{Q}'^{*}_{\varepsilon_b})}{\partial x_s} \mathbf{P}^T \quad . \tag{26}$$

Equation (16) gives

$$\frac{\partial(\mathbf{Q}'^{*}_{\varepsilon_b})}{\partial x_s} = -\mathbf{Q}'^{*}_{\varepsilon_b} \frac{\partial \mathbf{Q}'_{\varepsilon_b}}{\partial x_s} \mathbf{Q}'^{*}_{\varepsilon_b} \quad , \tag{27}$$

and using (12),

$$\frac{\partial \mathbf{Q}'_{\varepsilon_b}}{\partial x_s} = \frac{\partial(\mathbf{T}^{-T}_{\varepsilon_b})}{\partial x_s} \mathbf{Q}_q \mathbf{T}^{-1}_q + \mathbf{T}^{-T}_q \mathbf{Q}_q \frac{\partial(\mathbf{T}^{-1}_q)}{\partial x_s} \tag{28}$$

where

$$\frac{\partial(\mathbf{T}'^{-1}_{\varepsilon_b})}{\partial x_s} = -\mathbf{T}^{-1}_{\varepsilon_b}\frac{\partial \mathbf{T}_{\varepsilon_b}}{\partial x_s}\mathbf{T}^{-1}_{\varepsilon_b} \ . \tag{29}$$

From (11) we have

$$\frac{\partial \mathbf{T}_{\varepsilon_b}}{\partial x_s} = \begin{pmatrix} \frac{\partial \mathbf{R}^{\mathrm{B}}_{0,b}}{\partial x_s}\mathbf{R}^{\mathrm{E}}_{\varepsilon_b} & \frac{\partial \mathbf{t}^{\mathrm{B}}_{0,b}}{\partial x_s} + \frac{\partial \mathbf{R}^{\mathrm{B}}_{0,b}}{\partial x_s}\mathbf{t}^{\mathrm{E}}_{\varepsilon_b} \\ \mathbf{0}^T & 0 \end{pmatrix} \tag{30}$$

where the gradient for the position of bone b is found by differentiating (9):

$$\frac{\partial \mathbf{t}^{\mathrm{B}}_{0,b}}{\partial x_s} = \frac{\partial \mathbf{x}_{\mathrm{sp}}}{\partial x_s} + \sum_{m \in \mathbf{c}_{b_{\mathrm{p}}}} s_m \frac{\partial \mathbf{R}^{\mathrm{B}}_{0,m}}{\partial x_s}\mathbf{e}_2 + \frac{1}{2}s_b\frac{\partial \mathbf{R}^{\mathrm{B}}_{0,b}}{\partial x_s}\mathbf{e}_2 \ . \tag{31}$$

For the cases when $1 \leq s \leq 3$, the reduced-state parameter x_s corresponds to the spatial position of the root bone, i.e. $x_s \in \mathbf{x}_{\mathrm{sp}}$, and $\frac{\partial \mathbf{R}^{\mathrm{B}}_{0,m}}{\partial x_s} = \mathbf{0}$. This is because the state-dependent rotation matrices $\mathbf{R}_{0,m}(\mathbf{x})$ only depend on the orientation of the bones inside $\mathbf{c}_b$, as indicated by (8) and are therefore independent w.r.t. the parameters stored in $\mathbf{x}_{\mathrm{sp}}$. The gradient for the position of bone b in this case is therefore

$$\frac{\partial \mathbf{t}^{\mathrm{B}}_{0,b}}{\partial x_s} = \frac{\partial \mathbf{x}_{\mathrm{sp}}}{\partial x_s} = \mathbf{e}_s \tag{32}$$

where $\mathbf{e}_s \in \mathbb{R}^3$ is the unit vector along the s-axis, and calculation of $[\mathbf{J}_{\mathcal{P}}(\mathbf{x})]_{p,s}$ stops here. For the rest of the state parameters $4 \leq s \leq S$, the reduced-state parameter corresponds to a rotation for a specific bone β, i.e. $x_s \in \mathbf{x}_\beta$, and we have

$$\frac{\partial \mathbf{t}^{\mathrm{B}}_{0,b}}{\partial x_s} = \sum_{m \in \mathbf{c}_{\beta,b_{\mathrm{p}}}} s_m \frac{\partial \mathbf{R}^{\mathrm{B}}_{0,m}}{\partial x_s}\mathbf{e}_2 + \frac{1}{2}s_b\frac{\partial \mathbf{R}^{\mathrm{B}}_{0,b}}{\partial x_s}\mathbf{e}_2 \tag{33}$$

where $\mathbf{c}_{i,j}$ is a subset of $\mathbf{c}_j$ and contains all the bones between bones i and j inclusively (note that this also means that bone i is closer to the root than j). Now, $\frac{\partial \mathbf{R}^{\mathrm{B}}_{0,b}}{\partial x_s}$ is non-zero only when bone $\beta \in \mathbf{c}_b$, in which case

$$\frac{\partial \mathbf{R}^{\mathrm{B}}_{0,b}}{\partial x_s} = \mathbf{R}^{\mathrm{B}}_{0,\beta_{\mathrm{p}}}\frac{\partial \mathbf{R}^{\mathrm{B}}_{\beta}}{\partial x_s}\mathbf{R}^{\mathrm{B}}_{\beta,b} \tag{34}$$

where $\mathbf{R}^{\mathrm{B}}_{i,j}$ represents the rotation of bone j w.r.t. the frame of reference of bone i and is given by

$$\mathbf{R}^{\mathrm{B}}_{i,j} = \prod_{m \in \mathbf{c}_{i_{\mathrm{o}},j}} \mathbf{R}^{\mathrm{B}}_m \ , \tag{35}$$

with i_{o} being the offspring of i inside $\mathbf{c}_j$, i.e. the entry in $\mathbf{c}_j$ following the entry for i. It should be noted here that with the above definition, $\mathbf{R}^{\mathrm{B}}_{i_{\mathrm{p}},i} = \mathbf{R}^{\mathrm{B}}_i$, and that $\mathbf{R}^{\mathrm{B}}_{i,i} = \mathbf{I}_3$.

The derivative of the state rotation $\mathbf{R}_\beta^\mathrm{B}$ w.r.t. the state is found by differentiating (6), and e.g. the (1,1) component of the resulting 3×3 matrix is given by

$$\left(\frac{\partial \mathbf{R}_\beta^\mathrm{B}}{\partial x_s}\right)_{(1,1)} = 2\left(q_0\frac{\partial q_0}{\partial x_s} + q_x\frac{\partial q_x}{\partial x_s} - q_y\frac{\partial q_y}{\partial x_s} - q_z\frac{\partial q_z}{\partial x_s}\right) \tag{36}$$

where $q_0\ ,\ \ldots\ ,\ q_z$ are the quaternion coefficients associated with bone β, as defined in (5). The rest of the components in $\frac{\partial \mathbf{R}_\beta^\mathrm{B}}{\partial x_s}$ are found similarly. Finally, the derivatives of the quaternion coefficients w.r.t. the state parameters are found by differentiating (5):

$$\frac{\partial q_0}{\partial x_s} = -\sin|\mathbf{x}_\beta|\frac{\partial|\mathbf{x}_\beta|}{\partial x_s} \tag{37}$$

$$\frac{\partial}{\partial x_s}\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \left(\frac{\cos|\mathbf{x}_\beta|}{|\mathbf{x}_\beta|}\frac{\partial|\mathbf{x}_\beta|}{\partial x_s} - \frac{\sin|\mathbf{x}_\beta|}{|\mathbf{x}_\beta|^2}\frac{\partial|\mathbf{x}_\beta|}{\partial x_s}\right)\mathbf{x}_\beta + \frac{\sin|\mathbf{x}_\beta|}{|\mathbf{x}_\beta|}\mathbf{e}_\sigma \tag{38}$$

where $1 \leq \sigma \leq 3$ is the index of $\mathbf{x}_\beta$ that corresponds to s, and

$$\frac{\partial|\mathbf{x}_\beta|}{\partial x_s} = \frac{x_s}{|\mathbf{x}_\beta|}\ . \tag{39}$$

5 Results and Discussion

In all our experiments we used a reduced state vector containing only those elements of the full state that are free to change. This was done with the help of the $\mathbf{f}_b$'s, and produced a dimension for the (reduced) state $S = 26$. In the remainder of this paper $\mathbf{x}$ will refer to the reduced state. For every frame in our experiments we used equation (3) to search the 26-dimensional state-space for locating $\hat{\mathbf{x}}^{ML}$. The initial state per frame was taken to be the ML estimate from the previous frame. Then, for each iteration i in a frame, we employed equations (8) to (20) to calculate $\mathcal{P}(\hat{\mathbf{x}}_i)$ and (21) to (39) to calculate $\mathbf{J}_\mathcal{P}(\hat{\mathbf{x}}_i)$. The initial state for the first frame in our experiments was chosen to be the hand's rest-position $\mathbf{x}_0$ (see Fig. 1), so we only had to manually select the 6 state parameters for the position and orientation for the root-bone.

We first investigate the performance of the system running on simulated data. For the production of the simulated data we used our model to get images similar to $\mathcal{P}(\mathbf{x})$ in Fig. 1(c). The state that produced the simulated data was discarded and the algorithm was employed to recover it. The top row in Fig. 2 shows the simulated data from a selection of frames as seen from one of the 2 cameras used for the experiment, and a visualisation of the Corpulence model at the converged $\hat{\mathbf{x}}^{ML}$, as seen from the same view. It should be noted that under this scenario we are tracking the (simulated) subject using a perfect model, since the model can describe the data with no error.

In reality the observations $\mathbf{Z}_k$ are binary silhouette images, implying that our model will *never* be able to fully describe the visual data with no error, thus making tracking more difficult. To produce such 'semi'-realistic data we thresholded every pixel in $\mathcal{P}(\mathbf{x})$ such that a binary image was produced from each viewpoint. The second row of Fig. 2

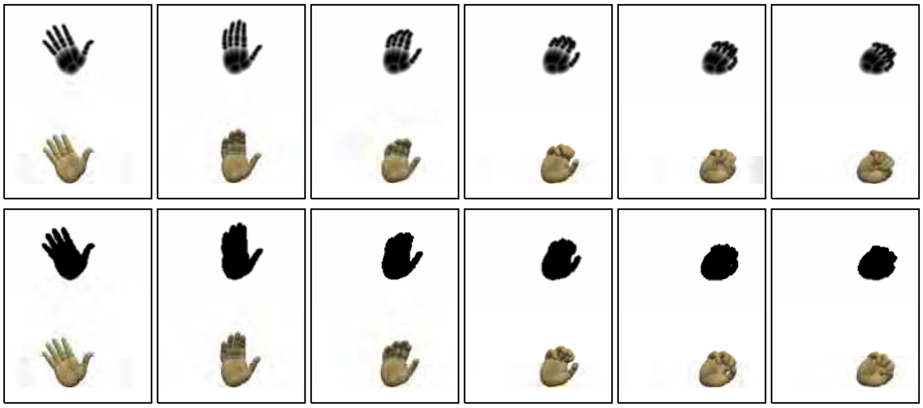


Fig. 2. Results from two experiments based on simulated data. For each experiment we show the silhouette data $\mathbf{Z}_k$ (black-and-white image) as recorded by one of the cameras, and the locus $\mathcal{C}(\hat{\mathbf{x}}^{ML}) = 0$ under lighting conditions, from the same viewpoint. **Top Row:** Experiment based on purely simulated data using 2 cameras. **Bottom Row:** Experiment based on 'semi'-realistic (binary) data from the same simulated gesture using 6 cameras; the degree of ambiguity is now greater, hence a greater number of cameras was needed.

shows the semi-real data for the same selection of frames, seen from the same viewpoint as for the previous experiment, and a visualisation of the Corpulence model at the converged $\hat{\mathbf{x}}^{ML}$, as seen from the same view. It has to be stressed that as the simulated hand closes into a fist, the semi-realistic data $\mathbf{Z}_k$ is now a lot more ambiguous (compare the silhouette data in the last snapshots of row 1 against those of row 2 in Fig. 2), meaning that several distinct values for $\mathbf{x}$ exist that produce a model observation $\mathcal{P}(\mathbf{x})$ close to the data. In such cases $\hat{\mathbf{x}}^{ML}$ is not unique, and χ lies on an S-dimensional ridge. Any algorithm that maximises the likelihood will be unable to choose between positions along this ridge, and the subject's true state χ is therefore *unobservable*. Due to this ambiguity the converged state $\hat{\mathbf{x}}^{ML}$ deviates significantly from χ and tracking with 2 cameras loses its realism at this stage. For this reason we used 6 cameras for this experiment; the fact that more cameras helped the tracker converge to a state close to χ demonstrates that using data from different viewpoints reshapes the likelihood distribution and parts of the ridge are now suppressed so as to create a better-defined maximum, thus improving the observability for χ.

Finally, we tested the system under real data captured by 3 calibrated cameras running on 25 frames per second. We recorded our data in a controlled environment to make segmentation easy; alternatively we could have used one of the existing real-time foreground segmentation algorithms, e.g. [15]. We allowed 10 iterations for our system to converge to $\hat{\mathbf{x}}^{ML}$ in each frame of the data movies and this took around 2.5s per 3-camera frameset using Matlab, meaning that future implementation of our system in C will have real-time capability. Fig. 3 shows the data and the tracking results for three different gestures as seen by one of the cameras. Again, our system's capability to track is reduced when the real data correspond to a hand closing to a fist. In such cases (e.g. bottom row, second snapshot of Fig. 3) the converged state $\hat{\mathbf{x}}^{ML}$ corresponding to

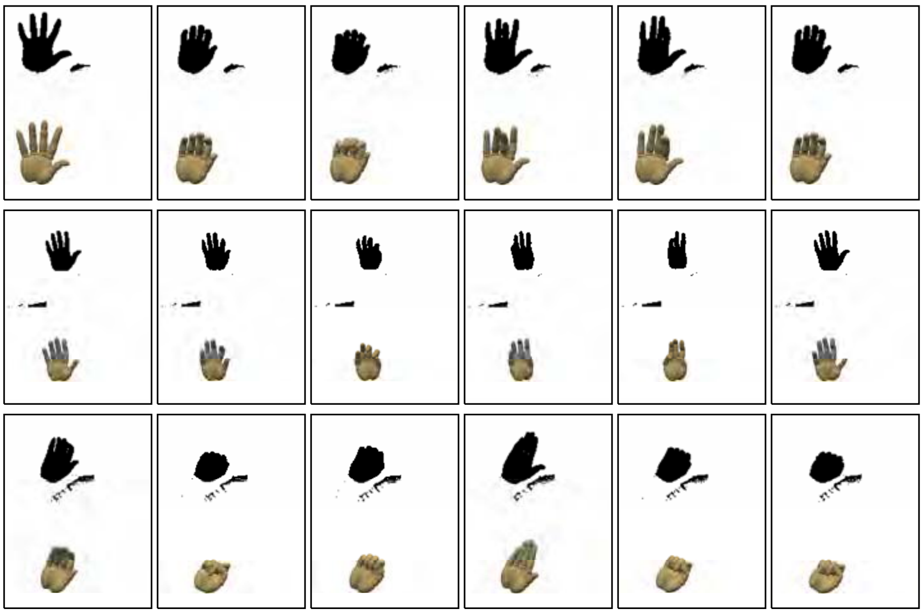


Fig. 3. Results from three experiments based on real data. **Top Row:** Minimal degree of self-occlusion is encountered. **Middle Row and Bottom Row:** Severe cases of self occlusion as the hand closes to a fist. Despite the lack of observability and our tracker converging to a $\hat{\mathbf{x}}^{ML} \neq \chi$ (e.g. bottom row, second snapshot from left), the deviation between $\hat{\mathbf{x}}^{ML}$ and χ is small enough for the tracked gesture to maintain its character. When the state becomes observable, the tracker is able to converge to the unique $\hat{\mathbf{x}}^{ML} = \chi$ once again.

these frames is close but not equal to χ, which is further down the same S-dimensional ridge. Unfortunately, addition of more cameras was not an option in our real-data experiment. Nevertheless, in subsequent frames, where there is less self-occlusion and the state is more observable given the data (meaning that $\hat{\mathbf{x}}^{ML}$ is unique once again), the tracker converges to the global maximum, and correct tracking resumes.

6 Conclusions

In this paper we have presented a system that can track a human hand of 26 degrees of freedom, relying exclusively on 2D silhouette data which are easily available with no need for special equipment or dedicated hardware. Our system is the first in the literature to utilise derivatives of conic fields in order to establish a silhouette-driven, gradient-based tracking scenario. The state-space of the resulting likelihood distribution is thus searched faster, and the tracker converges to the global maximum in a minimal number of iterations. Despite the limited number of cameras used in our real-data experiments at this stage, our results are very promising: gestures suffering from extensive self-occlusion are tracked successfully, whilst maintaining realism in the recovered motion. Contrary to several existing systems, our algorithm requires no training stage, which means that any previously unknown gesture can still be tracked.

Limitations to the tracking ability of our system arise due to the inherent ambiguity associated with silhouette data: in cases of severe self-occlusion the state becomes unobservable and the tracked gesture becomes less realistic. Although this limitation is not a problem in the majority of the gestures encountered in HCI applications, we aim to address this issue by considering constraints about the subject's posture without limiting the tracking ability of the system to pre-specified gestures. This may be done via a consideration of the functional limitations of the skeleton as in [6]. Finally, we will address the issue of state initialisation for the first frame by means of MCS methods.

References

1. Vicon Motion Systems and Peak Performance Inc.: Motion capture systems (Online Material), `http://www.vicon.com`
2. PhaseSpace Inc.: Optical motion capture systems (Online Material), `http://www.phasespace.com`
3. Isard, M., Blake, A.: CONDENSATION – conditional density propagation for visual tracking. Trans. International Journal of Computer Vision (IJCV) 29, 5–28 (1998)
4. Deutscher, J., Blake, A., Reid, I.: Articulated body motion capture by annealed particle filtering. In: CVPR 2000. Proc. IEEE International Conference on Computer Vision and Pattern Recognition, Hilton Head, SC, USA, pp. 2126–2133. IEEE Computer Society Press, Los Alamitos (2000)
5. Wu, Y., Lin, J., Huang, T.S.: Analyzing and capturing articulated hand motion in image sequences. Trans. IEEE Pattern Analalysis and Machine Intelligence (PAMI) 27 (2005)
6. Sudderth, E.B., Mandel, M.I., Freeman, W.T., Willsky, A.S.: Visual hand tracking using nonparametric belief propagation. In: Proc. CVPR Workshop, Washington, DC, USA (2004)
7. Urtasun, R., Fua, P.: 3D tracking for gait characterization and recognition. In: Proc. IEEE Automatic Face and Gesture Recognition, Seoul, Korea, pp. 17–22. IEEE Computer Society Press, Los Alamitos (2004)
8. Campos, T.E.D., Murray, D.W.: Regression-based hand pose estimation from multiple cameras. In: Proc. CVPR, New York, NY, USA, vol. 1, pp. 782–789 (2006)
9. Mikić, I., Trivedi, M.M., Hunter, E., Cosman, P.C.: Human body model acquisition and tracking using voxel data. Trans. IJCV 53, 199–223 (2003)
10. Plänkers, R., Fua, P.: Tracking and modeling people in video sequences. Trans. Computer Vision and Image Understanding (CVIU) 81, 285–302 (2001)
11. Bray, M., Koller-Meier, E., Gool, L.V.: Smart particle filtering for high-dimensional tracking. Trans. CVIU 106, 116–129 (2007)
12. Stenger, B., Mendonça, P.R.S., Cipolla, R.: Model-based 3D tracking of an articulated hand. In: Proc. CVPR, Kauai, HI, USA, vol. 2, pp. 310–315 (2001)
13. Stenger, B., Thayananthan, A., Torr, P.H.S., Cipolla, R.: Model-based hand tracking using a hierarchical bayesian filter. Trans. PAMI 28, 1372–1384 (2006)
14. Kaimakis, P., Lasenby, J.: Markerless motion capture with single and multiple cameras. In: Proc. IEEE International Conference on Image Processing, Singapore, pp. 2607–2610. IEEE Computer Society Press, Los Alamitos (2004)
15. Horprasert, T., Harwood, D., Davis, L.S.: A statistical approach for real-time robust background subtraction and shadow detection. In: Proc. IEEE International Conference on Computer Vision FRAME-RATE Workshop, Kerkyra, Greece, IEEE Computer Society Press, Los Alamitos (1999)
16. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2003)

Using Gaussian Processes for Human Tracking and Action Classification

Leonid Raskin, Ehud Rivlin, and Michael Rudzsky

Computer Science Department,
Technion—Israel Institute of Technology, Haifa, Israel, 32000
{raskinl,ehudr,rudzsky}@cs.technion.ac.il

Abstract. We present an approach for tracking human body parts and classification of human actions. We introduce Gaussian Processing Annealed Particle Filter Tracker (GPAPF), which is an extension of the annealed particle filter tracker and uses Gaussian Process Dynamical Model (GPDM) in order to reduce the dimensionality of the problem, increase the tracker's stability and learn the motion models. Motion of human body is described by concatenation of low dimensional manifolds which characterize different motion types. The trajectories in the latent space provide low dimensional representations of sequences of body poses performed during motion. Our approach uses these trajectories in order to classify human actions. The approach was checked on HumanEva data set as well as on our own one. The results and the comparison to other methods are presented.

Keywords: Articulated body tracking, Dimensionality reduction, Action classification.

1 Introduction

Human body pose estimation and tracking is a challenging task for several reasons. First, the large dimensionality of the human 3D model complicates the examination of the entire subject and makes it harder to detect each body part separately. Secondly, the significantly different appearance of different people that stems from various clothing styles and illumination variations, adds to the already great variety of images of different individuals. Finally, the most challenging difficulty that has to be solved in order to achieve satisfactory results of pose understanding is the ambiguity caused by body.

Despite the high dimensionality of the problem, many poses can be presented in a low dimensional space This space can be obtained using poses from different motion types [1,2]. This paper presents an approach to 3D people tracking and motion analysis. In this approach we apply a nonlinear dimensionality reduction using Gaussian Process Dynamical Model (GPDM) [3,4] and then we use annealed particle filter [5], which has been modified to operate in the reduced latent space. GPDM is better able to capture properties of high dimensional motion data than linear methods such as PCA. This method generates a mapping function from the low dimensional latent space to the full data space based on learning from previously observed poses of different motion types. For the tracking we separate model state into two independent parts: one contains information about 3D location and orientation of the body and the second one describes the pose. We learn latent space that describes poses only. The tracking algorithm consists of two stages. Firstly the particles are generated in the latent space and are transformed into the data space by using learned a priori mapping function. Secondly we add rotation and translation parameters to obtain valid poses. The likelihood function calculated in order to evaluate how well a pose matches the visual data. The resulting tracker estimates the locations in the latent space that represents poses with the highest likelihood. As the latent space is learned from sequences of poses from different motion types, each action is represented by a curve in the latent space. The classification of the motion action is based on the comparison of the sequences of latent coordinates that the tracker has produced, to the ones that represent poses sequences of different motion types. We use a modified Frèchet distance [6] in order to compare the pose sequences. This approach allows introducing different actions from the ones we have used for learning by exploiting the curve that represents it.

We show that our tracking algorithm provides good results even for the low frame rate. An additional advantage of our tracking algorithm is the capability to recover after temporal loss of the target. We also show that the task of action classification, when performed in the latent space, is robust.

2 Related Works

One of the common approaches for tracking is Particle Filtering methods. This method uses multiple predictions, obtained by drawing samples of pose and location prior and then propagating them using the dynamic model, which are refined by comparing them with the local image data, calculating the likelihood [7]. The prior is typically quite diffused (because motion can be fast) but the likelihood function may be very peaky, containing multiple local maxima which are hard to account for in detail [8]. Annealed particle filter [5] or local searches are the ways to attack this difficulty.

There exist several possible strategies for reducing the dimensionality of the configuration space. Firstly it is possible to restrict the range of movement of the subject [9]. Due to the restricting assumptions the resulting trackers are not capable of tracking general human poses. Another way to cope with high-dimensional data space is to learn low-dimensional latent variable models [10]. However, methods like Isomap [11] and locally linear embedding (LLE) [12] do not provide a mapping between the latent space and the data space, and, therefore Urtasun et. al [13] proposed to use a form of probabilistic dimensionality reduction by GPDM [3,4] to formulate the tracking as a nonlinear least-squares optimization problem.

During the last decade many different methods for behavior recognition and classification of human actions have been proposed. The popular methods are based on Hidden Markov Models (HMM), Finite State Automata (FSA), context-free grammar (SCFG) etc. Sato et al. [14] presented a method to use extraction of human trajectory patterns that identify the interactions. Park et al. [15] proposed a method using a nearest neighbor classifier for recognition of two-person interactions such as hand-shaking, pointing, and standing hand-in-hand. Hongeng et al. [16] proposed probabilistic finite state automata for recognition of a sequential occurrence of several scenarios. Park et al. [17] presented a recognition method that combines model-based tracking and deterministic finite state automata.

3 GPAPF Tracking

The annealed particle filter tracker [5] suffers from several drawbacks. First of all the number of the particles that need to be generated depends on the state space dimensionality and, practically, on the frame rate. An increase in the dimensionality of the state space causes an exponential increase in the number of particles that are needed to be generated in order to preserve the same density of particles. This makes this algorithm computationally ineffective for low frame rate videos (30 fps and lower). The other problem is that once a target is lost (i.e. the body pose was wrongly estimated, which can happen for the fast and not smooth movements) it becomes highly unlikely that the next pose will be estimated correctly in the following frames.

We propose using space reduction on the state of the particle filter. In order to reduce the dimension of the space we propose using Gaussian Process Dynamical Model (GPDM) [3,4]. We embedded several types of poses into a low dimensional space. The poses are taken from different sequences, such as walking, running, punching and kicking. We divide our state into two independent parts. The first part contains the global 3D body rotation and translation, which is independent of the actual pose. The second part contains only information regarding the pose (26 DoF). We use GPDM to reduce the dimensionality of the second part. This way we construct a latent space (Fig. 1). This space has a significantly lower dimensionality (for example 2 or 3 DoF). The latent space includes solely pose information and is therefore rotation and translation invariant. For the tracking task we use a modified annealed particle filter tracker [5]. We are using a 2-stage algorithm. The first stage is generation of new particles in the latent space, which is the main modification of the tracking algorithm. Then we apply the learned mapping function that transforms latent coordinates to the data space. As a result, after adding the translation and rotation information, we construct 31 dimensional vectors that describe a valid data state, which includes location and pose information, in the data space. In order to estimate how well the pose matches the images the likelihood function is calculated [18].

Suppose we have M annealing layers. The state is defined as a pair $\Gamma = \{\Lambda, \Omega\}$, where Λ is the location information and Ω is the pose information. We also define ω as a latent coordinates corresponding to the data vector Ω: $\Omega = \wp(\omega)$, where $\wp$ is the mapping function learned by the GPDM. $\Lambda_{n,m}$, $\Omega_{n,m}$ and $\omega_{n,m}$ are the location, pose vector and corresponding latent coordinates on

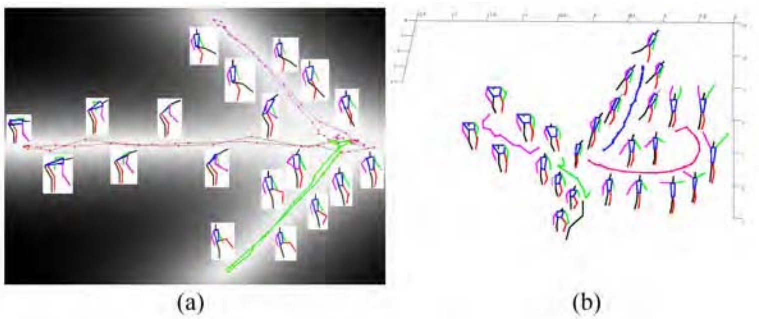


Fig. 1. The latent space that is learned from different motion types. (a) 2D latent space from 3 different motions: lifting an object (red), kicking with the left (green) and the right (magenta) legs. (b) 3D latent space from 3 different motions: hand waving (red), lifting an object (magenta), kicking (blue), sitting down (black), and punching (green).

the frame n and annealing layer m. For each $1 \leq m \leq M-1$ $\Lambda_{n,m}$ and $\omega_{n,m}$ are generated by adding multi-dimensional Gaussian random variable to $\Lambda_{n,m+1}$ and $\omega_{n,m+1}$ respectively. Then $\Omega_{n,m}$ is calculated using $\omega_{n,m}$. Full body state $\Gamma_{n,m} = \{\Lambda_{n,m}, \Omega_{n,m}\}$ is projected to the cameras and the likelihood $\pi_{n,m}$ is calculated using likelihood function.

The main difficulty is that the latent space is not uniformly distributed and sequential poses may be not close on the latest space to each other. Therefore we use a dynamic model, as proposed by Wang et al. [4], in order to achieve smoothed transitions between sequential poses in the latent space. However, there are still some irregularities and discontinuities. Moreover, in the latent space each pose has a certain probability to occur and thus the probability to be drawn as a hypothesis should be dependent on it. For each location in the latent space the variance can be estimated that can be used for generation of the new particles. In Fig. 1(a) the lighter pixels represent lower variance, which depicts the regions of latent space that corresponds to more likely poses.

The additional modification that needs to be done is in the way the optimal configuration is calculated. In the original annealed particle filter algorithm the optimal configuration is achieved by averaging over the particles in the last layer. However, as the latent space is not an Euclidian one, applying this method on ω will produce poor results. The other method is choosing the particle with the highest likelihood as the optimal configuration $\omega_n = \omega_{n,0}^{(i_{max})}$, where $i_{max} = \arg\min_i \left(\pi_{n,m}^{(i)}\right)$. However, this is unstable way to calculate the optimal pose, as in order to ensure that there exists a particle which represents the correct pose we have to use a large number of particles. Therefore, we propose to calculate the optimal configuration in the data space and then project it back to the latent space. At the first stage we apply the $\wp$ on all the particles to generate vectors in the data space. Then in the data space we calculate the average on these vectors and project it back to the latent space. It can be written as follows: $\omega_n = \wp^{-1}\left(\sum_{i=1}^{N} \pi_{n,0}^{(i)} \wp\left(\omega_{n,0}^{(i)}\right)\right)$.

The resulting tracker is capable of recovering after several frames of poor estimations. The reason for this is that particles generated in the latent space are representing valid poses more authentically. Furthermore because of its low dimensionality the latent space can be covered with a relatively small number of particles. Therefore, most of possible poses will be tested with emphasis on the pose that is close to the one that was retrieved in the previous frame. So if the pose was estimated correctly the tracker will be able to choose the most suitable one from the tested poses. At the same time, if the pose on the previous frame was miscalculated the tracker will still consider the poses that are quite different. As these poses are expected to get higher value of the weighting function the next layers of the annealed will generate many particles using these different poses. In this way the pose is likely to be estimated correctly, despite the miss-tracking on the previous frame as shown in Fig. 2. Another advantage of our

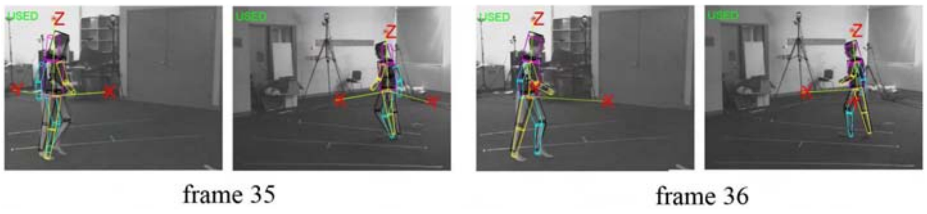


Fig. 2. Losing and finding the tracked target despite the miss-tracking on the previous frame

approach is that the generated poses are, in most cases, natural. In case of CONDENSATION or annealed particle filter , the large variance in the data space, can cause generation of unnatural poses. Poses that are produced by the latent space that correspond to points with low variance are usually natural and therefore the number of the particles effectively used is higher, which enables more accurate tracking.

The problem with such a 2-staged approach is that Gaussian field is not capable to describe all possible poses. As we have mentioned above, this approach resembles using probabilistic PCA in order to reduce the data dimensionality. However, for tracking issues we are interesting to get pose estimation as close as possible to the actual one. Therefore, we add an additional annealing layer as the last step. This layer consists only from one stage. We are using data states, which were generated on the previous two staged annealing layer, in order to generate data states for the next layer. This is done with very low variances in all the dimensions, which practically are even for all actions, as the purpose of this layer is to make only the slight changes in the final estimated pose. Thus it does not depend on the actual frame rate, contrarily to original annealing particle tracker, where if the frame rate is changed one need to update the model parameters (the variances for each layer).

4 Action Classification

The classification of the actions is done based on the sequences of the poses that were detected by the tracker during the performed motion. We use Frèchet distance [6] in order to determine the class of the motion, i.e. walking, kicking, waving etc. The Frèchet distance between two curves measures the resemblance of the curves taking into consideration their direction. This method is quite tolerant to position errors.

Suppose there are K different motion types. Each type k is represented by a model M_k which is a sequence of the l_k+1 latent coordinates $M_k = \{\mu_0, ..., \mu_{l_k}\}$. The GPAPF tracker generates a sequence of $l+1$ latent coordinates: $\Gamma = \{\varphi_0, ..., \varphi_l\}$. We define a polygonal curve P^E as a continuous and piecewise linear curve made of segments connecting vertexes $E = \{v_0, ..., v_n\}$. The curve can be parameterized with a parameter $\alpha \in [0, n]$, where $P^E(\alpha)$ refers to a given position on the curve, with $P^E(0)$ denotes v_0 and $P^E(n)$ denotes v_n. The distance between two curves is defined as

$$F\left(P^{M_i}, P^{\Gamma}\right) = \min_{\alpha,\beta}\left\{f\left(P^{M_i}(\alpha), P^{\Gamma}(\beta)\right) : \alpha\,[0,1] \rightarrow [0, l_k], \beta\,[0,1] \rightarrow [0, l]\right\}$$

where $f\left(P^{M_i}(\alpha), P^{\Gamma}(\beta)\right) = \max\left\{\|P^{M_k}(\alpha(t)) - P^{\Gamma}(\beta(t))\|_2 : t \in [0,1]\right\}$; $\alpha(t)$ and $\beta(t)$ represent sets of continuous and increasing functions with $\alpha(0) = 0$, $\alpha(1) = l_k$, $\beta(0) = 0$, $\beta(1) = l$. The model with the smallest distance is chosen to represent the type of the action.

While in general it is hard to calculate the Frèchet distance, Alt et. al [6] has shown an efficient algorithm to calculate it between two piecewise linear curves.

5 Results

We have tested GPAPF tracking algorithm using HumanEva data set. The data set contains different activities, such as walking, boxing etc. and provides the correct 3D locations of the body joints, such as hips and knees, for evaluation of the results and comparison to other tracking algorithms. We have compared our results to the ones produced by the annealed particle filter body tracker [19] and compared the results with the ones produced by the GPAPF tracker. The error measures the average 3D distance between the locations of the joints that is provided by the MoCap system and by ones that were estimated the tracker [19]. Fig. 3 shows the actual poses that were estimated for this sequence. The poses are projected to the first and second cameras. The first two rows show the results of the GPAPF tracker. The last two rows show the results of the annealed particle filter. Fig. 4.a shows the error graphs, produced by GPAPF tracker (blue circles) and by the annealed particle filter (red crosses) for the walking sequence taken at 30 fps. The graph suggests that the GPAPF tracker produces more accurate estimation. We compared the performance of the tracker with and without the additional annealed layer. We have used 5 double staged annealing layers in both cases. For the second tracker we have added additional

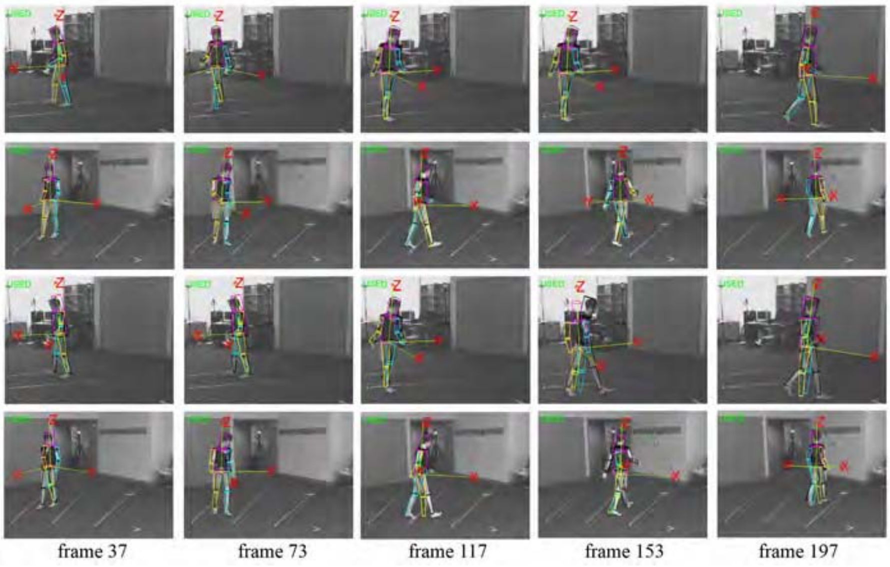


Fig. 3. Tracking results of annealed particle filter tracker and GPAPF tracker. Sample frames from the walking sequence. First row: GPAPF tracker, first camera. Second row: GPAPF tracker, second camera. Third row: annealed particle filter tracker, first camera. Forth row: annealed particle filter tracker, second camera.

single staged layer. The Fig. 4.b shows the errors of the GPAPF tracker version with the additional layer (blue circles) and without it (red crosses); Fig. 5 shows sample poses, projected on the cameras. The improvement is not dramatic. This is explained by the fact that the difference between the estimated pose using only the latent space annealing and the actual pose is not very big. That suggests that the latent space represents accurately the data space.

We have also created a database, which contains videos with similar actions, produced by a different actor. The frame rate was 15 fps. We have manually

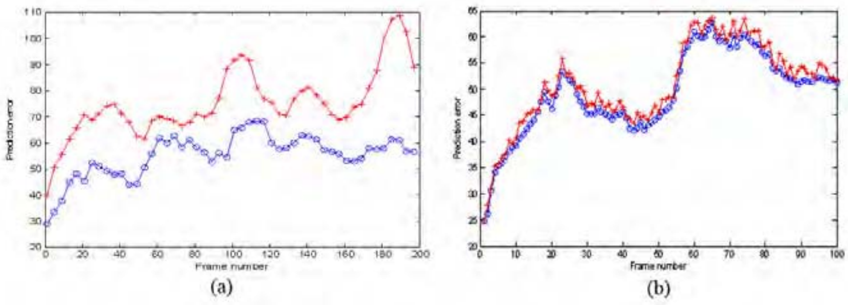


Fig. 4. (a) The errors of the annealed tracker (red crosses) and GPAPF tracker (blue circles) for a walking sequence captured at 30 fps. (b) The errors GPAPF tracker with additional annealing layer (blue circles) and without it (red crosses) for a walking.

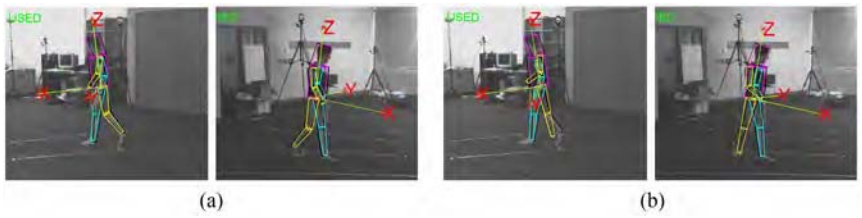


Fig. 5. GPAPF algorithm with (a) and without (b) additional annealed layer

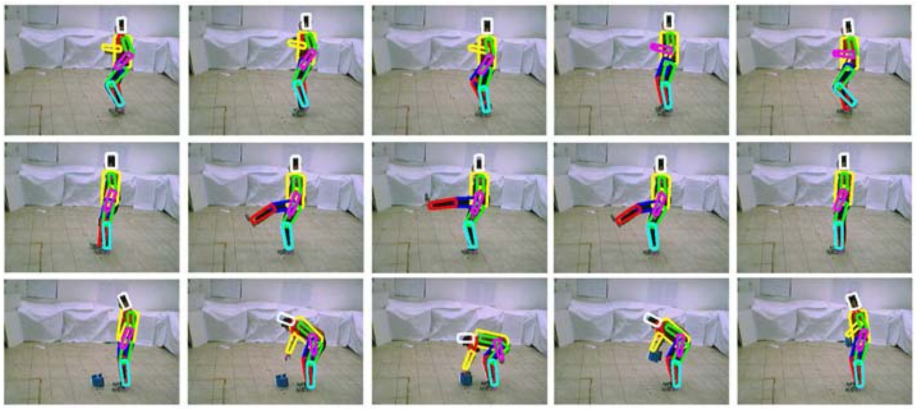


Fig. 6. Tracking results of annealed particle filter tracker and GPAPF tracker. Sample frames from the running, leg movements and object lifting sequences.

marked some of the sequences in order to produce the needed training sets for GPDM. After the learning we validated the results on the other sequences containing same behavior.

5.1 Motion Classification

The classification algorithm was tested on two different data sets. The first set contained 3 different activities: (1) lifting an object, kicking with (2) the left and (3) the right leg. For each activity 5 different sequences were captured. We have used one sequence for each motion type in order to construct the models. The latent space was learned based on the poses in these models (Fig. 1.a). The latent space had a clear and very distinguishable separation between these 3 actions. Therefore, although the results of the tracker contained much noise as shown in Fig. 7, the algorithm was able to perform perfect classification.

The second set contained 5 different activities: (1) hand waving, (2) lifting an object, (3) kicking, (4) sitting down, and (5) punching. Once again 5 different sequences were captured for each activity. The cross validation procedure was

Table 1. The accuracies of the classification for 5 different activities: hand waving, object lifting, kicking, sitting down, and punching. The rows represent the correct motion type; the columns represent the classification results.

	Hand waving	Object lifting	Kicking	Sitting down	Punching
Hand waving	15	0	0	0	5
Object lifting	0	17	0	3	0
Kicking	0	0	20	0	0
Sitting down	0	3	1	16	0
Punching	6	0	20	0	14

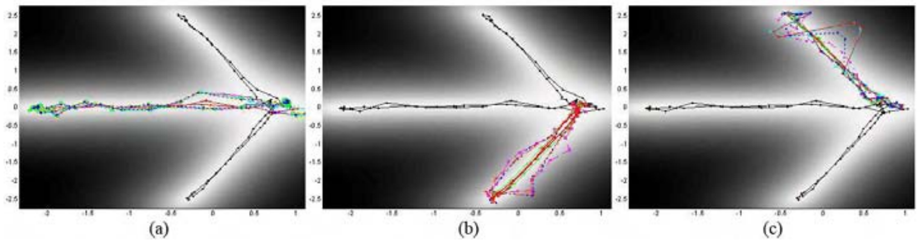


Fig. 7. Tracking trajectories in the latent space for different activities: (a) lifting an object, kicking with (b) the left and (c). On each image the black lines represent incorrect activities, the red line represents the correct one, and other colored lines represent the trajectories produced by the GPAPF tracker.

used to classify the sequences (see Fig. 1.b). The accuracies of the classification, as shown in Table 1, are 75, 85, 100, 80, 70 percent for the above interactions (1)-(5) respectively. The low classification rates of actions involving the hand gestures are due to the similarity of the native actions. The low classification rates of sitting down and object lifting actions are due to the high self occlusions, which caused the tracker to perform wrong estimations of the actual poses.

6 Conclusion and Future Work

We have introduced an approach for articulated body tracking and human motion classification using a low dimensional latent space. The latent space is constructed from pose samples from different motion types. The tracker generates trajectories in the latent space, which are classified using Frèchet distance.

The interesting issue that has not been solved yet is to perform classification of the interactions between multiple actors. While a single persons poses can be described using a low dimensional space it may not be the case for multiple people and the ability to construct a latent space has not been proven yet.

References

1. Christoudias, C.M., Darrell, T.: On modelling nonlinear shape-and-texture appearance manifolds. In: Proc. CVPR, vol. 2, pp. 1067–1074 (2005)
2. Elgammal, A., Lee, C.: Inferring 3d body pose from silhouettes using activity manifold learning. In: Proc. CVPR, vol. 2, pp. 681–688 (2004)
3. Lawrence, N.: Gaussian process latent variable models for visualization of high dimensional data. Information Processing Systems (NIPS) 16, 329–336 (2004)
4. Wang, J., Fleet, D., Hetzmann, A.: Gaussian process dynamical models. Information Processing Systems (NIPS), 1441–1448 (2005)
5. Deutscher, J., Blake, A., Reid, I.: Articulated body motion capture by annealed particle filtering. In: Proc. CVPR, pp. 2126–2133 (2000)
6. Alt, H., Knauer, C., Wenk, C.: Matching polygonal curves with respect to the frèchet distance. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 63–74. Springer, Heidelberg (2001)
7. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. International Journal of Computer Vision 29(1), 5–28 (1998)
8. Sidenbladh, H., Black, M., Fleet, D.: Stochastic tracking of 3d human figures using 2d image motion. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 702–718. Springer, Heidelberg (2000)
9. Rohr, K.: Human movement analysis based on explicit motion models. Motion-Based Recognition 8, 171–198 (1997)
10. Wang, Q., Xu, G., Ai, H.: Learning object intrinsic structure for robust visual tracking. In: Proc. CVPR, vol. 2, pp. 227–233. Madison, WI (2003)
11. Tenenbaum, J., de Silva, V., Langford, J.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
12. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326 (2000)
13. Urtasun, R., Fleet, D.J., Fua, P.: 3d people tracking with gaussian process dynamical models. In: Proc. CVPR, vol. 1, pp. 238–245 (2006)
14. Sato, K., Aggarwal, J.: Recognizing two-person interactions in outdoor image sequences. IEEE Workshop on Multi-Object Tracking (2001)
15. Park, S., Aggrawal, J.: Recognition of human interactions using multiple features in a grayscale images. In: Proc. ICPR, vol. 1, pp. 51–54 (2000)
16. Hongeng, S., Bremond, F., Nevatia, R.: Representation and optimal recognition of human activities. In: Proc. CVPR, vol. 1, pp. 818–825 (2000)
17. Park, J., Park, S., Aggrawal, J.: Video retrieval of human interactions using model-based motion tracking and multi-layer finite state automata. In: Bakker, E.M., Lew, M.S., Huang, T.S., Sebe, N., Zhou, X.S. (eds.) CIVR 2003. LNCS, vol. 2728, Springer, Heidelberg (2003)
18. Deutscher, J., Reid, I.: Articulated body motion capture by stochastic search. International Journal of Computer Vision 61(2), 185–205 (2004)
19. Balan, A., Sigal, L., Black, M.: A quantitative evaluation of video-based 3d person tracking. IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), 349–356 (2005)

Superpixel Analysis for Object Detection and Tracking with Application to UAV Imagery

Christopher Rasmussen

Dept. Computer & Information Sciences
University of Delaware
cer@cis.udel.edu

Abstract. We introduce a framework for object detection and tracking in video of natural outdoor scenes based on fast per-frame segmentations using Felzenszwalb's graph-based algorithm. Region boundaries obtained at multiple scales are first temporally filtered to detect stable structures to be considered as object hypotheses. Depending on object type, these are then classified using *a priori* appearance characteristics such as color and texture and geometric attributes derived from the Hough transform. We describe preliminary results on image sequences taken from low-flying aircraft in which object categories are relevant to UAVs, consisting of sky, ground, and navigationally-useful ground features such as roads and pipelines.

1 Introduction

The problem of segmenting an image into semantically meaningful units is a fundamental one in image processing. Watersheds [1], mean-shift clustering [2], and graph theoretic [3,4,5,6] techniques which group pixels based on proximity, color, texture, and other Gestalt characteristics are powerful tools, but none are silver bullets. Besides practical issues of running time or the necessity of some user interaction [4], the key issue is what objects, if any, correspond to the segmented regions (aka *superpixels*). Given the frequency of oversegmentation, in which objects are broken in multiple superpixels, or the opposite phenomenon, a realistic hope of resolving this issue is only possible in constrained scenarios in which much is known *a priori* about the objects in the scene. A sample of recent work tackling superpixel grouping and classification includes the domains of human body pose recognition [7], categorizing scene surface layout [8], and finding trees, vehicles, and buildings in aerial images [9].

In this paper we take inspiration from such work in proposing a multi-stage framework for object detection and tracking that is based on the output of the graph-based image segmentation algorithm of Felzenszwalb and Huttenlocher [6] (aka the *FH segmenter*). One major advantage of the FH segmenter is its speed: on an Intel Core2 Extreme X6800, it segments 360×240 color images at over 30 Hz, compared to several minutes per image using spectral graph cut methods

[3,5]. This makes it practical to use for real-time video. Thus, a chief novelty of our approach is that we can analyze superpixels over time for *consistency* according to the intuition that temporally transient borders are likely spurious. Candidate superpixels that pass this restrictive test are further examined using region characteristics such as color, texture, and shape to see if they match any of the objects we are searching for.

The task area we apply the framework to here is navigation and control for unmanned aerial vehicles (UAVs) flying at very low altitudes (under 500 m). There are many possible image processing tasks for UAVs; we focus on horizon detection for attitude determination and recognizing ground features for navigation. At such low altitudes, terrain relief means that the horizon may be quite jagged (see Fig. 1) rather than straight as most work assumes [10,11,12]. Raw segmentations of mountainous images are shown in the classification approach of [13], but there is no discussion of explicit recovery of the boundary curve in a single image or tracking it over time.

Other object categories of interest besides sky and ground are navigationally-useful linear features along the ground, which we term *trails*. These include engineered artifacts such as dirt and paved roads and above-ground pipelines, but also natural phenomena like rivers and streams. While vision-based road segmentation is a well-studied topic for unmanned ground vehicles [14,15,16], the relatively large size and stable centroid of the road region in their camera view allows a robust approach to shape estimation from edges and easy sampling of on-road pixels that is not possible from a low-altitude UAV perspective. From the air, trail pixels do not dominate the image, making mere detection of the trail among many other features a difficult aspect of the problem. Also, the camera platform is often anything but stable due to wind gusts and pitching and rolling as the UAV maneuvers, causing the horizon and trail to go in and out of view. Unfortunately, because of the wide range of shapes and appearance characteristics possible, aerial trail finding is not amenable to traditional machine learning-based object detection methods such as the face- and person-finders of [17,18].

Image processing techniques for road or pipeline tracing in higher altitude imagery [19,20,21] also do not carry over directly. The low-altitude UAV perspective is unlikely to be straight down, leading to perspective effects like non-parallel road edges. On the positive side, though, enough pixels can typically be resolved within the trail region to gather detailed appearance statistics. Examples of existing techniques that are more tracing-oriented than region-based include [22], which demonstrates tracking of a short, straight runway with relatively high contrast from a UAV, and [23] from the same authors, which tracks video of a slightly curving canal. The tracker in the latter paper initializes automatically under the assumption that the ground feature is homogeneous, only gently curved, and occupies a fairly large fraction of the image. [13] has some results on finding straight roads in low-altitude aerial images using a Hough transform on static images only, with no tracking.

The organization of the paper is as follows. First, we detail our methods for discriminating between the sky and ground objects reliably and tracking the border between them. Second, we discuss preliminary results on extending our superpixel analysis method to trail detection for several trail types. Finally, we present a separate section of results processed offline on three diverse sequences: a winding canyon road in the desert, a flyover of a section of the Alaska pipeline, and a helicopter flight along a mountain stream.

2 Sky/Ground Discrimination

The FH segmenter works by greedily splitting regions when the intensity difference along the candidate child regions' border exceeds their internal intensity variations by a certain threshold (color image segmentations result from merging the segmentations of their channels). Two key parameters govern the operation of the segmenter: k, which sets the threshold for splitting, with larger k leading to a preference for the retention of larger regions; and min, which sets an absolute minimum on the area of any region that is output.

Applying Felzenszwalb's code for the FH segmenter to the canyon sequence images in the first row of Figure 1 with standard parameters $k = 100$ and min = 100 (explanation below) yields the segmentations shown in the second row of the figure. The colors of the segmentation are randomly generated to show superpixel memberships. While the sky-ground border is clearly visible, it is but one of many edges generated by this oversegmentation. Because the object we are looking for, sky, is expected to be a large region, we can obtain a cleaner segmentation with a different set of FH segmenter parameters. We bias the segmenter to find much larger regions by changing k to 500 and setting min = 5% of the area of the image. This results in the segmentations shown in the third row of the figure. Over all of our data, these parameters yield 2-5 superpixels in almost all images of the canyon sequence. The sky-ground border is generally very stable, with only transient "extra" regions due mostly to cloud formations, hazy distant terrain, and intermittent video transmission interference when the capture device was not onboard the aircraft (as seen in the right column of Figure 1).

We were able to temporally filter these superpixel borders directly to get quite good results for the sky-ground dividing line, but because of the mountainous terrain in the canyon sequence and the steep banking of the UAV, it is not always clear from geometry alone which side of the line is sky and which is ground. It is thus desirable to differentiate the two based on region characteristics such as color. There are many options here, but we chose to use the *surface layout* classifier of [8,24]. This method divides images into geometric classes corresponding to three kinds of surfaces: *support* (aka ground), *vertical*, or *sky*. Within the vertical class are several subclasses corresponding to different normal directions if the surface is planar and the designations *solid* or *porous* if it is not. A particular pixel is labeled by a learned classifier using features such as as image location, color, texture, and any association with vanishing points. An important aspect of the classification process is that pixels are not labeled independently, but rather in groups by superpixel. They use the FH segmenter

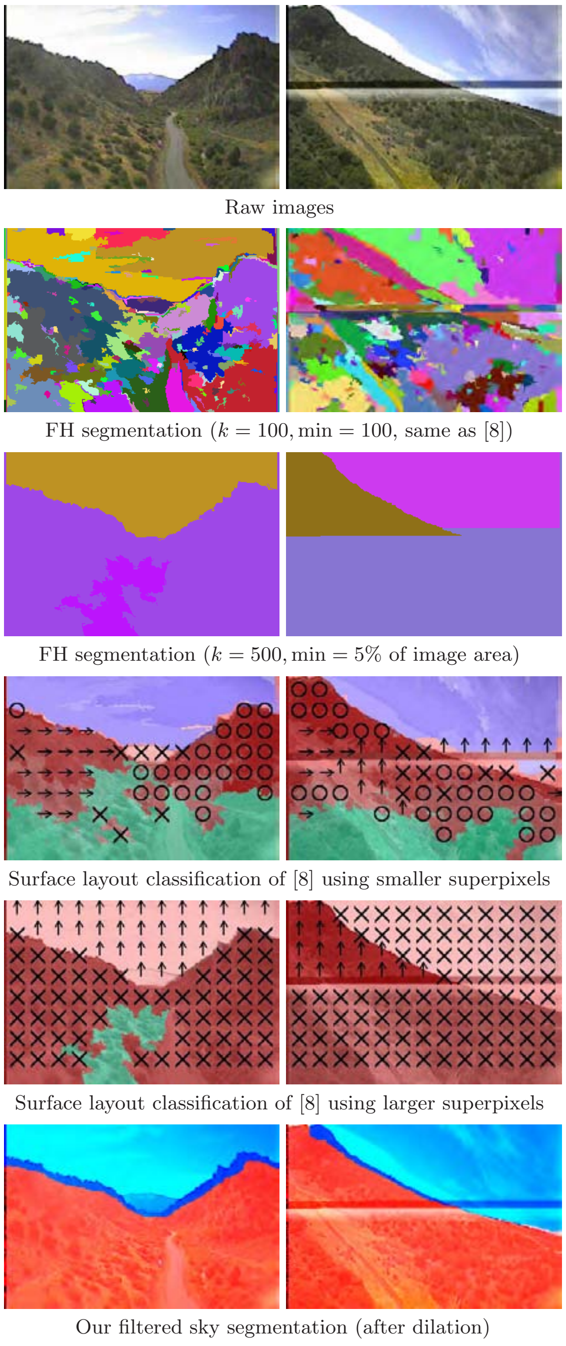

Raw images
FH segmentation ($k = 100$, min $= 100$, same as [8])
FH segmentation ($k = 500$, min $= 5\%$ of image area)
Surface layout classification of [8] using smaller superpixels
Surface layout classification of [8] using larger superpixels
Our filtered sky segmentation (after dilation)

Fig. 1. Sky-ground object detection steps for two images. The horizontal bar in the right image is a wireless video transmission artifact.

with the parameters $k = 100$ and min $= 100$ for all of their results, which is why we show what is obtained with these parameters in the second row of Figure 1).

Feeding these to the author-provided implementation of the scene layout algorithm (the later version from [24]) with the outdoor-trained version of their classifier yields the classifications shown in the third row of the figure. The scene layout labels are indicated by tinting: blue for sky, red for vertical, and green for support. Subclasses within vertical are indicated by overlaid icons: "O" for porous, "X" for solid, or an arrow for the normal direction of planar surfaces. One can see that the support and vertical class labels, as well as vertical's subclass labels, are mostly non-informative. The sky label is only reasonably accurate, with a sample from the entire sequence indicating that nontrivial fractions of the sky are mislabeled in over 50% of the frames.

We were able to significantly improve the accuracy of the scene layout sky labels by rerunning the segmenter with the parameters that yield larger regions. The new scene layout labels with these parameters are shown in row four of the figure. The erroneous results shown are not representative: the scene layout classifier actually does quite well overall. A manual tally of the entire canyon sequence (2500 frames) shows that a majority of sky pixels are mislabeled in only 9% of the images—most commonly as vertical—with a non-zero minority of the sky mislabeled in about 2%. Practically none of the ground pixels are mislabeled as sky. These numbers agree quite well with the confusion matrix in [24].

Our point in showing error examples is to motivate our use of temporal filtering: simply trusting a single-image classifier, however good, will occasionally result in nonsensical output. We know more: the sky region does not disappear and reappear from frame to frame—it has continuity of appearance and shape.

Before describing our tracking framework, we should mention that we also modified how the scene layout classifier's output is interpreted. We found that misclassifications of the sky as vertical most often resulted from random fluctuations in the two class likelihoods when they were nearly the same magnitude. We observed that the sky likelihood distribution was quite bimodal, and that instead of choosing the class with the highest likelihood for a superpixel, setting a low threshold on the sky likelihood for classification as simply sky or not sky eliminated almost all such errors.

2.1 Temporal Filtering

Going beyond performing segmentations and scene layout classifications *de novo* on each frame is fairly straightforward. Let $\mathbf{B}_t$ be the sky-ground boundary obtained after the steps of the previous subsection for frame t. $\mathbf{B}_t$ is simply represented as a set of points along the border of the sky region (after some smoothing and cropping to eliminate image edge effects and capture artifacts). There are no restrictions on connectivity or indeed any attempt made to parametrize the curve.

We want to mitigate the effects of transient classification errors and segmentation artifacts by temporal smoothing. We choose a majority filter: given n successively segmented borders $\mathbf{B}_{t-n+1}, \ldots, \mathbf{B}_t$ registered with one another (i.e., motion stabilized), an aberrant border segment or even an entire missing border can be detected by treating each $\mathbf{B}_t$ as voting for a particular hypothesis and only accepting those that achieve consensus. We do this by spreading each border out with a linear falloff, summing, and thresholding.

In order to register each sky-ground boundary $\mathbf{B}_t$ with its predecessor $\mathbf{B}_{t-1}$, we assume that a 2-D rigid transformation is a good approximation of horizon movement over a short time scale (in fact, we found that translation alone is good enough). RANSAC [25] does quite well at registering successive curves in the face of possible large outliers. Over the long term, nonlinear deformation occurs as new mountains come into view; this is handled implicitly with the finite "memory" of the border voter buffer, with $n = 5$.

Example filtered sky-ground regions are shown in the last row of Figure 1). Some small misregistrations remain because of the implicit lag of the majority filter.

3 Trail Detection

Besides the control benefits of horizon orientation information for a UAV, finding the sky region is also critical for successful trail detection. First, since we know the trail is on the ground, sky pixels can be masked out, reducing the image area to search and helping efficiency. Second, accuracy is also improved: since the horizon line is often the highest-strength edge in the image, removing it makes voting techniques for identifying trail edges less susceptible to distraction.

We make several simplifying assumptions. First we assume that there is a trail visible over nearly all of the image sequence—we are not trying to decide *whether* there is a trail or not, only to find and parametrize it in a way that could be passed to a flight control system. This does allow for the trail to briefly disappear from view as the aircraft maneuvers. A second assumption is that there is only one trail visible over the sequence. Parallel roads, forks, and intersections are not accounted for, as they introduce ambiguity about which trail is to be followed. Finally, we assume that the off-trail terrain is *natural* rather than urban, as our primary means of recognizing trails is through the rarity of the structural regularity associated with them.

Our approach to trail detection is similar to that for sky segmentation: first the image is segmented into superpixels, of which we expect trail edges to be relatively stable. Applying a region classifier along the lines of the scene layout method described above is problematic because trails have much more appearance variation than sky and we do not know *a priori* which kind of trail we are tracking. Rather, we narrow the candidates using a geometric test based on the intuition that trails often have linear or smoothly curving borders, unlike the natural background that occupies the rest of the image. Surprisingly, even the river trail shown in Figure 2, which has somewhat a stochastic boundary shape, fits this profile for much of its length.

This test currently consists of a Hough transform applied to the superpixel borders (using a different set of segmentation parameters $k = 200$, $\min = 1000$).

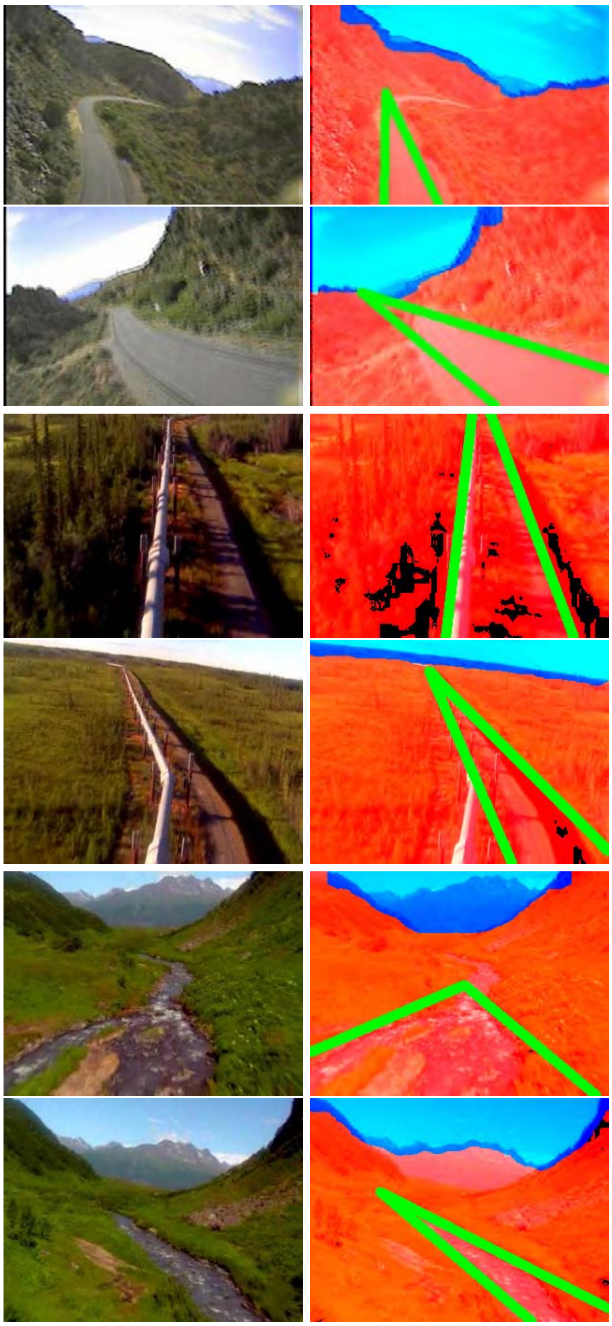


Fig. 2. Sample object detections. The envelopes of trail detections are demarcated in green; dark spots in the ground regions are compression artifacts.

The top line candidates associated with the trail should cluster by vanishing point, which is trackable with filtering. Using the superpixel borders rather than low-level edge features such as Canny or gradient magnitude allows more sophisticated grouping information such as color and texture to play a role, if at some cost of smoothness.

4 Results

Our results are summarized with sample object detections on all three sequences in Figure 2. Each sequence exhibits artifacts of some kind: the canyon sequence has a fair degree of video transmitter interference, and the pipeline and river sequences have significant compression artifacts. Nonetheless, the algorithm presented here performs extremely well on all three at sky-ground border tracking. Trail detection results are somewhat preliminary, as further filtering could be done to counter noise in the segmentation, but still quite promising, especially for the pipeline sequence. Counter to one of our assumptions above, there is ambiguity in the pipeline sequence about what the trail is: the pipe itself, the road, or both? This is reflected in the trail object finder shifting around a bit between these alternative interpretations.

We note that the sky-ground tracker's ability to handle the disappearance and reappearance of the horizon is shown in the pipeline images. Also seen in the river sequence is the phenomenon of flipping back and forth between sky and ground classifications for distant mountains, which through haze have a similar color to the sky. The filtering mechanism smooths these transitions.

5 Conclusion

We have demonstrated a promising approach to object detection and tracking in video that is based on appearance and geometric analysis of quickly-segmented superpixels. The competence of the technique was shown through application to several realistic UAV tasks and over diverse image sequences.

Our current work focuses on the trail detection component. While the linear Hough transform is a reasonable start, further refinement to estimate trail curvature and smoothness is necessary. Rather than approximating the trail boundaries with parametric curves as we currently do, we are investigating how to extract irregular boundaries directly from the segmentation.

Acknowledgments

Thank you to Clark Taylor of Brigham Young University for supplying the canyon UAV imagery. This material is based upon work supported by the National Science Foundation under Grant No. 0546410.

References

1. Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Trans. Pattern Analysis and Machine Intelligence 13(6) (1991)
2. Comaniciu, D., Meer, P.: Mean shift analysis and applications. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (1999)
3. Shi, J., Malik, J.: Normalized cuts and image segmentation. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (1997)
4. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: Proc. Int. Conf. Computer Vision (2001)
5. Fowlkes, C., Martin, D., Malik, J.: Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (2003)
6. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. Int. J. Computer Vision (2004)
7. Mori, G., Ren, X., Efros, A., Malik, J.: Recovering human body configurations: Combining segmentation and recognition. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (2004)
8. Hoiem, D., Efros, A., Hebert, M.: Geometric context from a single image. In: Proc. Int. Conf. Computer Vision (2005)
9. Kaufhold, J., Collins, R., Hoogs, A., Rondot, P.: Recognition and segmentation of scene content using region-based classification. In: Proc. Int. Conf. Pattern Recognition (2006)
10. Ettinger, S., Nechyba, M., Ifju, P., Waszak, M.: Vision-guided flight stability and control for micro air vehicles. In: Proc. Int. Conf. Intelligent Robots and Systems (2002)
11. McGee, T., Sengupta, R., Hedrick, K.: Obstacle detection for small autonomous aircraft using sky segmentation. In: Proc. IEEE Int. Conf. Robotics and Automation (2005)
12. Cornall, T., Egan, G., Price, A.: Aircraft attitude estimation from horizon video. IEE Electronics Letters 42(13) (2006)
13. Todorovic, S., Nechyba, M.: A vision system for intelligent mission profiles of micro air vehicles. IEEE Trans. Vehicular Technology 53(6) (2004)
14. Stein, G., Mano, O., Shashua, A.: A robust method for computing vehicle egomotion. In: Proc. IEEE Intelligent Vehicles Symposium (2000)
15. Southall, B., Taylor, C.: Stochastic road shape estimation. In: Proc. Int. Conf. Computer Vision, pp. 205–212 (2001)
16. Rasmussen, C.: A Hybrid Vision + Ladar Rural Road Follower. In: Proc. IEEE Int. Conf. Robotics and Automation (2006)
17. Viola, P., Jones, M.: Robust real-time object detection. Int. Workshop on Statistical & Computational Theories of Vision (2001)
18. Viola, P., Jones, M.: Detecting pedestrians using patterns of motion and appearance. In: Proc. Int. Conf. Computer Vision (2003)
19. Geman, D., Jedynak, B.: An active testing model for tracking roads from satellite images. IEEE Trans. Pattern Analysis and Machine Intelligence 18 (1996)
20. Yuille, A., Coughlan, J.: Fundamental limits of Bayesian inference: Order parameters and phase transitions for road tracking. IEEE Trans. Pattern Analysis and Machine Intelligence 22(2), 160–173 (2000)

21. Perez, P., Blake, A., Gangnet, M.: Jetstream: Probabilistic contour extraction with particles. In: Proc. Int. Conf. Computer Vision, pp. 524–531 (2001)
22. Frew, E., McGee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., Sengupta, R.: Vision-based road following using a small autonomous aircraft. In: IEEE Aerospace Conference, IEEE Computer Society Press, Los Alamitos (2004)
23. Rathinam, S., Kim, Z., Soghikian, A., Sengupta, R.: Vision based following of locally linear structures using an unmanned aerial vehicle. In: IEEE Conference on Decision and Control, IEEE Computer Society Press, Los Alamitos (2005)
24. Hoiem, D., Efros, A., Hebert, M.: Recovering surface layout from an image. Int. J. Computer Vision (to appear, 2007)
25. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)

Nonuniform Segment-Based Compression of Motion Capture Data

Yi Lin and Michael D. McCool

University of Waterloo, RapidMind Inc.

Abstract. This paper presents a lossy compression method for motion capture data. Each degree of freedom of a motion clip is smoothed by an anisotropic diffusion process and then divided into segments at feature discontinuities. Feature discontinuities are identified by the zero crossings of the second derivative in the smoothed data. Finally, each segment of each degree of freedom is approximated by a cubic Bézier curve. The anisotropic diffusion process retains perceptually important high-frequency parts of the data, including the exact location of discontinuities, while smoothing low-frequency parts of the data. We propose a hierarchical coding method to further compress the cubic control points. We compare our method with wavelet compression methods, which have the best compression rates to date. Experiments show that our method, relative to this work, can achieve about a 65% higher compression rate at the same approximation level.

1 Introduction

Motion capture data is now widely used in games, simulations, and animation. A motion is represented in motion capture data as an ordered sequence of frames. Each frame specifies all positions, orientations, and joint angles for a pose at a certain point in time. Each pose parameter is called a *degree of freedom (DOF)*. How to efficiently store such data in usable form with limited memory resources is still a challenging problem. Compression is possible for motion capture data because of two properties: *temporal coherence* and *DOF correlations* [1]. Temporal coherence is present in motion capture data because all DOFs are sampled simultaneously from the same continuous physical motion. Due to the physical limits inherent in its origin, motion data can be approximated by continuous functions, such as splines, wavelets, or linear dynamical systems, which can capture temporal coherence. Correlation between different DOFs is caused by the fact that all DOFs are related to the same physical structure. Also of importance is the fact that some DOFs are more important perceptually than others. Therefore, it is possible to remove redundancy between DOFs using dimensionality reduction techniques and also to omit some uninfluential DOFs without significantly affecting the perceptual accuracy.

There are two kinds of compression: lossless and lossy. Lossless compression has zero error but cannot achieve high compression rates. Lossy compression techniques can achieve higher compression rates but always involve a tradeoff between compression rate and error. The goal of a lossy compression method such as the one we present here is to achieve the best ratio between compression rate and error. Compressing in both temporal and DOF spaces can achieve better compression rate than temporal compression only [2]. However, compressing individual DOFs makes reuse of the motion data and update of the database easier. Beaudoin *et al* presented a wavelet compression method [1] based on cubic interpolating biorthogonal wavelets which exploited temporal coherence only. This method has achieved the best compression rate relative to error so far, so we use it as our baseline method for performance comparison.

In this paper, we present a lossy compression method for motion data clips. We first divide motion data in each DOF into segments at feature discontinuities. We define feature discontinuities by zero crossings of the second derivative after an anisotropic smoothing process has suppressed less important detail. The segments bounded by the feature discontinuities can be approximated by cubic Bézier splines. Based on the hierarchical structure generated from the anisotropic diffusion smoothing process, we further code the cubic control points using a hierarchical coding method. We exploit temporal coherence at each DOF, possibly achieving a different compression rate for each.

The specific contributions of this paper are as follows. We present a simple compression method which achieves very high compression rates compared with previous work. We smooth motion data using anisotropic diffusion, which avoids the influence of noise on the compression rate, and we segment a motion non-uniformly and hierarchically based on its intrinsic structure. We do not rely on heuristic parameters as previous work does. Heuristic parameters may cause the system to fail when the parameter selections are not appropriate. Unlike some other previous work based on exploiting inter-DOF correlation, the motion clips in our compressed database can be decompressed independently. Updating the database also will not influence the compression of unchanged data. Finally, the decompression is fast enough for real-time applications. It is very easy to evaluate the motion at any point in time directly from our compressed representation.

2 Related Work

Compression operates by removing redundant information. Motion data compression exploits redundant data across three dimensions: the temporal dimension, the DOF dimension, and the motion clip dimension [2].

For motion capture data, decorrelation of temporal redundancy can be done by wavelet transformations and decorrelation between DOFs or motion clips can be done with principal component analysis (PCA). Guskov and Andrei [3] encoded differential wavelet coefficients to compress an animation sequence. Beaudoin *et al* proposed a modified wavelet technique [1] with properties well-suited to motion data. They worked directly with joint angles and use a cubic interpolating spline wavelet basis. Liu and McMillan [4] compress the raw 3D marker positions obtained from motion capture using PCA. Temporal redundancy is then exploited by adaptively fitting cubic splines to the reduced-basis coefficients and only storing the keyframes for the resulting cubic splines. Other temporal simplifications have been used to extract key poses in an animation sequence [5,6], space-time optimization [7], and motion editing [8,9]. Ibarria and Rossignac [10] proposed a predictor/corrector method to exploit temporal coherence between frame meshes.

Compression can also exploit redundancy between DOFs. The behavior of a large number of DOFs can often be expressed relative to the behavior of a lower-dimensional set of data. Safonova *et al* [11] used a low-dimensional space to represent a high-dimensional human behavior. Their work showed that 10-20 DOFs can represent a motion of 40-60 DOFs for a typical human skeleton model. Representations of poses in a reduced dimensional space have been proposed for applications other than compression, including animation retrieval [12], motion editing [13,14], and motion synthesis and texturing [15,16,17,18]. PCA can also compress motions by exploiting inter-DOF redundancy. Liu and McMillan [4] used PCA to extract a reduced marker set which can represent a full body pose. PCA can be used to compress shapes or animations. PCA compresses shapes by finding portions of the mesh that move rigidly and only encoding the rigid transformation and residuals [19,20].

Compression over the motion clip dimension is useful when the database has many related motion clips. Arikan [2] applied clustered PCA to compress linearly related motion clips. They connected all motion clips in the database to a long sequence. They uniformly divided this sequence into segments of same length. Then they exploited both joint correlations and time coherence by using PCA for each segment. Instead of using joint angles directly, virtual markers computed from joint angles are used as an internal representation. Since joint angles are required by current game and simulation engines, extra time and storage are needed for conversion back from this representation, however. This method has a good compression rate but requires a complicated and expensive compression procedure. If the database contains too many linearly unrelated motions, which is a common case, then clustered PCA among motion clips may also produce artifacts.

Relative to previous work, we are most comparable with the wavelet approaches, since we only exploit temporal redundancy. However, this has advantages since it makes database update and access easier. We will compare our compression rates with the best previous wavelet approach [1] as well as with a Haar wavelet approach.

3 Nonuniform Segment Compression

We assume a motion clip can be regarded as a d-channel sequence of length n. The value d specifies the number of degrees of freedom, including the translation and rotation of the root and rotations for each joint. The value n specifies the number of frames. We assume the motion is sampled at regular intervals and the number of DOFs does not change from frame to frame or between motion clips. We model the data for each DOF as a sequence of values $M = \{m_1, m_2, \cdots, m_n\}$, which can be interpreted as sampled curve. We segment this curve into variable-length segments at its own natural discontinuities and use a cubic Bézier curve to approximate each segment. The boundary points are indicated by zero-crossings of the second derivative.

Motion databases often have a great deal of noise. To avoid noise impacting the segmentation result, we smooth each DOF curve while maintaining the position of the boundary points between segments using anisotropic diffusion. This smoothing removes boundary points selectively, so the number of segments decreases monotonically as we go to coarser scales.

3.1 Motion Curve Smoothing

For motion curve smoothing, we use the Perona-Malik anisotropic diffusion filter [21], which was originally proposed as a noise reduction algorithm for image edge detection. The Perona-Malik filter is a diffusion process that encourages intra-region smoothing while inhibiting inter-region smoothing. The smoothing process can avoid the blurring and localization problems of filters based on convolution. It has the useful property of maintaining the positions of natural discontinuities across scales.

The continuous form of the 1D anisotropic diffusion filter applied to a continuous signal $f(x)$ is

$$\frac{\partial}{\partial s} f(x,s) = \frac{\partial}{\partial x} \cdot \left(c(x,s) \frac{\partial}{\partial x} f(x,s) \right). \tag{1}$$

The value s is the scale. The function c is a conductance function and is a monotonically decreasing function of the first derivative. We use the following conductance function:

$$c(x,s) = \exp\left(- \left(\kappa^{-1} \frac{\partial}{\partial x} f(x,s) \right)^2 \right). \tag{2}$$

We discretize Equation 1 over the sequence M. By replacing the scale s with the number of iterations $\sigma = \lambda^{-1} s$, we get the following implementation:

$$\begin{aligned} m_i^{\sigma+1} &= m_i^\sigma + \lambda c(i+1,\sigma)(m_{i+1}^\sigma - m_i^\sigma) \\ &\quad -\lambda c(i-1,\sigma)(m_i^\sigma - m_{i-1}^\sigma). \end{aligned} \tag{3}$$

The boundary conditions are $m_i^0 = m_i$, $m_1^\sigma = m_1$, and $m_n^\sigma = m_n$. The value i is the frame number and satisfies $1 \leq i \leq n$. For stability, we must have $0 \leq \lambda \leq 1/4$. The conductance values are conceptually interdigitated with the smoothed signal with c_i between m_i and m_{i+1}. The value $c(i,\sigma)$ is given by $g(m_{i+1}^\sigma - m_i^\sigma)$, where the function g takes the form given by Equation 2, but with the difference given replacing the derivative.

The value κ is called the *diffusion constant* which controls conduction. The value κ can be set as a constant, but a too-small κ may cause staircases in the smoothed result and slow convergence, while a too-large κ may cause diffusion across boundaries. The "noise estimator" proposed in [22] could be used for a more appropriate κ setting. We use the Canny noise estimator which computes a histogram of the absolute values of the gradient and sets κ to the 90% percentile at each iteration.

We use the distortion error proposed in [1] to control the number of iterations. The distortion error is defined as:

$$\epsilon_x = \sqrt{\frac{1}{n} \sum_{j=1}^{p} \sum_{i=1}^{n} (x_i - x_i^{'})^2 \frac{l_j}{l}}. \tag{4}$$

The value x_i is the 3D position of the endpoint of each bone and $x_i^{'}$ is the 3D position of each such endpoint reconstructed from the compressed data. The value l_j is the length of the joint j, and $l = \sum_{j=1}^{p} l_j$. The value $\frac{l_j}{l}$ normalizes the influences of the lengths of the joints. In other words, this is a weighted error metric where longer bones are given more weight. The value p is the number of joints. For each joint j, we compute the error

$$\epsilon_j = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_i^{'})^2\frac{l_j}{l}. \tag{5}$$

When $\epsilon_j > w_j\epsilon_x^2/p$, the iterations for joint j are halted. The values w_j are the weights assigned to each joint, which have the porperties: $0 \le w_j \le 1$ and $\sum_{j=1}^{p} w_j = 1$.

3.2 Motion Curve Segmentation

To extract the boundary points of segments, we apply the 1D Canny edge detector [22] to the smoothed curves. The Canny edge detector detects boundaries at the zero-crossings of the second derivative of data when the gradient magnitude is also above some threshold α, i.e., $\frac{\partial^2}{\partial x^2}f(x,s) = 0$ and $|\frac{\partial}{\partial x}f(x,s)| \ge \alpha$. The positions of the zero crossings of the second derivative are invariant under anisotropic diffusion and so can be aligned across scales. Coarser scales simply eliminate weaker boundary points. An example is shown in Figure 1.

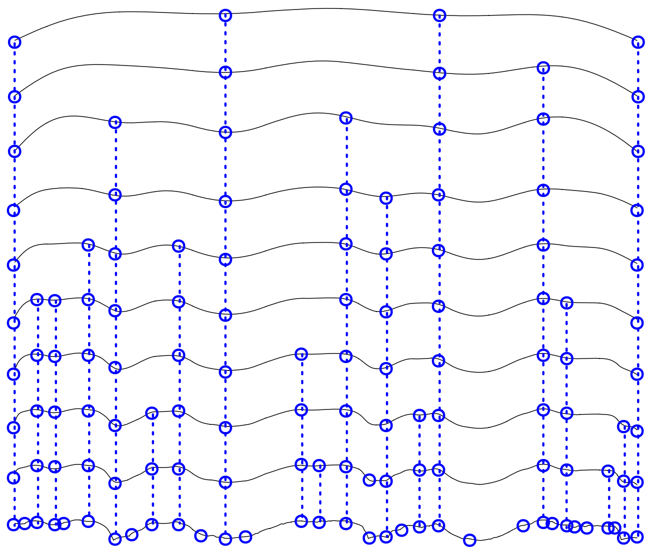


Fig. 1. Anisotropic diffusion smoothing of a motion curve. The boundary points are lined up at the same positions in different scales.

At each scale σ, the boundary points divide the original data $M = \{m_1, \cdots, m_n\}$ into k segments with $1 \le k \le n$. The jth $(1 \le j \le k)$ segment $M[j : j + l_j] = \{m_j, \cdots, m_{j+l_j}\}$ is a subsequence of length l_j.

The anisotropic diffusion process inhibits smoothing across regions with large values of the first derivative. The segmentation strategy identifies boundaries with zero values of the second derivative. A cubic Bézier curve is enough to approximate each segment with a very small approximation error. The segment $[j : j+l_j]$ is fitted to a cubic Bézier curve and is represented by a vector including the length and three cubic control points. The last control point of one segment is the first control point of the next segment, except for the first segment and the last segment of the curve. Applying this process to all DOFs constructs a multichannel compressed sequence.

4 Hierarchical Coding

A motion curve of n frames is represented using $3k + 12$ cubic control points approximating each of the segments identified by the algorithm given in Section 3, where the value k is the number of segments. The entire sequence is approximated with C^0 continuity. If we quantize each control point to 16 bits, the original data data requires $16n$ bits of storage is compressed to $16 \times (4k + 1) = 64k + 16$ bits, including $3k + 1$ bits to store the control points and $8k$ bits to store the offsets of each segment. The compression rate is therefore $n/(4k+1)$. However, the anisotropic diffusion algorithm also generates a natural scale-space hierarchy, and we can exploit this to gain further compression efficiency.

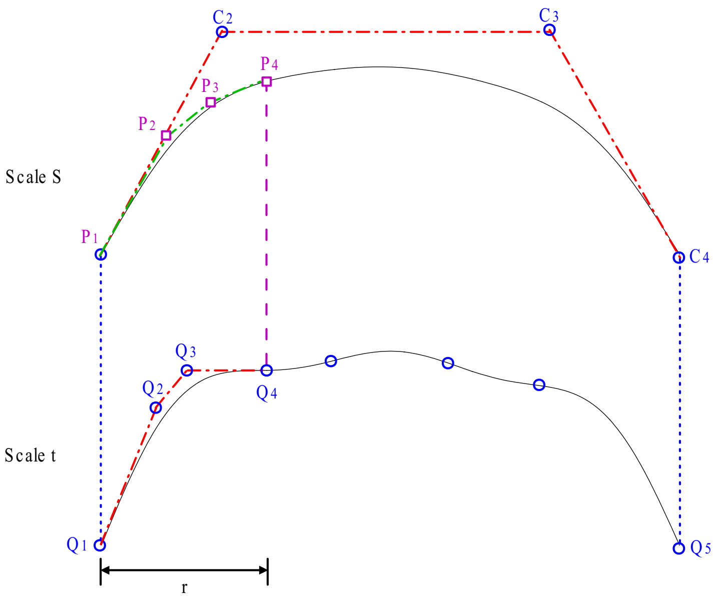


Fig. 2. Hierarchical coding

The hierarchical coding method is illustrated in Figure 2. Basically, if we have a coarse scale curve S, we can code the segment curves of a fine scale t relative to S in fewer bits. Suppose scale S has one segment with 4 control points $C_0 = P_1$, C_2, C_3, and C_4. In Figure 2, the scale t has five segments. The number of points in t should be no less than the number of points in S. We can estimate the control points Q_1, Q_2, Q_3, Q_4 in the first segment of scale t using P_1, P_2, P_3, and P_4. Then we only have to code the differences between the estimated values and the true values, and usually the differences can be encoded in fewer bits.

Since the points P_i $(1 \leq i \leq 4)$ are very close to the points Q_i, we can use the difference P_i, $\delta_i = Q_i - P_1$, and the offset r from Q_4 to Q_1, to represent Q_i. The point P_i can be computed by the points P_1, C_2, C_3, and C_4. Each δ_i is small enough to be stored in 8 bits (or fewer, using more sophisticated variable-rate schemes, but this can be done in a postprocess). Our experiments have shown that there are very few cases that each of these values cannot fit in 8 bits. If the value cannot fit in 8 bits, we use an extra small space to save it.

We code all other segments in scale t using the same method. To represent any scale with k segments, the compressed data includes 4 control points in the coarsest scale S which require 4×16 bits of storage, the differences δ which require $(3k + 2) \times 8$ bits of storage (in which 2×8 bits are used to store the starting and ending points Q_1 and Q_5), and the offsets r which require $k \times 8$ bits of storage. The total required storage is $32k + 80$ bits. The compression rate achieved by this hierarchical coding method, not even using variable-rate coding, is therefore $(64k + 16)/(32k + 80)$.

5 Results

We tested our implementation using the CMU motion capture database [23], which contains 4 million frames (about 9.3 hours sampled at 120Hz). The database contains 2493 AMC files ranging in length from about 100 frames (0.83 seconds) to 23000 frames (191.67 seconds). The database consists of various kinds of motions, including walking, running, kicking, jumping, boxing, dancing, and gymnastics. The uncompressed database requires 3.6GB of storage. We used the ASF skeleton file, which has 62 degrees of freedom. We ran all experiments on a machine with 512MB of memory and a 3GHz Intel Pentium 4 processor.

We use two baseline methods for performance comparisons. The first baseline is the cubic interpolating bi-orthogonal wavelet compression (BWC) method [1]. BWC is a reasonable baseline method because it is very recent work and has the best results for a temporal coherence scheme to date. Unfortunately, this paper did not present many examples suitable for direct comparison. Therefore, we also use another baseline wavelet method HWC similar to BWC. HWC transforms the original motion data using the simpler Haar wavelet basis but uses the same coefficient selection method as BWC.

A wavelet coefficient selection algorithm [1] determines how to quantize coefficients optimally. It selects dn/r coefficients from among the dn wavelet coefficients, taking all DOFs into account. Other wavelet coefficients are clamped to zero. A heuristic simulated annealing method is used to compute the vectors $c \in [0, n]^d$ and $\sum_{j=1}^{p} c_j = dn/r$. The value c_j is the number of wavelet coefficients that are selected for compressing the

Table 1. Compressed size and compression rates using BWC, HWC, and NSC. The uncompressed running motion takes 35.8KB of storage (148 frames). The uncompressed jumping motion takes 505KB of storage (2085 frames).

	ϵ_x	BWC		HWC		NSC	
Running	1.4	0.75	48:1	0.8	45:1	0.51	70:1
	0.96	0.94	38:1	0.99	36:1	0.66	54:1
	0.58	1.24	29:1	1.43	25:1	0.85	42:1
	0.26	2.18	16:1	2.56	14:1	1.23	29:1
	0.08	5.08	7.0:1	6.07	5.9:1	2.86	12.5:1
Jumping	0.67	9.97	51:1	10.31	49:1	7.01	72:1
	0.45	13.5	37:1	15.3	33:1	9.02	56:1
	0.29	18.4	27:1	21.96	23:1	11.22	45:1
	0.14	32.9	15:1	42.08	12:1	15.78	32:1
	0.05	79.2	6.4:1	91.82	5.5:1	34.59	14.6:1

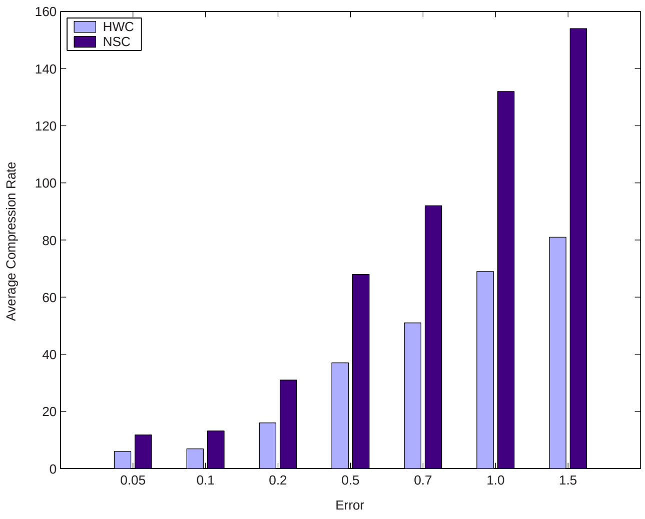


Fig. 3. The average compression rate of the whole database for each error using HWC and NSC. The error is the average joint distance error in cm.

joint j. The vector c is selected to minimize the distortion error presented earlier in Equation 4. More details can be found in the original paper.

We use the same motion examples as Beaudoin *et al* [1] to compare our method (NSC) with BWC and HWC (Table 1). All compressed results are further coded using *gzip*, which yields an additional 1.1:1 compression ratio. Table 1 shows that on average, over the various error rates chosen, NSC achieves a 50% higher compression rate than BWC and 61% higher compression rate than HWC for the running motion, and a 65% higher compression rate than BWC and a 79% higher compression rate than HWC for the jumping motion.

To broaden our evaluation, we have also compressed the whole database. The comparison results are shown in Figure3. NSC is slower than BWC and HWC in compression time due to the iterative nature of anisotropic diffusion. However, compression time is not as important as decompression time. The average compression time per frame of NSC is about 1 ms. However, the average decompression time from NSC is about $95\mu s$ per frame, which is much faster than real time. At 120Hz, we can decompress 87 motions simultaneously.

As mentioned in [1], visible artifacts will appear when $\epsilon_x > 0.5$. The most visible artifact is footskate. There exist sophisticated methods to solve the footskate problem [24,2,1]. We use the method proposed in [1], which compresses the foot joints separately and uses inverse kinematics to remove the footskate.

6 Conclusion

This paper presents a lossy compression method based on nonuniform segmentation. Motion curves at each DOF are segmented at feature discontinuities. A cubic Bézier curve is then used to approximate each segment. The cubic control points are further coded using a hierarchical coding method. We compare our method with the latest related work and an optimized Haar wavelet method. Our experiments show that this method can achieve about 65% higher compression rates than previous work with the same distortion error, and that the compression improvement increases for lower error tolerances. This method is easy to implement and has a fast decompression speed which makes it suitable for game and animation engines.

One disadvantage of this method is the comparatively slower compression process. However, this problem is not critical for most real-world applications, and it may be possible to use better anisotropic diffusion implementations in the future, such as multigrid, to address this.

In this paper, we only exploit temporal coherence in the data. In the future work, we will try to exploit DOF redundancy to improve compression performance. However, the segments in different DOFs in this method do not align along the boundaries, so it will be necessary to apply dimensionality reduction before applying our technique to the resulting coefficients.

References

1. Beaudoin, P., Poulin, P., van de Panne, M.: Adapting wavelet compression to human motion capture clips. In: Proceedings of Graphics Interface 2007 (2007)
2. Arikan, O.: Compression of motion capture databases. In: Proceedings of ACM SIGGRAPH 2002, pp. 890–897. ACM Press, New York (2006)
3. Guskov, I., Khodakovsky, A.: Wavelet compression of parametrically coherent mesh sequences. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 183–192. ACM Press, New York (2004)
4. Liu, G., McMillan, L.: Segment-based human motion compression. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer animation, pp. 127–135. ACM Press, New York (2006)

5. Assa, J., Caspi, Y., Cohen-Or, D.: Action synopsis: Pose selection and illustration. In: Proceedings of ACM SIGGRAPH 2005, pp. 667–676. ACM Press, New York (2005)
6. Kondo, K., Matsuda, K.: Keyframes extraction method for motion capture data. Journal for Geometry and Graphics 8, 81–90 (2004)
7. Liu, Z., Gortler, S., Cohen, M.: Hierarchical spacetime control. In: Proceedings of ACM SIGGRAPH 1994, pp. 35–42. ACM Press, New York (1994)
8. Lee, J., Shin, S.: Multiresolution motion analysis with applications. In: International Workshop on Human Modeling and Animation, pp. 131–143 (2000)
9. Lee, J., Shin, S.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of ACM SIGGRAPH 1999, pp. 39–48. ACM Press, New York (1999)
10. Ibarria, L., Rossignac, J.: Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 126–135. ACM Press, New York (2003)
11. Safonova, A., Hodgins, J., Pollard, N.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In: Proceedings of ACM SIGGRAPH 2004, pp. 514–521. ACM Press, New York (2004)
12. Forbes, K., Fiume, E.: An efficient search algorithm for motion data using weighted PCA. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 67–76. ACM Press, New York (2005)
13. Barbič, J., Safonova, A., Pan, J., Faloutsos, C., Hodgins, J., Pollard, N.: Segmenting motion capture data into distinct behaviors. In: Proceedings of Graphics Interface 2004, pp. 185–194 (2004)
14. Grochow, K., Martin, S., Hertzmann, A., Popović, Z.: Style-based inverse kinematics. In: Proceedings of ACM SIGGRAPH 2004, pp. 522–531. ACM Press, New York (2004)
15. Chai, J., Hodgins, J.: Performance animation from lowdimensional control signals. In: Proceedings of ACM SIGGRAPH 2005, pp. 686–696. ACM Press, New York (2005)
16. Glardon, P., Boulic, R., Thalmann, D.: A coherent locomotion engine extrapolating beyond experimental data. Computer Animation and Social Agents, 73–84 (2004)
17. Pullen, K., Bregler, C.: Motion capture assisted animation: Texturing and synthesis. In: Proceedings of ACM SIGGRAPH 2002, pp. 501–508. ACM Press, New York (2002)
18. Rose, C., Cohen, M., Bodenheimer, B.: Verbs and adverbs: Multidimensional motion interpolation. IEEE Computer Graphics and Applications 18, 32–41 (1998)
19. Lengyel, J.E.: Compression of time-dependent geometry. In: Proceedings of the Symposium on Interactive 3D Graphics, pp. 89–95 (1999)
20. Kondo, K., Matsuda, K.: Compression of dynamic 3D geometry data using iterative closest point algorithm. Computer Vision and Image Understanding 87, 116–130 (1987)
21. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, 629–639 (1990)
22. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8, 679–698 (1986)
23. Graphics Lab, Carnegie-Mellon University (Carnegie-Mellon Motion Capture Database), `http://mocap.cs.cmu.edu`
24. Kovar, L., Gleicher, M., Schreiner, J.: Footstake cleanup for motion capture editing. In: Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation, pp. 97–104. ACM Press, New York (2002)

Image-Space Collision Detection Through Alternate Surface Peeling

Han-Young Jang, TaekSang Jeong, and JungHyun Han

Game Research Center, College of Information and Communications, Korea University, Seoul, Korea

Abstract. This paper presents a new image-space algorithm for real-time collision detection, where the GPU computes the potentially colliding sets, and the CPU performs the standard triangle/triangle intersection test. The major strengths of the proposed algorithm can be listed as follows: it can handle dynamic models including deforming and fracturing objects, it can take both closed and open objects, it does not require any preprocessing but is quite efficient, and its accuracy is proportional to the visual sensitivity or can be controlled on demand. The proposed algorithm would fit well to real-time applications such as 3D games.

1 Introduction

Collision detection is a fundamental problem in many applications such as 3D games, virtual reality, medical simulation, physically-based simulation, and robotics. A lot of algorithms for collision detection have been proposed. The algorithms based on triangulated models can be classified into two broad categories. One is *object-space approach* and the other is *image-space approach*.

In the object-space approach, most of the proposed algorithms are accelerated by utilizing spatial data structures which are often hierarchically organized and are based on bounding volumes [1,2]. A state-of-the-art algorithm in the object-space approach is found in the work of Zhang and Kim [3]. The algorithm performs AABB overlap test which is accelerated by GPU. The algorithm runs quite fast enough to handle deformable objects with high accuracy. However, it is not suitable for fracturing objects. When the triangles in an AABB fall apart due to fracture, the AABB stream often has to be restructured. Such restructuring hampers real-time performance.

The image-space approach typically measures the volumetric ranges of objects along the viewing direction, and then compares the ranges to detect collision. Since the seminal work of Shinya and Forgue [4], various algorithms for image-space approach have been proposed, attempting to maximally utilize the powerful rasterization capability of the GPUs. Recent efforts in the image-space approach include the work of Heidelberger *et al.* [5] and Govindaraju *et al.* [6]. The approach of Heidelberger *et al.* can handle concave objects, but requires a considerable amount of time for collision detection of objects with complex geometry, due to the rendering and readback overhead. As an effort to alleviate the readback problem, Govindaraju *et al.* proposes CULLIDE algorithm, which can reduce the readback overhead using occlusion query. However CULLIDE algorithm requires an off-line setup stage, which defines the sub-objects used for each occlusion query. In case of a fracturing model, the topology may vary frame by frame. The time-consuming pre-processing has to be executed per each frame.

This paper proposes a new algorithm for the image-space approach. Its key components are implemented in shader programs. Like many of the image-space collision detection algorithms, the proposed algorithm can handle deformable models. The unique strength of the proposed algorithm is its versatility: it can handle both closed and open objects, and more importantly it can take as input various dynamic models including fracturing meshes. Moreover, our algorithm overcomes not only the readback and rendering overhead but also the accuracy problem of the ordinary image-space approach. The proposed algorithm does not require any pre-processing, is simple to implement, and shows superior performance. Such an algorithm is attractive for real-time applications such as 3D games.

2 Overview of the Approach

This paper proposes to compute potentially colliding sets(PCSs) using GPU and leave the primitive-level intersection test to CPU. This framework is similar to the state-of-the-art work in the image-space approach, CULLIDE [6], and that in the object-space approach, the work of Zhang and Kim [3]. Unlike CULLIDE, however, the proposed algorithm computes the PCSs always at the primitive level, and further it resolves the major drawbacks of CULLIDE discussed in the previous section. Unlike the work of Zhang and Kim [3], the proposed algorithm maintains the trade-off between accuracy and efficiency, and can handle fracturing objects.

Fig. 1 shows the flow chart of the proposed image-space collision detection algorithm. Each object in the scene is associated with an axis-aligned bounding box (AABB). If the AABBs of two objects O_1 and O_2 intersect, the intersection is passed to GPU as a *region of interest* (*ROI*). Fig. 2 illustrates three examples, each with a pair of objects, their AABBs, and the ROI. No ROI is found in Fig. 2-(a) whereas ROIs are found and passed to GPU in Fig. 2-(b) and -(c). Given an ROI, GPU computes PCSs. A PCS consists of two triangle sets, one from O_1 and the other from O_2. No PCS is computed in Fig. 2-(b) whereas two PCSs

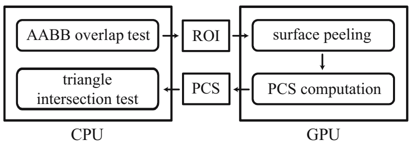


Fig. 1. System architecture

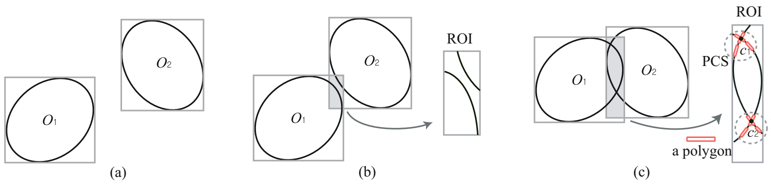


Fig. 2. Object AABBs and ROIs. (a) no ROI, (b) ROI with no collision, (c) ROI with collision.

are computed in Fig. 2-(c). Given the PCSs in Fig. 2-(c), CPU performs the traditional triangle intersection test to obtain the intersection points c_1 and c_2.

3 Surface Peeling and PCS Computation

Given an ROI passed by CPU, GPU computes PCSs by rendering the ROI surfaces into textures. Rendering is done in a layer-by-layer fashion, and we call it *surface peeling*. (The idea of surface peeling is not new, and its root comes from the area of transparent surface rendering [7]. It has been named *depth peeling*. There have been lots of applications of depth peeling, and in fact the work of Heidelberger *et al.* [5] described in Section 1 adopted the depth peeling algorithm. In this paper, we use the terminology 'surface peeling' instead of 'depth peeling' for stressing the differences between the two approaches, which will be discussed later.)

As in many image-space collision detection algorithms, orthographic projection along a viewing direction is used for rendering. It can be described as casting of parallel rays. Fig. 3 illustrates the surface peeling process with the example of Fig. 2-(c). Between the two objects, the one farther from the viewpoint is first rendered. It is O_2 in the example. The rendering result is stored in a 32-bit floating-point texture, `texture #1`, as illustrated in Fig. 3-(a). In the texture, the R channel stores the *depth* value of the rendered pixel, and the G channel stores the *triangle ID* of the pixel, which is the ID of O_2's triangle hit by the ray. (Triangle identification will be discussed in Section 5).

In the next phase, a new shader program renders O_1 through the *depth test* with the output of the previous stage (currently, stored in `texture #1`). The shader program discards a pixel of O_1 if it is not deeper than the corresponding texel in `texture #1`. The result is stored in `texture #2`, as shown in Fig. 3-(b), where the depth values and triangle IDs are stored into color channels R and G.

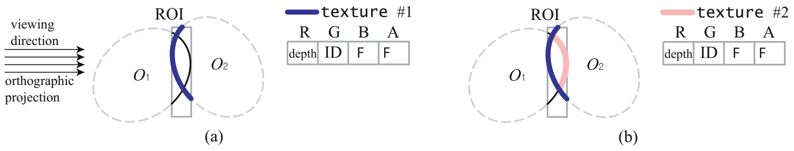


Fig. 3. Surface peeling. (a) phase 1, (b) phase 2.

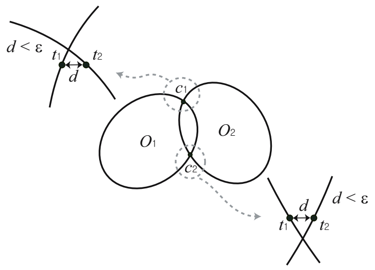


Fig. 4. PCS computation

In order to compute PCSs, a simple test is invoked for the space between `texture #1` and `texture #2` in Fig. 3. If the distance d between texel t_1 from `texture #1` and texel t_2 from `texture #2` is less than the threshold ϵ, as shown in Fig. 4, the triangle IDs are retrieved from t_1 and t_2, and then passed to CPU as PCSs. Finally, given the PCSs, CPU computes the intersection points, c_1 and c_2 in Fig. 4.

4 Alternate Surface Peeling

The ROI is composed of a pair of two objects, and in general PCS computation requires the two objects to be *alternately rendered*. Let us discuss the alternate rendering using the example in Fig. 5. O_2 is deeper than O_1, and therefore O_2 is rendered first to create `texture #1`, as shown in Fig. 5-(a). O_1 is then rendered to create `texture #2`, as shown in Fig. 5-(b). Surface peeling does not stop here. The shader program again renders O_2 through the depth test with `texture #2`, i.e. the shader program discards a pixel of O_2 if it is not deeper than the corresponding texel of `texture #2`. The result is stored in `texture #3`, shown in Fig 5-(c).

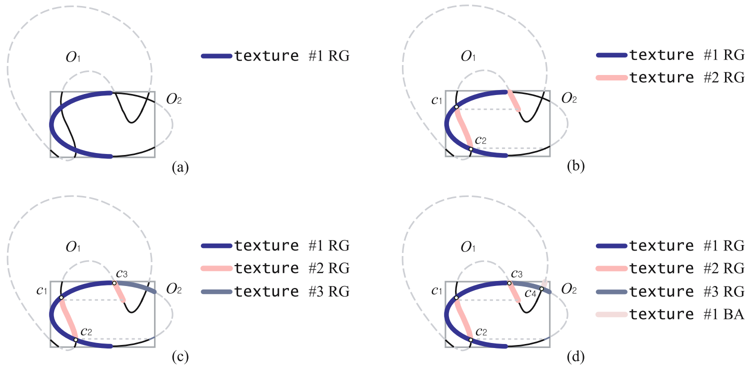


Fig. 5. Alternate surface peeling with three textures. (a) phase 1, (b) phase 2, (c) phase 3, (d) phase 4.

Suppose that the current implementation uses a graphics chip with four render target textures. (The state-of-the-art graphics hardware such as GeForce 8800 series supports eight render targets.) Among the four textures, three are used for surface peeling, and the remaining one is for recording PCSs. For efficient recording and readback of the PCSs, a hierarchical technique proposed by [8] is employed.

Note that only color channels R and G have been used so far to store the depth values and triangle IDs. Color channels B and A are empty. It is time to render O_1 through the depth test with `texture #3`. The rendering result is recorded at color channels B and A of `texture #1`, as shown in Fig. 5-(d). No more surface remains in the ROI, and the alternate rendering stops. (Occlusion query is used to decide when to stop rendering.) If we had more surfaces to render, B and A channels of `texture #2` and then `texture #3` would be used.

For a complex configuration of objects, however, more than six times of surface peeling may be needed. Then, the texture storing the PCSs is read back to CPU, and the second stage of the surface peeling is started, where the 6th surface peeled in the first stage is used for depth test.

Note that, as shown in Fig. 5-(d), not all surfaces in the ROI are rendered. In contrast, the Heidelberger′s algorithm based on depth peeling [5] renders *all* surfaces in the ROI, and all of the rendered surfaces are read back to CPU, which causes a serious overhead. Recall that our algorithm computes PCSs from the rendered surfaces, and passes the PCSs to CPU, not the rendered surfaces themselves. Given objects with complex geometry, the alternate surface peeling algorithm proposed in this paper is superior in performance to the Heidelberger′s algorithm.

PCSs are created between two adjacent textures. For example, in Fig. 5-(b), two PCSs are obtained between `texture #1` and `texture #2`, and eventually lead to the intersection points c_1 and c_2. Similarly, intersection point c_3 in Fig. 5-(c) is computed by CPU from the PCS obtained between `texture #2` and `texture #3`. Finally, c_4 in Fig. 5-(d) is computed from the PCS between `texture #3` and `texture #1`.

5 Triangle Identification for PCS Computation

A well-known problem of image-space collision detection is that its effectiveness is limited by the image-space resolution and the image-space approach often misses overlapping primitives. See Fig. 6 where two objects, O_1 and O_2, collide. O_1 includes the triangles (polygons) p_1, p_2 and p_3, and O_2 includes p_4 and p_5. Along the upper ray, t_1 is first recorded into the texture, but no pairing is possible because the intersection between the ray and p_1 lies in front of t_1. Therefore, no PCS is obtained. In contrast, the mid ray detects $<t_2,t_4>$ and identifies the PCS, $\{(p_2),(p_5)\}$. In actuality, p_2 and p_5 do not intersect, and CPU computes no intersection point. The lower ray detects $<t_3,t_5>$ and identifies the PCS, $\{(p_3),(p_5)\}$. Then, the CPU will compute the intersection point c_2. Note that the other intersection point c_1 is not found. Unless the image-space resolution reaches the infinity, overlapping primitives can be missed in the proposed algorithm.

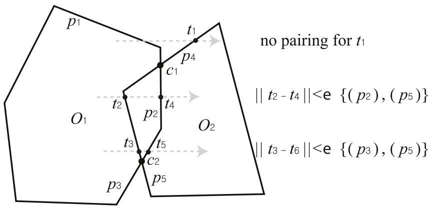


Fig. 6. Sampling error alleviation

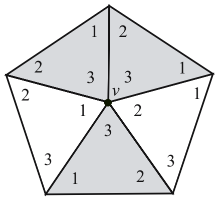


Fig. 7. Triangle ID: The center vertex v is assigned a gray color. In each triangle, vertex ordering is denoted by numbers. All of the three gray-colored triangles have v as the last vertex.

In order to alleviate (not solve) the inaccuracy problem, the proposed algorithm adopts a trick based on *flat shading*, where a triangle's color is set to the color of its last vertex. (It is OpenGL convention. In DirectX, the color of the first vertex determines the triangle color.) In the proposed scheme, a distinct color c is associated with each vertex v. Therefore, every flat-shaded triangle whose last vertex is v is colored in c, as illustrated in Fig. 7. For each pixel stored in the texture during the surface peeling process, its color is taken as the triangle ID. Note that the triangle ID is not unique, i.e. multiple triangles may have an ID. For example, the three gray-colored triangles in Fig. 7 will be given an identical ID.

Such non-unique IDs alleviate the problem caused by an insufficient image-space resolution. Let us revisit the mid ray in Fig. 6, and suppose p_4 and p_5 of O_2 have an identical ID. Then, the PCS will be $\{(p_2),(p_4,p_5)\}$, not just $\{(p_2),(p_5)\}$, and the intersection point c_1 can be obtained.

Fig. 8 illustrates the ratio, which is named *miss ratio*, of the missing to all intersection points. The miss ratios depend on viewing directions. In Fig. 8, a pair of θ (longitude) and ϕ (latitude) defined in a spherical coordinate system represents a viewing direction, where θ is sampled at intervals of 30^o and ϕ is at 10^o. Given the configuration of bunny and dragon in Fig. 8-(a), Fig. 8-(b) shows the miss ratios when the ROI is assigned a 256×256-sized texture. The algorithm adopts the non-unique ID scheme. The average miss ratio is 5.02% while the largest miss ratio is 12.7% and the smallest is 0.65%.

Fig. 8-(c) shows the miss ratios when the algorithm adopts the unique ID scheme, i.e. a triangle is given a unique ID. The miss ratios become high. The

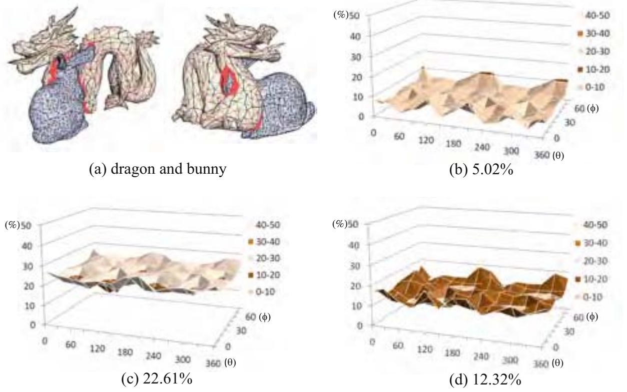


Fig. 8. Miss ratios: dragon (2.9K triangles) and bunny (2.7K triangles)

average miss ratio is 22.61% while the largest miss ratio is 29.31% and the smallest is 14.98%.

Fig. 8-(d) shows the result of using a 128×128-sized texture, where non-unique ID scheme is used. The average miss ratio is 12.32% while the largest miss ratio is 20.52% and the smallest is 5.86%. Compare the miss ratios of Fig. 8-(b) and Fig. 8-(d): 5.02% and 12.32%. Obviously, the miss ratio increases when a smaller texture is assigned to an ROI.

Recall that a smaller texture is assigned to an ROI when there are many ROIs in the scene. In such a scene of crowded colliding objects, each ROI generally takes just a small fraction in the screen. Therefore, inaccuracy in collision detection and consequent collision response would not be easily perceived. In contrast, suppose that the scene consists of the two objects in Fig. 8-(a). Then, the entire 512×512-sized texture is assigned to the ROI, and the miss ratio drops significantly, leading to realistic collision response. In summary, the precision of the collision detection is roughly proportional to the user's visual sensitivity. This feature makes the proposed algorithm distinguished from other image-space algorithms.

6 Implementation and Test

The proposed algorithm has been implemented in C++, OpenGL and Cg on a PC with 2.4 GHz Intel Core2 Duo CPU, 2GB memory, and NVIDIA GeForce 7900GTX GPU with 512 video memory and PCI-Express interface. Various functionalities of the graphics hardware are exploited, e.g. the GL_NV_occlusion_query for surface peeling, EXT_framebuffer_object for off-screen rendering, etc.

As discussed earlier, the proposed algorithm can handle a variety of dynamic objects. Fig. 9 shows a scene where a deforming object, lizard, walks through the balls on the billiard table. (See the attached video.) Table 1 shows the performance statistics for 4 configurations in Fig. 9. Each configuration has a distinct

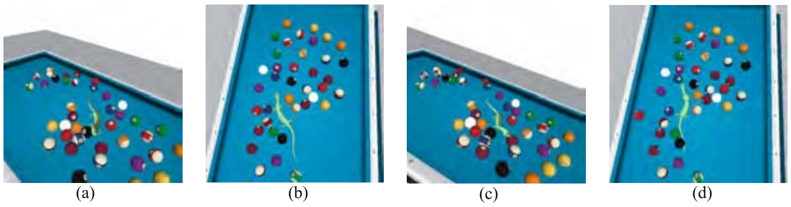


Fig. 9. Test #1: deforming lizard (2.1K triangles) and 40 billiard balls (each of 4.5K triangles)

Table 1. Performance evaluation for lizard and billiard table simulation (times in ms)

	nCTP	AABB	GPU	readback	CPU	total
(a)	15	0.07	3.11	0.11	0.07	3.36
(b)	33	0.07	3.08	0.13	0.14	3.42
(c)	96	0.07	3.28	0.15	0.44	3.94
(d)	173	0.07	3.02	0.36	0.68	4.13

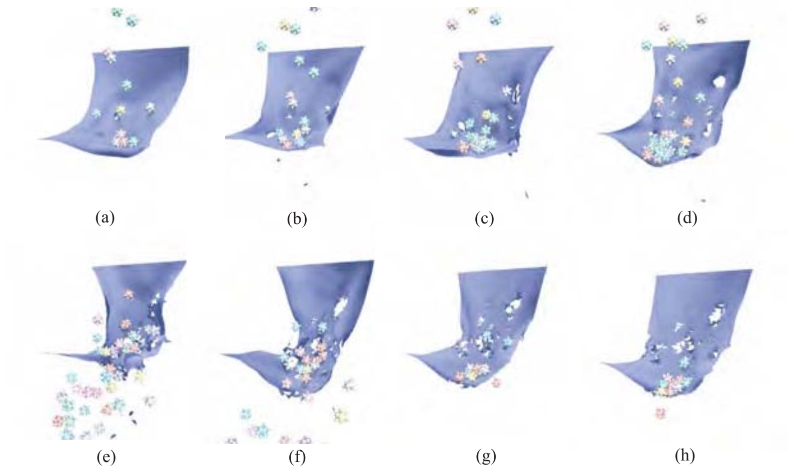


Fig. 10. Test #2: Fracturing cloth (1.6K triangles) and 40 falling objects (each of 2.1K triangles)

number of colliding triangle pairs (nCTP). AABB construction requires negligible amount of time. GPU time is spent for surface peeling and PCS computation. Note that the GPU times remain almost constant for varying nCTPs. In contrast, the readback time is proportional to the PCS size. So is the CPU time spent for triangle/triangle intersection test.

Fig. 10 shows simulation of a deforming and fracturing object, cloth, which is being torn by the falling sharp rigid objects. (See the attached video.) When the cloth is torn, its mesh connectivity changes. Furthermore, new triangles are

Table 2. Performance evaluation for cloth simulation (time in ms)

	nCTP	AABB	GPU	readback	CPU	total
(a)	16	0.03	2.66	0.12	0.10	2.91
(b)	115	0.03	3.05	0.68	0.66	4.42
(c)	96	0.03	4.26	0.39	0.48	5.16
(d)	39	0.03	2.39	0.14	0.36	2.93
(e)	125	0.03	3.22	0.73	0.67	4.65
(f)	222	0.03	2.38	0.84	0.95	4.20
(g)	339	0.03	3.15	0.92	1.66	5.75
(h)	262	0.03	2.90	0.71	1.17	4.81

dynamically added in the areas being split, to achieve more natural simulation. The collision detection algorithm proposed in this paper does not require any extra time for handling such a dynamic and fracturing object. In contrast, handling this kind of simulation would be difficult in the state-of-the-art collision detection algorithms such as those of Govindaraju *et al.* [6] (CULLIDE) and Zhang and Kim [3]: CULLIDE requires each sub-object to be a single triangle, which would lead to a huge number of occlusion queries, and the algorithm by Zhang and Kim requires an AABB to contain a single triangle, which would lead to a serious readback overhead.

Table 2 shows the performance statistics for 8 configurations in Fig. 10. Cloth simulation itself takes about 2 ms for all configurations. Collision detection among the falling rigid objects is done using RAPID 2.0 [9] which is better for rigid body collision detection.

7 Conclusion

This paper presented an efficient image-space algorithm for real-time collision detection. In the current implementation, shader programs compute the PCSs, and CPU performs the primitive-level intersection test. The algorithm can handle a variety of dynamic objects including fracturing meshes, and does rarely suffer from the readback problem. The experimental results show the feasibility of the shader-based collision detection and its performance gain in real-time applications such as 3D games.

The proposed algorithm also has disadvantages. It works in a synchronous mode between CPU and GPU, i.e. CPU waits until GPU computes the PCSs. The proposed algorithm cannot handle self-collision. The proposed algorithms are being extended for overcoming these disadvantages.

Acknowledgement

This work was supported by grant No. R01-2006-000-11297-0(2007) from the Basic Research Program of the Korea Science & Engineering Foundation, and MIC, Korea under ITRC IITA-2006-(C1090-0603-0046).

References

1. van den Bergen, G.: Efficient collision detection of complex deformable models using aabb trees. Journal of Graphics Tools 2, 1–13 (1997)
2. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree: A hierarchical structure for rapid interference detection. In: Proc. of ACM SIGGRAPH 1996, pp. 171–180. ACM Press, New York (1996)
3. Zhang, X., Kim, Y.: Interactive collision detection for deformable models using streaming AABBs. IEEE Transaction on Visualization and Computer Graphics 13, 318–329 (2007)
4. Shinya, M., Forgue, M.: Interference detection through rasterization. Journal of Visualization and Computer Animation 2, 131–134 (1991)
5. Heidelberger, B., Teschner, M., Gross, M.: Detection of collisions and self-collisions using image-space techniques. In: Proc. of Winter School of Computer Graphics 2004, pp. 145–152 (2004)
6. Govindaraju, N.K., Redon, S., Lin, M.C., Manocha, D.: CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. In: Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware 2003, pp. 25–32. ACM Press, New York (2003)
7. Everitt, C.: Interactive order-independent transparency. NVIDIA corporation technical report (2001), `http://www.cs.unc.edu/~{}geom/obb/obbt.html`
8. Choi, Y., Kim, Y., Kim, M.: Self-CD: Interactive self-collision detection for deformable body simulation using GPUs. In: Proc. of Asian Simulation Conference, pp. 187–196 (2005)
9. Gottschalk, S.: RAPID: Robust and accurate polygon interference detection system (1996), `http://www.cs.unc.edu/~{}geom/obb/obbt.html`

Robust Classification of Strokes with SVM and Grouping

Gabriele Nataneli and Petros Faloutsos

University of California Los Angeles
nataneli@cs.ucla.edu,
pfal@cs.ucla.edu

Abstract. The ability to recognize the strokes drawn by the user, is central to most sketch-based interfaces. However, very few solutions that rely on recognition are robust enough to make sketching a definitive alternative to traditional WIMP user interfaces. In this paper, we propose an approach based on classification that given an unconstrained sketch, can robustly assign a label to each stroke that comprises the sketch. A key contribution of our approach is a technique for grouping strokes that eliminates outliers and enhances the robustness of the classification. We also propose a set of features that capture important attributes of the shape and mutual relationship of strokes. These features are statistically well-behaved and enable robust classification with Support Vector Machines (SVM). We conclude by presenting a concrete implementation of these techniques in an interface for driving facial expressions.

1 Introduction

In recent years there has been a proliferation of sketch-based solutions designed for a variety of different applications. Some approaches rely on a procedural mapping between two-dimensional strokes and possibly higher-dimensional geometries [1] or models [2]. The problem is generally well-defined and well-constrained in these cases. As a result, these solutions tend to be very usable and effective, although they may be limited to a specific domain. Other approaches rely on the detection of a well-defined dictionary of symbols. In this context, there is a large literature on interfaces based on handwriting recognition and the recognition of other symbols that belong to the vernacular of a particular application. Alternatively, some solutions define a new set of symbols that is easy to learn and can be employed to simplify certain interactive tasks. Motion Doodles [3] is an example of the latter, which enables users to define character animation through a fluid set of cursive motions.

The most challenging task is that of recognition-based sketching. A general approach for recognition-based solutions , such as [4], is to analyze the sketch by identifying and recognizing components that capture the semantics of the sketch. We call these components simply as strokes.

In this paper, we explore the problem of recognizing strokes robustly with particular emphasis on the problem of classification. We use a support vector machine that is trained on quantifiable aspects of the sketch to perform the classification. We then use stroke grouping to augment the capability of the classifier to work with complex sketches in the presence of outliers.

The contributions of this work can be summarized as follows:

– We propose a machine learning approach to classify the elements of unconstrained input sketches based on support vector machines
– We propose a set of attributes that capture both individual shape features and spatial relationships of groups of strokes.
– We propose a grouping technique that makes our approach robust to outliers.

2 Related Work

There is a large body of work concerned with the related problem of handwriting recognition. [5] presents an approach for recognizing and grouping handwritten text. This work groups strokes by minimizing a cost function efficiently using a dynamic programming algorithm. This paper is similar in spirit to our approach, but is tailored around the problem of symbol recognition, while our approach strives to be more general. [6] is related to [5] and uses stroke groupings to aid in the recognition of numeral strings. Their work selects the best grouping based on a *grouping-hypothesis* and their focus is restricted to the recognition of numeral strings. This paper also puts particular emphasis on segmentation which is absent both in [5] and our paper. The work by Saunds on perceptual organizations [7,8] is one of the main sources of inspiration for our approach based on groupings. These papers show how to organize strokes into groups that are perceptually sound. Sharon in the paper [4] presents an approach for sketch-recognition based on a computer vision technique referred to as *constellation model*, which is related to our work. The classic work by Arnheim [9] presents an enlightening study of visual perception in the context of art and psychology. This work is the basis and main inspiration for the shape attributes that we use to train the SVM. We briefly mention a few additional papers that present interesting ideas that are related to our work. The paper [2] describes a similar application for posing 3D faces via sketches, but it does not rely on sketch recognition. The work by Yang [10] presents an approach for the analysis of sketches based on their connectivity. Some of the ideas introduced by this paper, such as the use of templates and the parametrization, resemble some aspects of our application described in section 9.

3 Overview

As the user draws a sketch interactively, we record every continuous motion of the mouse or stylus as a separate stroke. In order to recognize the sketch, a classifier assigns a label to each stroke that comprises the sketch based on the content of a user-generated training set. The training set is based on a compact and statistically well-behaved set of shape descriptors, called shape attributes, that are discussed in section 7. We augment the capabilities and robustness of the classifier by grouping strokes [7,8]. What is novel in our approach is that our mechanism for selecting the groupings is largely data driven. As a result we do not simply rely on spacial groupings [5] or groupings based on the geometry of strokes [5,8], but we also enable groupings based on the semantics of the sketch that can be inferred from the training set. In the following sections, we describe our classification mechanism and then show how it can be associated with groupings to robustly handle complex sketches containing outliers. Our discussion is followed by a detailed description of shape attributes, which lie at the heart of the classification. A detailed understanding of shape attributes is not required to understand the following sections, but the interested reader may want to skip ahead to section 7 before reading sections 4 and 5.

4 Classification

Our approach for classification relies on a support vector machine (SVM). The robustness of SVM depends primarily on the amount and choice of training data as well as the choice of a kernel. If the training data makes the structure of the data obvious, we can expect machine learning to perform robustly even for a fairly small training set. Otherwise, even a large training set may perform poorly. For the specific problem of classification the quality of a training set can be evaluated quantitatively by studying the variance of the data. Items that belong to the same class should exhibit relatively low variance, while items that belong to different classes should be well-separated and therefore exhibit large variance. A classifier trained on a training set that possesses these properties is expected to perform robustly. We represent shape through a set of features that we call *shape attributes*. In this paper, we use the term shape attribute interchangeably to express both the features of strokes and the functions used to quantify these features.

We use the following formalism to clarify our approach. When the user draws a stroke, we store it in memory as a vector of points s_i. When the drawing is complete the entire sketch is merely a set of strokes $S = \{s_1, s_2, \ldots, s_n\}$. A shape attribute is a function $A(s_i)$ that analyzes the stroke s_i and produces a real number $a \in \Re$. During training the user also needs to manually associate a label l_i to each stroke s_i in the sketch. We construct a training vector t_i by combing the label l_i with a selection of shape attributes. Hence, a training vector t_i has the form

$$t_i = [l_i, A_1(s_i), A_2(s_i), \ldots, A_m(s_i)]$$

A training example T for the SVM corresponding to the sketch S is therefore represented by the set of training vectors $T = \{t_1, t_2, \ldots, t_n\}$. During performance, the user draws a new sketch $S' = \{s'_1, s'_2, \ldots, s'_k\}$. For each stroke s'_i we construct a query vector

$$q_i = [A_1(s'_i), A_2(s'_i), \ldots, A_m(s'_i)]$$

The query vector q_i is then used to query the classifier C and obtain a class label l'_i for each corresponding input stroke

$$l'_i = C(q_i)$$

We initially developed 12 shape attributes that capture the notions described in [9]. However, for the classifier we restricted the number of shape attributes to a stable subset that results in robust classifications. To find this stable subset, we generated many alternative training sets using different selections of shape attributes and we studied their variance. We chose a selection of shape attributes that exhibits small variance for strokes that belong to the same class and large variance for strokes that belong to distinct classes. We repeated this process for various categories of sketches corresponding to faces, houses, and cars to make sure that our selection is independent of the content of the sketch. Table 1 shows the stable selection of shape attributes that we chose for the classification. Surprisingly all these shape attributes correspond to high level features of strokes. This finding agrees with our intuition. In fact, a few generic strokes are generally sufficient for a human to recognize the content of a sketch and a robust classifier is expected to do the same.

Bounding Box Width
Bounding Box Height
Bounding Box Aspect Ratio
Centroid X
Centroid Y
Horizontal Ordering
Vertical Ordering
Overall Stroke Count
Depth

Fig. 1. The stable selection of shapes attributes used for the classification

We verified the robustness of the classification based on this selection of shape attributes by training the SVM and running cross-correlation tests. Subsequently, we tested the classification manually by drawing many sample sketches and observing classification results. The average accuracy of our classifier with different training sets is about 93%. Our sample sketches did not contain either outliers or duplicate strokes - that is multiple strokes of the same class. We refer to sketches that satisfy these two properties as *clean* sketches.

Our experiments show that in general a fairly small training set that contains approximately ten training vectors per class is sufficient to obtain a robust classification of clean sketches using our method. The actual size of the training set that is needed to achieve a robust classification is correlated to the variance of training vectors. If the training vectors of two distinct classes exhibit a small variance, we need more examples for those two classes.

A SVM is sensitive to the scaling of data, so we always have to normalize shape attributes before using them in the training. However, we observed a 20% improvement in the classification by exploiting a non-uniform scaling of the data. Specifically, the ordering attributes, the overall stroke count, and the depth attribute are left unchanged as integer values. This improvement follows from the fact that the different scaling makes the SVM weigh these shape attributes differently. This choice also impacts how the choice of kernel for the SVM affects the performance of the classifier.

Support Vector Machines rely on a choice of kernel for their operation. The only two kernels that provide acceptable performance for our task are linear and polynomial. A linear kernel works best when we want to classify sketches containing a single stroke or sketches in which the relationship of strokes is not relevant. In these cases, the SVM prioritizes the shape attributes that are properly scaled, which for us are the ones that relate to individual features of a stroke. On the other hand, A polynomial kernel is less influenced by the non-linearity of the training set, so it is the best choice in all common sketch recognition tasks. This property of our classifier offers an advantage over solutions that rely solely on the relation of strokes [4].

5 Grouping

In the previous section we highlighted the fact that our core classification mechanism is only expected to work robustly for clean sketches. Again, a clean sketch is one that satisfies the following two properties:

1. It does not contain outliers.
2. No two strokes belong to the same class (duplicate strokes).

We know that a clean sketch is classified successfully only if each query vector is assigned a distinct label. Therefore, if we find a duplicate label in the classification results, we are certain that a misclassification has occurred. This is a central property for searching the space of possible groupings as explained later. Furthermore, we also expect more reliable classifications for clean sketches. Clean sketches do not contain outliers, which are one of the major sources of errors in classification. An outlier is a stroke whose correct label is not present in the training set. Hence, outliers always lead to misclassifications, since the classifier is always expected to output a label for any given query vector. The reason why the second property of clean sketches is important is more subtle. Duplicate strokes are strokes whose corresponding query vectors exhibit low variance and thus are assigned the same label by the SVM during performance. Therefore, duplicate strokes are intrinsically ambiguous and they are generally not well-behaved statistically. Moreover, if we allowed duplicate strokes, we would not have any automated way for distinguishing a duplicate stroke from a misclassified one. In summary, clean sketches give us a better way to attest the robustness of the classifier.

Sketches of practical interest, however, do not satisfy the properties of clean sketches in most cases. Therefore we use groupings to accommodate more complex sketches within the existing framework for classification. Given a Sketch $S = \{s_1, s_2, \ldots, s_n\}$ a grouping $G = \{g_1, g_2, \ldots, g_n\}$ is a set of groups g_i where

1. $g_i \subset S$
2. $g_i \cap g_j = \emptyset$ with $i \neq j$

Our objective is to find a proper grouping G of sketch S that is guaranteed to be clean. Any sketch that is not clean can be reduced to a clean sketch by some grouping as follows:

– Group duplicate strokes with the same label into the same group
– Group every outlier with some other stroke

Groupings, however, present an additional challenge as well. After we group several strokes, we need to define the meaning of applying a shape attribute to the group - that is we have to define the meaning of $A(g_i)$. We do this by replacing the group g_i with a new stroke that characterizes the basic features of the group. This is explained more in detail later in this section.

The space of possible groupings is a power set over the number of strokes given as input and it is clearly intractably large to be searched exhaustively. In order to make the problem tractable, we need to prune the space. We consider three different kinds of groupings to accomplish this task: structural grouping, overlap grouping, and semantic grouping. Each kind of grouping corresponds to a specific level of abstraction of the properties of the sketch.

Structural Grouping. At the lower level we have structural groupings. A structural grouping is based on purely geometric notions of *proximity* and *continuity. Proximity* refers to strokes that are aligned (vertically or horizontally) and are near each other. In terms of shape attributes, a proximity grouping selects strokes that have the same ordering (refer to Section 7) and whose distance between the centroids is below a threshold. Figure 2a is an example of proximity group. We characterize the proximity group with a new stroke that connects the centroids of each stroke that participates in the group.

Continuity refers to strokes whose endpoints are near each other. Unlike [7] we are not considering the orientation of the strokes in order to establish continuity, since the purpose of groupings in our case is slightly different. Figure 2b is an example of continuity group.

A structural grouping corresponds to a perceptual organization as described in [7] and [8]. A structural grouping is also the simplest kind of grouping and does not depend on the training set. The strokes that are candidates for a structural grouping are generally very common and very numerous in a typical sketch, so we do not want to perform expensive calls to the SVM in order to establish these groups. At the same time, even though structural groupings are not tailored to the training set -that is the particular domain of the recognition task - this fact does not affect the ability of higher level groupings, such as overlap groupings and semantic groupings, to analyze the semantics of the sketch successfully. Lastly, structural groupings are very effective at pruning the space of possible groupings and removing common outliers.

Overlap Grouping. At an intermediate level of abstraction there are groupings based on overlap. Stroke s_i overlaps stroke s_j if $A_{depth}(s_i) > A_{depth}(s_j)$ and $\|centroid(s_i) - centroid(s_j)\| < \alpha$, where A_{depth} is the depth shape attribute, *centroid* is a function that returns the position of the centroid for the given stroke, and α is a threshold. In this example, strokes s_i and s_j form an *overlap group candidate* (OGC) $g_{oc} = \{s_i, s_j\}$. We determine all OGCs efficiently by building an adjacency matrix of all strokes, while computing the shape attributes. The depth of an OGC g_{oc} is

$$depth(g_{oc}) = max(|A_{depth}(s_i) - A_{depth}(s_j)|)$$

where $s_i, s_j \in g_{oc}$. Figure 2c is an example of overlap group. The body of the house, the window, and the window grille form an OGC. The body has depth 0, the window has depth 1, and the window grille has depth 2. Hence, the depth of the OGC is 2.

We first reduce the number of strokes in each OGC by grouping the strokes in each OGC so that their depth falls below a given threshold β. The value of β depends on the amount of detail that we want to preserve. In Figure 2c if we set β to 2, we will discard the window grille. For each OGC we run an algorithm that can be summarized as follows:

1. Consider the power set of all OGCs. This will generate many subsets of OGCs of the form $\{g_{oc}^1, \ldots, g_{oc}^k\}$.
2. For each subset of OGCs group the strokes in each OGC. This will partition the original set of strokes S into a grouping of the form $G = \{g_1, \ldots, g_k, s_j, \ldots, s_n\}$ where each g is a group of strokes that is replaced with a stroke that represents the bounding box of the group and s_i (groups of cardinality 1) are strokes that are not grouped.
3. Classify the grouping G with SVM and obtain a set of labels $L = \{l_1, \ldots, l_n\}$. If L contains any duplicate, then the grouping G does not represent a clean sketch, so we reject this grouping. Otherwise, if L does not contain duplicates we accept the grouping G.
4. Rank all accepted groupings based on the heuristic described in section 6.
5. Repeat the steps for each subset generated in step 1.
6. Choose the grouping with the best rank.

Although the number of OGCs is generally fairly small, the number of subsets generated in step 1 may still be prohibitively large. However, performance is still acceptable since most subsets are either rejected or they get low rankings. In our experiments we only considered subsets of cardinality 2 or 3 without compromising the capability of our algorithm.

Semantic Grouping. In some cases, there are no strokes that overlap or there is not any subset of OGCs that results in a clean sketch. Semantic groupings consider groups that cannot be captured by either structural groupings or overlap groupings. Therefore, semantic groupings deal with groups of strokes that are generally not correlated by geometrical features, but are correlated by the semantics of the drawing that can be inferred from the training set.

We proceed similarly to overlap groups, but we replace step 1 with the power set over the strokes that are left ungrouped after performing structural grouping and overlap grouping. We refer to these as candidate strokes. Unfortunately there is no obvious way to prune the power set of candidate strokes, so we adopt a heuristic that was fairly effective in our experiments. We first run the classifier on all candidate strokes; the classification results are expected to contain duplicates at this stage. We then consider each group of duplicate as a candidate and run the same algorithm over the power set of these groups. The rationale is that duplicate strokes are likely to be related semantically.

Our approach for grouping gives us robustness against outliers. Outliers would always result in incorrect classifications if they are individually fed to the classifier. More precisely, outliers are very likely to generate classifications that contain duplicates by the pigeon hole principle. As a result, our approach for grouping forces outliers to be grouped with other strokes so that they will not reach the classifier in isolation. Moreover, our approach can cope with outliers gracefully, as we can improve the robustness by expanding the training set.

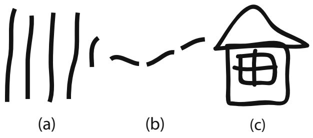


Fig. 2. Examples of different kinds of groups. (a) Proximity group. (b) Continuity Group. (c) Overlap group.

6 Ranking

Our approach for overlap and semantic groupings is effectively an optimization that searches for the grouping with a best rank. In turn, the ranking scheme is based on the results of the classifier. In general, we only want to rank classifications that do not contain duplicates - that is classifications that are known to correspond to clean sketches. We do this, because, we can only expect the classifier to perform robustly for clean sketches. We want to rank higher groupings that result in the largest number of distinct classification labels. This is because, we would like to exploit the variety of the training set as much as possible and we also want to avoid to group strokes that belong to distinct classes. Hence, we compute the rank by simply adding the number of distinct labels produced by the classification. In some cases, for semantic groupings we cannot simply reject groupings that contain duplicates; therefore, we actively penalize duplicates by subtracting the number of groups containing duplicates from the number of distinct labels.

7 Shape Attributes

In this section, we describe shape attributes in detail. Shape attributes provide an effective representation of strokes that acts as a useful abstraction for analyzing hand-drawn sketches. We originally developed 12 shape attributes for the application described in section 9. Here we only describe the subset of shape attributes that we use in the classification. For a complete list of shape attributes refer to [11].

Bounding Box. The bounding box captures the proportions of the drawing and is used to inform the classifier about the relative size of objects. We always normalize the actual size of the bounding box for each stroke with respect to the bounding box enclosing the whole sketch. This way proportions remain consistent even if sketches are drawn at different scales.

Centroid. The centroid is computed in the usual manner by averaging the x and y components of the points comprising the stroke. The centroid effectively captures the region of a stroke that carries the largest perceptual weight [9] and is used to inform the classifier about the relative position of strokes in the sketch. The centroid is also used to compute several other shape attributes described in [11].

Horizontal and Vertical Ordering. Horizontal and vertical ordering are very effective attributes that allow us to quantify the relation in the position of strokes without having to know the class of neighboring attributes in advance. Figure 3 shows a sample face with values of horizontal ordering for every stroke.

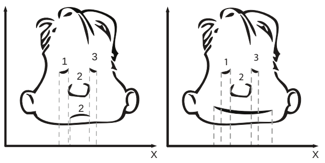


Fig. 3. Horizontal ordering. The image on the left shows a typical example in which the value of horizontal ordering is well defined. The image on the right shows an ambigous case in which the horizontal ordering for the mouth is not well defined.

Given an input sketch $S = \{s_1, s_2, \ldots, s_n\}$, let $ho : S \mapsto N$ be a function that maps a stroke to its value of horizontal ordering. The horizontal ordering is determined by applying the following rules in succession.

- Given strokes X and Y, $ho(X) = ho(Y)$ if the projection of their bounding boxes on the x axis overlaps

– If X and Y are not overlapping and the centroid of $X > Y$, then $ho(X) > ho(Y)$

It is not always possible to assign values of horizontal ordering and simultaneously satisfy both rules. Figure 3 shows one example of an ambiguous situation. If a stroke generates an ambiguity it means that the value of horizontal ordering is not relevant for that particular stroke and we can assign to it some arbitrary value. Referring again to Figure 3 we observe in fact that the value of horizontal ordering is essential to distinguish the left eye from the right eye, but it doesn't have any meaning for the mouth. On the other hand, the vertical ordering is very useful to distinguish the eyes from the mouth. We say that strokes s_i and s_j x-overlap iff the projection on the x axis of their bounding box overlaps and $width(s_1) < width(s_2)$. The algorithm for computing the horizontal ordering follows:

1. Scan strokes from the ones with smaller width to the ones with larger width and group them based on their x-overlap. A group is a vector of the form $g = [s_1, s_2, \ldots, s_n]$ where s_i is a stroke in the sketch
2. Compute the average x position for each group and arrange the values in a vector.
$$a = [avg_x(g_1), avg_x(g_2), \ldots, avg_x(g_n)]$$
3. Sort the vector a.
4. Assign a value of horizontal ordering to each g_i based on its sorted order in the vector.
5. Assign the value of horizontal ordering for group g_i to every stroke $s_i \in g_i$.

Depth. The depth attribute informs the classifier about the overlap of strokes. We say that stroke s_i overlaps s_j if the bounding box of s_i is fully contained in the bounding box of s_j. Note that this definition of overlap is different from the one given in section 5. Given a set of strokes $S = \{s_1, s_2, \ldots, s_n\}$, the depth satisfies the following rules:

– If s_i does not overlap with any stroke in the set $S \setminus \{s_i\}$ then its depth is zero
– If s_i overlaps with $s_j \in S \setminus \{s_i\}$ then $depth(s_i) < depth(s_j)$

The depth is computed as follows:

1. group all strokes that overlap in the same set.
2. Sort the items in the set based on the area of their bounding boxes.
3. Assign values of depth to each stroke in sorted order. From the way we defined overlap we are guaranteed that small strokes have a value of depth that is higher than the strokes that enclose them.

In practice we relax in our implementation the definition of overlap to detect overlap even if two strokes are partially overlapping.

8 Discussion and Results

Our classifier is based on a support vector machine and it is trained by the user. We are not making strong assumptions on the content of the sketches; therefore our classifier is expected to be flexible and work effectively for a variety of sketch-based applications. The main restriction on the capabilities of the classifier is determined by representation of sketches we construct by means of shape attributes. Specifically our method is designed to work best with diagrams or graphic drawings, such as the ones discussed in section 9. We assess the quality of the classifier on a set of controlled sketches we call clean sketches. We then extend the classifier to accomodate more complex sketches by using the concept of groupings. Our approach for grouping relies for the most part on the training set and it is therefore very adaptable. One of the most attractive aspects of grouping is that it gives us robustness against outliers.

All the concepts described in this paper are implemented in a framework called Sketch Analyzer. Several examples of our work and of Sketch Analyzer are shown in the video accompanying this paper. We tested our approach with sketches representing a house and a human face. We ran our experiments on a 2.13 Ghz Pentium 4 machine and the classifications took no more than 3 seconds to complete. Figure 4 show several sketches that were classified successfully. All the examples contain many outlier strokes that are not part of the training set, such as the window grille for the house. The example on the right contains two semantic groups that belong to the same class – the windows.

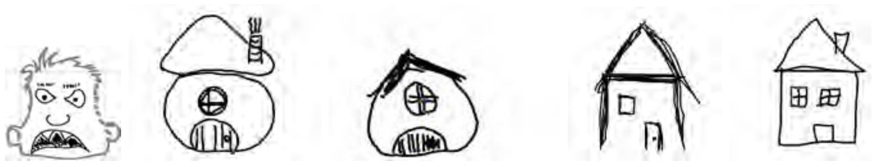


Fig. 4. Several sketches that are successfully handled by our technique using respectively a training set for the components of a face and a house. All examples contain several outlier strokes and are designed to put groupings to the test. The windows of the house on the right are an example of semantic grouping.

9 Application: Driving Facial Expressions

We used the concepts described in this paper to implement a sketch-based interface for driving facial expressions. The interface is described in detail in [11] and several examples are shown in the accompanying video. The classifier described in this paper along with grouping is used to recognize the components of the sketch and establish a correspondence with the face model. Each component of the face is then matched with a library of templates to establish the primary features of the facial expression. Lastly, the result is refined by varying the intensity of the various aspects of the facial expression through a parameterization of strokes.

10 Conclusion

We presented an approach for the classification of strokes that is based on SVM and grouping. Our work can handle fairly complex sketches and strives for robustness in the presence of outliers. Shape attributes are one of the key components of our approach and are designed to produce a well-behaved statistical characterization of strokes for generic drawings. Our results show that the classifier can generate robust classifications even with relatively small training sets. Two areas that need improvement and are interesting avenues for future research are the heuristics we use for ranking classification results and pruning the space of groupings, especially for semantic groupings. This work was partially supported by NSF grant No. CCF-0429983. We would also like to thank eFrontier for their generous donations of software licences.

References

1. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH 1999. Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press/Addison-Wesley Publishing Co., New York (1999)
2. Chang, E., Jenkins, O.C.: Sketching articulation and pose for facial animation, 19–26 (2006)
3. Thorne, M., Burke, D., van de Panne, M.: Motion doodles: an interface for sketching character motion. ACM Trans. Graph. 23, 424–431 (2004)
4. Sharon, D., van de Panne, M.: Constellation models for sketch recognition. SBIM 2006, 19–26 (2006)
5. Shilman, M., Viola, P., Chellapilla, K.: Recognition and grouping of handwritten text in diagrams and equations. In: IWFHR 2004, pp. 569–574. IEEE Computer Society Press, Washington, DC, USA (2004)
6. Cheong, C.E., Kim, H.Y., Suh, J.W., Kim, H.: Handwritten numeral string recognition with stroke grouping. In: ICDAR 1999, p. 745. IEEE Computer Society, Washington, DC, USA (1999)
7. Saund, E., Mahoney, J., Fleet, D., Larner, D., Lank, E.: Perceptual organization as a foundation for intelligent sketch editing (2002)
8. Saund, E., Moran, T.P.: A perceptually-supported sketch editor. In: ACM Symposium on User Interface Software and Technology, pp. 175–184. ACM, New York (1994)
9. Arnheim, R.: Art and Visual Perception: A Psychology of the Creative Eye (1974)
10. Yang, C., Sharon, D., van de Panne, M.: Sketch-based modeling of parameterized objects. In: Eurographics Workshop on Sketch-Based Interfaces and Modeling, pp. 63–72 (2005)
11. Nataneli, G., Faloutsos, P.: Technical report: Sketching facial expressions. UCLA (2007), URL: `http://www.cs.ucla.edu/~nataneli/pages/publications.html`

Locally Adjustable Interpolation for Meshes of Arbitrary Topology

Shuhua Lai[1], Fuhua (Frank) Cheng[2], and Fengtao Fan[2]

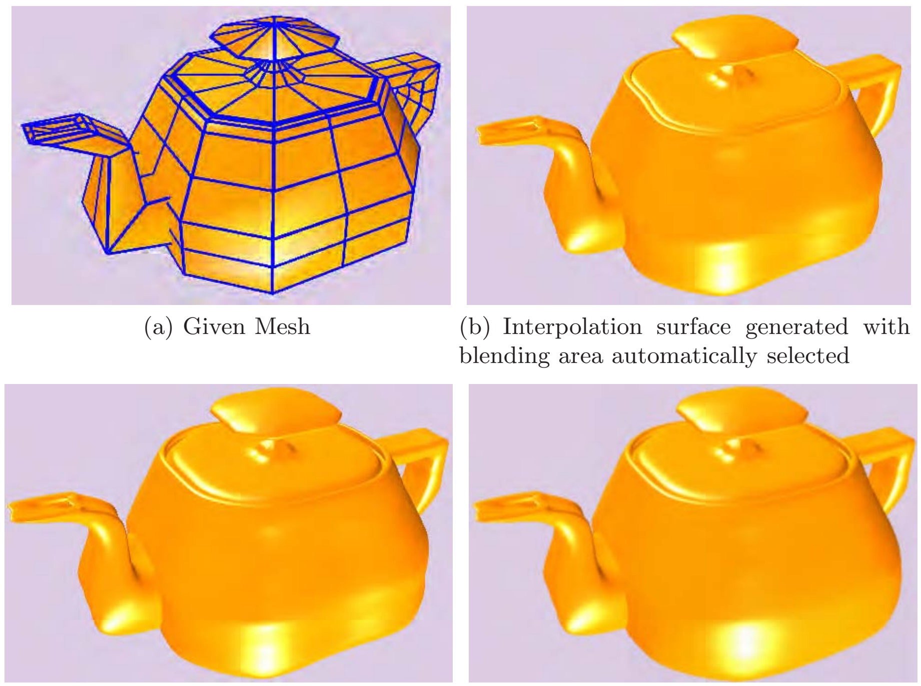
[1] Department of Mathematics & Computer Science, Virginia State University, Petersburg, VA 23806
[2] Graphics & Geometric Modeling Lab, Department of Computer Science, University of Kentucky, Lexington, Kentucky 40506

Abstract. A new method for constructing a smooth surface that interpolates the vertices of an arbitrary mesh is presented. The mesh can be open or closed. Normals specified at vertices of the mesh can also be interpolated. The interpolating surface is obtained by locally adjusting the limit surface of the given mesh (viewed as the control mesh of a Catmull-Clark subdivision surface) so that the modified surface would interpolate all the vertices of the given mesh. The local adjustment process is achieved through locally blending the limit surface with a surface defined by non-uniform transformations of the limit surface. This local blending process can also be used to smooth out the shape of the interpolating surface. Hence, a *surface fairing* process is not needed in the new method. Because the interpolation process does not require solving a system of linear equations, the method can handle meshes with large number of vertices. Test results show that the new method leads to good interpolation results even for complicated data sets. The new method is demonstrated with the Catmull-Clark subdivision scheme. But with some minor modification, one should be albe to apply this method to other parametrizable subdivision schemes as well.

1 Introduction

Constructing a smooth surface to interpolate the vertices of a given mesh is an important task in many areas, including geometric modeling, computer graphics, computer animation, interactive design, scientific visualization etc. The interpolating surface sometime is also required to interpolate normal vectors specified for some or all of the mesh vertices. Developing a general solution for this task is difficult because the required interpolating surface could be of arbitrary topology and with arbitrary genus.

Subdivision surfaces were introduced as an efficient technique to model complex shapes [1]. But building a connection between a given mesh and an interpolating subdivision surface has never really been successful when the number of vertices of the given mesh is large. One exception is a work published recently [7]. In that paper, an iterative interpolation technique similar to the one used in [5] for non-uniform B-spline surfaces is proposed for subdivision surfaces. Since



(a) Given Mesh

(b) Interpolation surface generated with blending area automatically selected

(c) Interpolation surface generated with more user selected blending areas around the upper portion of the teapot body

(d) Interpolation surface generated with more user selected blending areas around the lower portion of the teapot body

Fig. 1. Interpolation with local control

the iterative approach does not require solving a system of linear equations, it can handle meshes with large number of vertices. But the paper fails to prove the convergence of the iterative process.

In this paper a new method for constructing a smooth surface that interpolates the vertices of a given mesh is presented. The mesh can be of arbitrary topology and can be open or closed. Normal vectors specified at any vertices of the mesh can also be interpolated. The basic idea is to view the given mesh as the control mesh of a Catmull-Clark subdivision surface and locally adjust the limit surface of the given mesh so that the resulting surface would not only interpolate vertices of the given mesh, but also possess a satisfactory smooth shape. The local adjustment process is achieved through blending the limit surface S with a blending surface T. By performing the blending process at different selected areas, we are able to (1) ensure the modified surface would interpolate the given mesh, (2) prevent it from generating unnecessary undulations, and (3) smooth out the shape of the resulting surface.

The new method has two main advantages. First, since we do not have to compute the interpolating surface's control mesh, there is no need to solve a system of linear equations. Therefore, the new method can handle meshes with large number of vertices, and is more robust and stable. Second, because the local blending process can be used to smooth out the shape of the interpolating surface, a *surface fairing* process is not needed in the new method.

An example of this interpolation process is shown in Fig. 1. The surfaces shown in Figures 1(b), 1(c) and 1(d) all interpolate the mesh shown in Fig. 1(a). The blending areas in Fig. 1(b) are automatically selected by our system while Fig. 1(c) and Fig. 1(d) have user selected blending areas in the upper portion and lower portion of the teapot body afterward. It is easy to see from Fig. 1 that local control is necessary when better quality interpolating surfaces are needed.

2 Previous Work

There are two major ways to interpolate a given mesh with a subdivision surface: *interpolating subdivision* [2,4,10,13] or *global optimization* [3,6,8]. In the first case, a subdivision scheme that interpolates the control vertices, such as the Butterfly scheme [2], Zorin et al's improved version [13] or Kobbelt's scheme [4], is used to generate the interpolating surface. New vertices are defined as local affine combinations of nearby vertices. This approach is simple and easy to implement. It can handle meshes with large number of vertices. However, since no vertex is ever moved once it is computed, any distortion in the early stage of the subdivision will persist. This makes interpolating subdivision very sensitive to irregularity in the given mesh. In addition, it is difficult for this approach to interpolate normals or derivatives.

The second approach, *global optimization*, needs to build a global linear system with some constraints [9]. The solution to the system is a control mesh whose limit surface interpolates the vertices of the given mesh. This approach usually requires some fairness constraints in the interpolation process, such as the energy functions presented in [3], to avoid undesired undulations. The problem with this approach is that a global linear system needs to be built and solved. It is difficult for this approach to handle meshes with large number of vertices.

There are also techniques that produce surfaces to interpolate given curves or surfaces that near-interpolate given meshes. But those techniques are either of different natures or of different concerns and, hence, will not be discussed here.

3 Locally Adjustable Interpolation

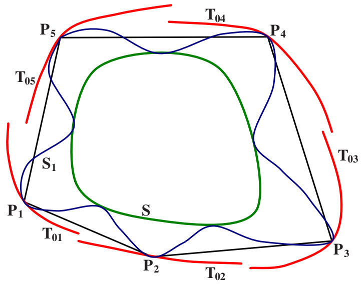
3.1 Basic Idea

Given a 3D mesh with n vertices: $P = \{\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_n\}$, the goal here is to construct a new surface that interpolates P (the vertices of P, for now). Contrast to existing interpolation methods, which either construct a new mesh whose limit surface interpolates P or perform interpolating subdivision schemes on the input mesh, we perform interpolation by manipulating the limit surface S of the given mesh directly. The basic idea is to *push* or *pull* the limit surface of the given mesh in vicinity of selected points so that the modified surface interpolates the given mesh and, in the meanwhile, prevent it from generating unnecessary undulations and maintain its smoothness. The push or pull process is done by constructing a new surface T, and blending T with S. T must be relatively easy to construct and interpolating P initially. For example, in Fig. 2(a), T is composed of five separate segments: T_{01}, T_{02}, T_{03}, T_{04} and T_{05}, and each of them interpolates a point of $P = \{P_1, P_2, P_3, P_4, P_5\}$. T and S must be blended in a way such that the resulting surface interpolates P and is C^2-continuous almost everywhere. The interpolating surface can be defined as follows:

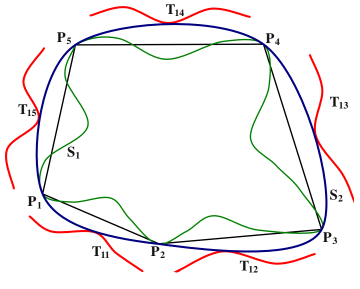
$$\bar{S} = S(u,v)W(u,v) + T(u,v)(1 - W(u,v)), \tag{1}$$

where $0 \leq W(u,v) \leq 1$ is a C^2-continuous weight function satisfying the property: $\lim_{(u,v)\to 0} W(u,v) = 0$. T must be parametrized so that $T(0,0) = P_i$, $(1 \leq i \leq n)$ and is C^2-continuous everywhere except at $(0,0)$ (where it is at least C^1-continuous) and except at $\{(u,v) \mid W(u,v) = 1\}$ (where it is not even necessary to be C^0-continuous). Therefore $\bar{S}$ is guaranteed to interpolate P and is C^2-continuous everywhere except at some extraordinary points.

Usually during the initial blending process, quality of the resulting interpolating surface would not be good enough yet. For example, the blue curve in Fig. 2(a), denoted S_1, is the resulting curve of the first blending process. As we can see, S_1 has a lot of undesired undulations although it interpolates the given mesh P exactly. To improve the shape of the interpolating surface and to reduce unnecessary oscillations, a second blending process can be performed in the vicinity of some selected points. For example, in Fig. 2(b), a second blending process is performed in the vicinity of all the *edge points* of the given mesh. To carry out the second blending process, a different blending surface T_1 has to be constructed. T_1 does not have to interpolate P. However, T_1 must not change the position of the images of P on the limit surface. In other words, the domain involved in constructing T_1 should be smaller than the domain of S_1, so that the images of P would not be involved in the construction process of T_1. For example, in Fig. 2(b), $T_1 = \{T_{11}, T_{12}, T_{13}, T_{14}, T_{15}\}$ and the images of P_is are not involved in the construction of T_1. Once T_1 is constructed, T_1 can be blended with S_1



(a) First blending



(b) Second blending

Fig. 2. Basic idea of the new interpolation method

similarly to get S_2 as follows: $S_2 = S_1(u,v)W_1(u,v) + T_1(u,v)(1 - W_1(u,v))$, where $W_1(u,v)$ is a blending function similar to $W(u,v)$ in Eq. (1), except $W_1(u,v)$ is constructed for vicinity of edge points, while $W(u,v)$ is constructed for vicinity of vertex points. Because the images of P are not involved in the construction process of T_1, the images of P are not affected by the above blending process. Hence interpolation requirement still holds.

Note that the blending process is done for individual pieces. For example, in Fig. 2(b), it is done for the pieces corresponding to $T_{1i},\ 1 \leq i \leq 5$, independently. Because T_1 is not required to interpolate P, not every $T_{1i},\ 1 \leq i \leq 5$, has to be blended with the corresponding piece of S_1. A blending process is performed for a selected region only if the shape of the surface is not good enough in that area. Hence, the blending process for each region is an optional operation.

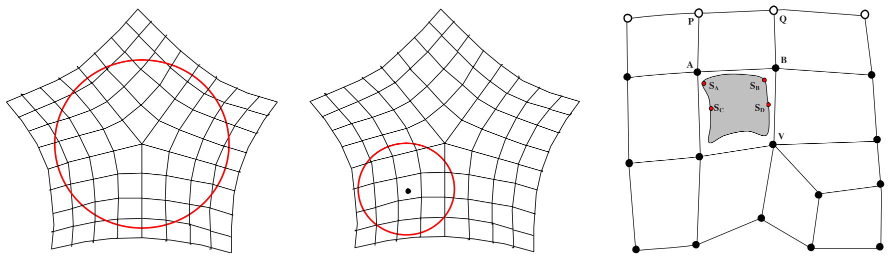
As we can tell from Fig. 2, the shape of S_2 is much better than that of S_1. However, if necessary, a third or even more blending processes can be performed on the resulting surface to further improve its quality. While the above idea seems to be simple and straightforward, the key here is how to construct a $T(u,v)$ for each local blending process and how to construct the corresponding blending weight function $W(u,v)$ such that the resulting interpolating surface is smooth and oscillation-free. The construction process of T will be shown in the next section. For consistency, we denote the $(i+1)$st blending surface T by T_i, and use T as a general reference to all possible levels of T_i.

3.2 Construction of T_0

Note that only T_0 is required to interpolate P, not the subsequent T_i, $i \geq 1$. T_0 can be constructed in several different ways. In this paper we construct T_0 by linearly transforming pieces of S in 3D object space. Note that T_0 is not necessary to be C^0-continuous at parameters where $W(u,v) = 1$. The affine transformation matrix can be chosen in a way such that T_0 interpolates P and, in the meanwhile, changes the original limit surface as little as possible. For example, in Fig. 2(a), to get T_0 for the limit surface S defined by $P = \{P_1, P_2, P_3, P_4, P_5\}$, we simply translate S segment by segment, in the direction from the image of P_i to P_i, and then scale each segment with appropriate scaling factors for X, Y and Z components such that T_0 interpolates P and has an appropriate size. Consequently, as it shows in Fig. 2(a), T is represented by five segments: $\{T_{01}, T_{02}, T_{03}, T_{04}, T_{05}\}$ and they are not C^0-continuous. But after a blending process using equation (1), we get a surface S_1 which smoothly interpolates P.

3.3 Local Parameterization of Subdivision Surfaces

The blending process defined by eq. (1) is performed on regions of the limit surface. Hence a local parameterization is needed for each region of the limit surface where a blending process is to be performed. Many local parameterization methods have been reported recently. In this paper we follow Reif's approach [11]. Reif's approach maps an extraordinary point to $(0,0)$ and is based on the *characteristic map* of a subdivision scheme [11]. A characteristic map is defined



(a) Blending around an original vertex (b) Blending around an arbitrary point (c) Handling open meshes

Fig. 3. Left & Center: Parameter space blending areas. Right: handling open meshes.

by calculating the limit of subdivision on a 2D mesh formed by the two sub-dominant eigenvectors of the local subdivision matrix [11]. The characteristic map for Catmull-Clark subdivision scheme around an extraordinary vertex of valence n is based on the topology of the 2-ring neighborhood of vertices around the extraordinary vertex. Thus, the two sub-dominant eigenvectors have $6n + 1$ entries each. Also as a normalization rule, the two sub-dominant eigenvectors should be scaled such that the parameters (u, v) at the end-points of edges emanating from the extraordinary vertex have coordinates $(\cos(i\alpha), \sin(i\alpha))$, $i = 1 \cdots n$, where $\alpha = 2\pi/n$ (see Fig. 3(a)).

3.4 Blending Around an Extraordinary Point

With parametrization available, it is now possible to perform blending process on regions of the limit surface. To maximize the blending area around a vertex, we define the blending region in the parameter space by the condition: $u^2 + v^2 \leq 1$. This is a circle centered at the extraordinary point in the parameter space (See Fig. 3(a)). It should be pointed out that the blending area defined here is different from the one used in [12], which is defined by $u^2 + v^2 \leq \lambda_n$ with λ_n being the sub-dominant eigenvalue of the subdivision matrix corresponding to a valence n extraordinary vertex. The reason for this difference is because we want to maximize the blending areas and overlapping of blending areas does not matter in our case. The blending weight function $W(u, v)$ must satisfy the condition $0 \leq W(u, v) \leq 1$ in the blending region and has to be at least C^2-continuous everywhere. We follow Levin's approach [12] to define $W(u, v)$ as follows: $W(u, v) = (u^2 + v^2)(3(u^2 + v^2) - 8\sqrt{(u^2 + v^2)} + 6)$. It is easy to see that $W(u, v)$ satisfies $0 \leq W(u, v) \leq 1$ in the region $u^2 + v^2 \leq 1$ and is C^2-continuous everywhere. At the extraordinary point, $W(u, v)$ approaches zero. When near the boundary of the blending (i.e. $u^2 + v^2 = 1$), $W(u, v)$ approaches 1, with zero partial derivatives up to order 2. Hence, the resulting surface is guaranteed to interpolate the given mesh and, meanwhile, cancels out the irregularity and discontinuity of the blending surface T.

3.5 Blending Around an Arbitrarily Selected Point

Because it is locally adjustable, our method allows local blending to be performed in regions around arbitrarily selected points of the surface, not only the images of the control vertices. For example, in Fig. 3(b), to adjust the interpolating surface in a small area around the selected point (u_0, v_0), the one marked with a black solid circle, we find the biggest circle in the parameter space whose center is (u_0, v_0) (the red circle in Fig. 3(b)). This circle defines the blending area for the selected point. To find the radius for the blending circle, we compare all the distances from (u_0, v_0) to all the boundary parameter values in the parameter space and the smallest one is the radius of our blending area, denoted r_0. In addition, as mentioned above, the blending area should not include the parameter point (0,0). So we also need to compare r_0 with the distance between $(0, 0)$ and (u_0, v_0) and the smaller one is called r. Therefore the blending area for the selected point (u_0, v_0) can be defined as follows: $(u - u_0)^2 + (v - v_0)^2 < r^2$. The corresponding blending weight function is defined by the following quadrtic formula [12]: $W(u, v) = \rho^2(3\rho^2 - 8\rho + 6)$, where $\rho = \sqrt{(u - u_0)^2 + (v - v_0)^2}/r$. It is easy to see $\rho \leq 1$ and $W(u, v)$ satisfies $0 \leq W(u, v) \leq 1$ in the blending region and is C^2-continuous everywhere. Note that at the selected point, $W(u, v)$ approaches zero. When near the boundary of the blending region, $W(u, v)$ approaches 1, with zero partial derivatives up to order 2. Consequently, it can still cancel out the irregularity and discontinuity of the blending surface T while locally modify the shape of the interpolating surface.

3.6 Construction of Blending Surface T_i

In the above section, we have discussed how to construct an initial blending surface T_0 around vertices to be interpolated. In this section, we show how to construct a blending surface T_i around an arbitrarily selected point. Again, we can use Affine transformation to construct T_i from S_i. The scaling factor components of the affine transformation matrix can be similarly determined, by simply comparing the dimensions of T_i and S_i. The question is how to determine the offset components of the affine transformation matrix. Note that, unlike the case of T_0 where the offset components of the affine transformation matrix are determined by the vertex to be interpolated and its limit point on S, in this case, there is no point in the given mesh that corresponds to the selected point on S_i. Therefore, the offset components for an arbitrarily selected point cannot be determined directly. We propose to determine the offset vector for each selected 3D point by constructing a Hermite surface for the patch that covers the selected point. For example, if the offset vectors for the four corner vertices of the patch are D_1, D_2, D_3 and D_4, then we construct a Hermite surface patch $H(u, v)$ based on D_1, D_2, D_3 and D_4. The tangent vectors at the four corners required for the construction of H are set to the partial derivatives of the limit surface S at the four corners. The offset vector for a selected point with parameter value (u_0, v_0) in S_i, now can be simply set to $H(u_0, v_0)$.

3.7 Interpolation of Normal Vectors

Direction of normal vectors specified at vertices of the given mesh can also be interpolated. The key is to modify the construction process of the blending surface T_0 so that it would have the same normals (actually the same partial derivatives) at the extraordinary points. This can be easily achieved by rotating each piece of T_0 with appropriate X, Y and Z rotation factors after the above mentioned translation and scaling process. This is possible because each piece of T_0 interpolates only one point of P. Hence we have a blending surface T_0 that not only interpolates the given mesh P but normals specified at some or all vertices of P as well. Because the value of $W(u, v)$ and its first partial derivatives at $(0, 0)$ are all zero, the resulting interpolating surface $\bar{S}$ then satisfies

$$\frac{\partial T(u,v)}{\partial u} = \frac{\partial \bar{S}(u,v)}{\partial u} \quad \text{and} \quad \frac{\partial T(u,v)}{\partial v} = \frac{\partial \bar{S}(u,v)}{\partial v}.$$

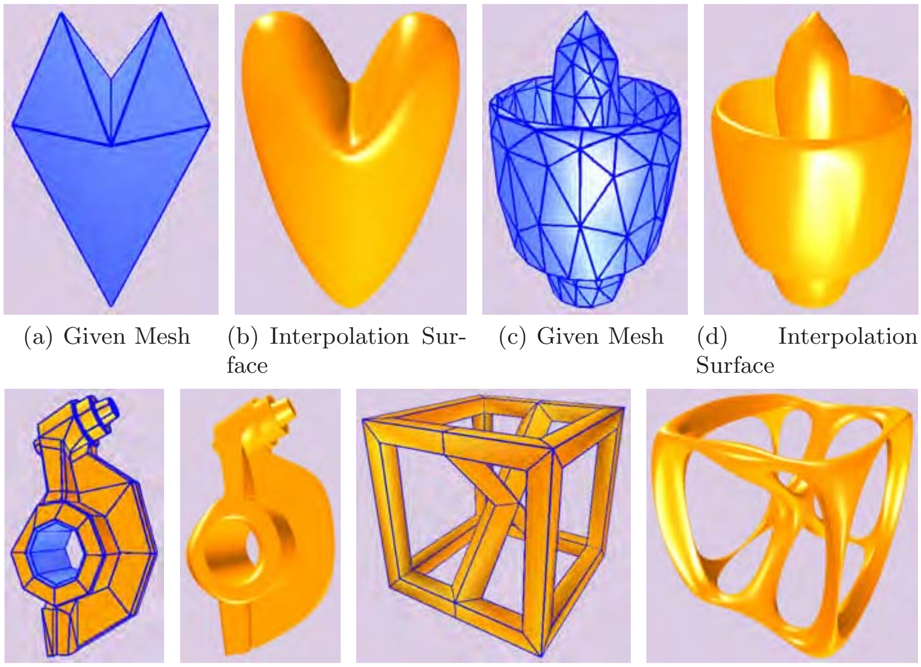
In other words, $\bar{S}$ and T_0 have the same normal. Hence, with one more Affine transformation we can construct an surface that not only interpolates the given mesh, but normals at all or some of the vertices of the mesh as well.

3.8 Handling Open Meshes

Interpolation of open meshes can be achieved by making minor modifications to the above proposed local blending interpolation process for closed meshes. Note that our method is locally adjustable. Hence a point on the limit surface actually can be moved to anywhere, as long as the interpolation requirement is satisfied. Consider the mesh shown in Fig. 3(c) where vertices marked with circles, like P and Q, are boundary vertices and vertices marked with solid circles, such as V, A and B, are interior vertices. The shaded surface patch is the corresponding limit surface where $S(A)$ and $S(B)$ are the images of A and B, respectively, $S(C)$ and $S(D)$ are the images of the corresponding edge points, respectively. According to our interpolation method, $S(A)$ and $S(B)$ should be moved to A and B, respectively. However, to interpolate the boundary points P and Q, we can modify our approach such that $S(A)$ and $S(B)$ are moved to P and Q, respectively, and $S(C)$ and $S(D)$ are moved to A and B, respectively. The resulting surface then interpolates all the vertices of the given open mesh.

4 Test Results

The proposed techniques have been implemented in *C++* using *OpenGL* as the supporting graphics system on the Windows platform. Quite a few examples have been tested with the techniques described here. All the examples have extra-ordinary vertices. Some of the tested results are shown in Fig.s 1 and 4. From these examples we can see smooth and visually pleasant shapes can be obtained by locally adjusting the original limit surfaces.



(a) Given Mesh (b) Interpolation Surface (c) Given Mesh (d) Interpolation Surface

(e) Given Mesh (f) Interpolation Surface (g) Given Mesh (h) Interpolation Surface

Fig. 4. Interpolation Examples

All the interpolation shown in Figures 1 and 4 are done with at least two blending processes. First one is done with T_0, which is based on all the given control vertices. T_1 for the second blending process is based on all edge points of the given mesh. Some figures in the examples went through more blending processes to further improve quality of the interpolating surface. T_i's for those blending processes are selected based on, for example, face points of all patches, or parameter values $(\frac{1}{2^j}, \frac{1}{2^k})$, where j and k are integers. User interaction is also possible. For example, Fig. 1(b), Fig. 1(c) and Fig. 1(d) all interpolate the given mesh shown in Fig. 1(a), but Fig. 1(c) and Fig. 1(d) are obtained with more local adjustment on the upper and lower part of the teapot body. The other parts are not adjusted, hence they are exactly the same as those shown in Fig. 1(b). Fig. 1 shows, with user local adjustment, a better shape can be obtained after some automatic blending processes.

The mesh shown in Fig. 1(a) consists of four separate open meshes: lid, handle, body and spout. Although each of these meshes can be interpolated separately, Fig. 1(b) and Fig. 1(c) are generated by regarding them as a single mesh. The mesh shown in Fig. 4(c) (its interpolation surface is shown in Fig. 4(d)) is another example of an open mesh with disconnected boundaries.

5 Summary

A new interpolation method for meshes with arbitrary topology is presented. The interpolation is a local process, it does not require solving a system of linear equations. Hence, the method can handle data set of any size. The interpolating surface is obtained by locally adjusting the limit surface of the given mesh so that the modified surface interpolates all the vertices of the given mesh. This local adjustment process can also be used to smooth out the shape of the interpolating surface. Hence, a *surface fairing* process is not needed in the new method. The new method can handle both open and closed meshes. It can interpolate not only vertices, but normals as well. The resulting interpolating surface is not a Catmull-Clark subdivision surface. It does not even satisfy the convex hull property [12]. But the resulting interpolating surface is guaranteed to be C^2 continuous everywhere except at some extraordinary points, where it is C^1 continuous.

Acknowledgement. Work of the second and third authors is supported by NSF (DMI-0422126) and KSTC (KSTC-144-401-07-015).

References

1. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design 10(6), 350–355 (1978)
2. Dyn, N., Levin, D., Gregory, J.A.: A butterfly subdivision scheme for surface interpolation with tension control. ACM TOG 9(2), 160–169 (1990)
3. Halstead, M., Kass, M., DeRose, T.: Efficient, fair interpolation using Catmull-Clark surfaces. ACM SIGGRAPH, 35–44 (1993)
4. Kobbelt, L.: Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In: Computer Graphics Forum, Eurographics, vol. 15 (1996)
5. Lin, H., Wang, G., Dong, C.: Constructing Iterative Non-Uniform B-spline Curve and Surface to Fit Data Points. Science in China 47, 315–331 (2004)
6. Lai, S., Cheng, C.F.: Similarity based Interpolation using Catmull-Clark Subdivision Surfaces. The Visual Computer 22(9-11), 865–873 (2006)
7. Maekawa, T., Matsumoto, Y., Namiki, K.: Interpolation by geometric algorithm. Computer-Aided Design 39, 313–323 (2007)
8. Nasri, A.H.: Surface interpolation on irregular networks with normal conditions. Computer Aided Geometric Design 8, 89–96 (1991)
9. Nasri, A.H., Sabin, M.A.: Taxonomy of interpolation constraints on recursive subdivision curves. The Visual Computer 18(4), 259–272 (2002)
10. Schaefer, S., Warren, J.: A Factored Interpolatory Subdivision Scheme for Quadrilateral Surfaces. Curves and Surface Fitting, 373–382 (2002)
11. Reif, U.A: unified approach to subdivision algorithms near extraordinary points. Computer Aided Geometric Design 2, 153–174 (1995)
12. Levin, A.: Modified Subdivision Surfaces with Continuous Curvature. In: Proceedings of SIGGRAPH, pp. 1035–1040 (2006)
13. Zorin, D., Schröder, P., Sweldens, W.: Interpolating Subdivision for meshes with arbitrary topology. In: ACM SIGGRAPH, pp. 189–192. ACM, New York (1996)

A GPU-Based Algorithm for Building Stochastic Clustered-Dot Screens

Meng Qi, Chenglei Yang, Changhe Tu, Xiangxu Meng, and Yuqing Sun

School of Computer Science and Technology, Shandong University, Jinan 250100
7777vivian@sohu.com, {chl_yang,chtu,mxx,yuqing_sun}@sdu.edu.cn

Abstract. In industrial pattern reproduction, clustered-dot screens are usually created to transform continuous tone image into halftone image for batch printing. But the algorithms generating clustered-dot screens are usually difficult to process large image because they are very slowly and need lot of memory. In addition, the generated halftone image often have periodic patterns, leading to poor tone reproduction. In this paper, a GPU-based algorithm for building stochastic clustered-dot screens is proposed. In the algorithm, after stochastically laying screen dot centers within a large dither matrix, Voronoi diagram is constructed to obtain the region of each screen dot, which is implemented with GPU. Then, each screen dot's region is filled to get the stochastic clustered-dot screens, where a better gray density filling method that can be implemented easily on GPU is used. Experiments show the method can generate screens faster and with less memory than traditional algorithms. Moreover, in a halftone image generated by our method, the details and highlight part can be better expressed.

Keywords: digital halftoning, GPU, stochastic clustered-dot screen, Voronoi Diagram.

1 Introduction

For many years, in printing and dyeing, chinaware, weave, silk, washrag, carpet, brand and other industries, how to reproduce continuous tone image with hundreds of gray levels and millions of colors using only one or limit colors is really a difficult but basically problem. The process of transforming continuous tone into binary image is called halftoning. Continuous tone image need to be transformed into halftone image before batch printing.

Digital halftoning is developed from the traditional halftoning. Error-diffusion algorithm is one kind of method for generating halftone images. General error-diffusion algorithms [1-3] are to compensate the quantization errors by distributing them on to the neighboring pixels of the input image. The problems has two main shortcomings: one is that invariable error-diffusion model can lead to periodically pattern especially in where the grayscale is changing smoothness or in the grayscale wedges; the other is that it is not easy to be parallelized. Another kind of methods that can generate halftone image utilizes Wang tiles to generate large point sets possessing a blue noise

Fourier spectrum and high visual quality [4]. But they and Error-diffusion algorithms are both not easy to be parallelized and batch printed.

In industrial pattern reproduction, clustered-dot screens are usually created to transform continuous tone image into halftone image for batch printing [5-6]. In this kind of method, for each output pixel, we can find its corresponding locations both in the dither array and in the input image, compare these pixel's intensity values, then write the pixel to the output image. The whole process is very efficient, only one comparison is needed, and each output device pixels can be computed independently, which enables the process to be parallelized and pipelined easily. But the algorithms generating clustered-dot screens are usually difficult to process large image because they are very slowly and need lot of memory, and are not easy to be implemented with GPU. In addition, the generated halftone image often have periodic patterns, leading to poor tone reproduction.

In this paper, a GPU-based algorithm for building stochastic clustered-dot screens is proposed. In the algorithm, after stochastically laying screen dot centers within a large dither matrix, Voronoi diagram is constructed to obtain the region of each screen dot, which is implemented with GPU. Then, each screen dot's region is filled with sets of different intensity levels to get the stochastic clustered-dot screens, where a better gray density filling method that can be implemented easily on GPU is used. Experiments show the method can generate screens faster and with less memory than traditional algorithms. Moreover, in a halftone image generated by our method, the details and highlight part can be better expressed.

2 Previous Relative Works

Stochastic screens can offer finer grain compared to traditional screen, and advance the quality of presswork. It also permits to superimpose more than three layers without moirés and can reproduce more colors, gray grades and finer images details. Recent years, multiple authors have proposed some excellent stochastic clustered-dot screens algorithm such as [5-6].

In [5], a large dither matrix comprising thousands of stochastically laid out screen dots is constructed by first laying out the screen dot centers. After Delaunay triangulation of the screen dot centers, the maximal region of each screen dot is computed and iso-intensity regions are created, and then fill each region with different intensity levels. The algorithm in [6] adopts similar method to generate the screens. They introduces a Delaunay triangulation algorithm using a uniformly grid, which exhibits linear time complexity to improve the speed. Then it fills the cluster (maximal region of each screen dot) according to area proportion and divides each cluster into several triangles and quadrangles, and deal with them one by one to generate screens.

Based on these work [5-6], we propose a GPU-based algorithm for building stochastic clustered-dot screens, which can generate screens faster and with less memory, and the details and highlight part in a halftone image generated by our method can be better expressed.

3 Algorithm Description

Like [5-6], our algorithm for building stochastic clustered-dot screens has three steps (see Fig.1):

Step1: randomly lay screen dot centers within a large dither matrix (see Fig. 1(a)).
Step2: compute each screen dot's maximal region by constructing Voronoi diagram (see Fig.1(b)).
Step3: fill every Voronoi region with a set of different gray levels to get stochastic clustered-dot screens (see Fig. 1(c)).

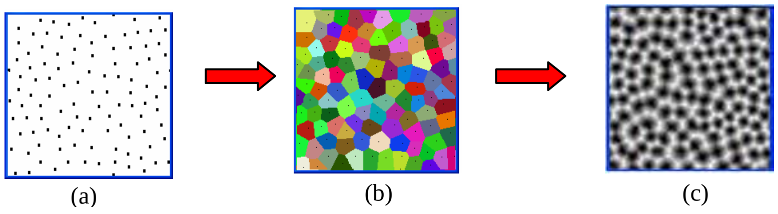


Fig. 1. the process of generating clustered-dot screens using our method. (a) lay screen dot centers. (b) compute Voronoi diagram. (c) fill Voronoi region to get screens.

In step 1, we use the disk-throwing method same as [5-6] to generate the screen dots by visiting the matrix along with the random space filling curve. As shown in the radial power graphic in Fig.2, its power distribution is well-proportioned, and the radial anisotropy spectrum graphic shows that the anisotropy value is very low. Hence, the screen dots' distribution has obvious blue noise property, and low frequencies may be avoided, i.e. the most representative frequencies present in the rendered halftone patterns should correspond to the screen dot periods.

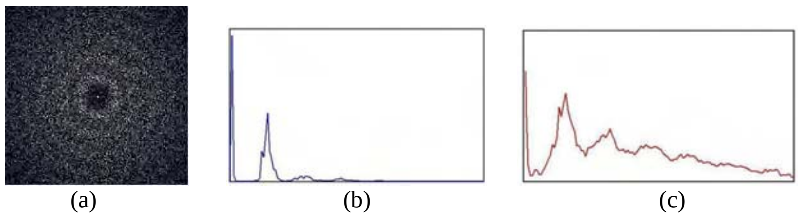


Fig. 2. (a) Fourier amplitude spectrums. (b) radial anisotropy spectrum. (c) radial anisotropy spectrum.

In step 2, in order to compute the cluster, [5-6] first compute Delaunay triangulation, by which to approximately construct Voronoi diagram of the screen dot centers. While our algorithm generate Voronoi diagram directly from the screen dots centers with GPU to generate clusters faster. In this paper, we use JFA method [7] to implement the step 2(Fig. 3(a) shows the results of JFA method in a 64*64 matrix). It is the first parallel algorithm for (approximated) Voronoi diagram in GPU with time independent of input size when the output resolution is invariant, that is, it is output sensitive where its running time is mainly depending on the output resolution (if no time is

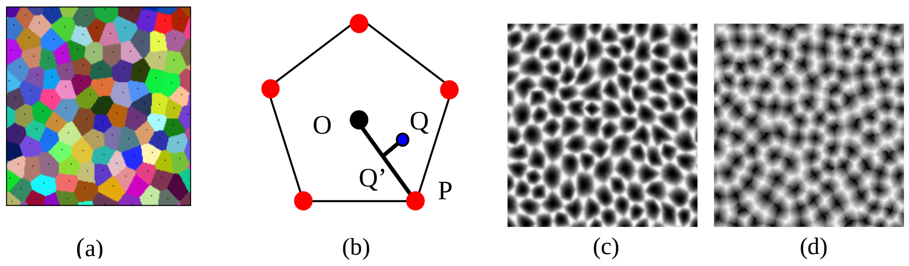


Fig. 3. (a) The Voronoi results of the JFA for radius r equals to 8. (b) a Voronoi region. (c) screen result using method in [6] with $r = 16$. (d) screen result using our method with $r = 16$.

charged to put the seeds into a texture, as in the case of an input is already a texture) [7]. Based on the method in step 2, step 3 can be implemented with GPU easily.

In step 3, we'll fill each dot center's Voronoi region (cluster) with a set of gray levels. In [5-6], the generated boundary of cluster is not smooth, even if some smoothing method is used. Further, their filling process is serial. Here, we propose a new method to fill the cluster so that we can finish all the filling in constant time complexity, and improve the screen quality obviously without smooth processing. We will describe the details in section 4.

4 Filling Clusters Using GPU

In step 3, we use different colors to distinguish different Voronoi regions. If a point has at least 2 different color neighbors belonging to other screen dot, we call it as *corner boundary*.

In order to fill the clusters, for each region of one screen dot center O, we find all corner boundaries firstly. And for each point Q in the region, we find its nearest corner boundary P, and then compute its gray level by linear interpolation according to the location relation of Q , O *and* P.

According to Fig.3(b), the filling process can be described as followings:

Step 3.1: Find corner boundaries.
Let the location of a given point Q be (x, y), its 8 neighbors are $(x+k_1, y+k_2)$, where $k = \{-1, 0, 1\}$, and k_1 and k_2 are not equal to 0 at the same time. Hence, just run a vertex program and a fragment program in turn, we can finish this step.

Step 3.2: Find nearest corner boundary for each point.
For a given point Q, we should find its correspond screen dot center O, and the nearest corner boundary P. Obviously, Q and P must have the same color. And any two adjacent screen dot center's distance is between r and $2*r$ [6], where r is the radii of the disk used in step 1. So when we use flooding method to find P, the total step may limit in r steps.

Step 3.3: Compute gray intensity level for each point.
In the following, we compute the projection point Q' on the line OP. We let the ratio of OQ' and OP as the gray level of the given point Q:

$$G(Q) = (|OQ'|) / |OP|) *255.$$

Fig.3(c) and Fig. 3 (d) show two stochastic clustered-dot screens produced by the method in [6] and our method respectively. The cluster's boundary in Fig. 3(c) is sharper than that in Fig. 3(d). And the whole screen looks like aggregate of sets of black dots and white dots in Fig. 3(c), while in Fig. 3(d) the transition between different intensity is much smoother. All these advantages indict our screen may increase the number of perceived colors on display devices with a limited number of intensity levels, and have better tone reproduction capability and detail resolution. We will show some experiment results and analyzes in the next section.

5 Experiment Results and Analyses

5.1 Time Analyses

Since our algorithm is based on GPU, the speed is faster than traditional methods. Fig. 4 shows the time cost comparison (Note: the time showed in the graph include the time of step 2 and step 3 only) between our method and the method in [6], where the disk radius *r* equals to 8, and the run environment is: CPU AMD Athlon™64 Processor 3500+, 2.21GHz, 2.00GB; GPU NVIDIA GeForce 7900 GT/GTO; VC++6.0.

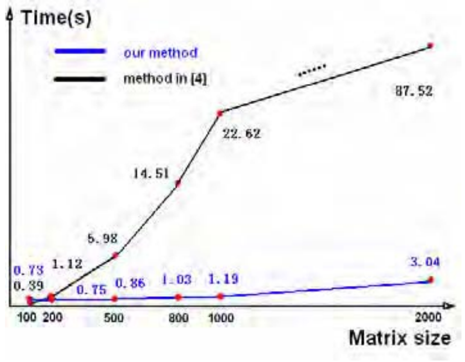


Fig. 4. Time comparison of the method in [6] and our method

In Fig.4, the black and blue lines show the time of the method in [6] and our method respectively. We find that when the matrix's size increases linearly, the time also increases linearly in [6]. However, when the matrix's size is large enough, the time cost is very high. For example, when the matrix's size is 1000*1000, the time for generating screen by the method in [6] is 22.62 second, and when the matrix's size is 2000*2000, the time is 87.52 second. However, using our screen algorithm, we can reduce the time for generating screens. Experiments show that when the matrix's size is 1000*1000, the time for generating screen by our method is 1.19 second, and even if the matrix's size is 2000*2000, we only need 3.04 second to compute the screen.

5.2 Quality Analyses

The disk radius *r* determines the minimal distance between the screen dots, i.e. determines the size of clusters. By changing the disk radius, the screen dot size can be adapted to the characteristics of particular printing devices. From appearance, the cluster in our screens are approximate to round shape, their boundaries are very smooth, the transition between different grays are very natural, so our screens are suitable to represent high quality images.

Fig. 5 shows some comparisons between our method and other methods, such as in [4] and [6]. In zone patterns, we may compare these methods' anti-aliasing ability. From the result, method in [6] and our method both have better anti-aliasing ability than other methods. Images produced by method in [4] and [6] have no obvious periodical patterns in the grayscale wedges, however, the grayscale transition is not smooth enough, and the number of perceived colors is fewer than our method.

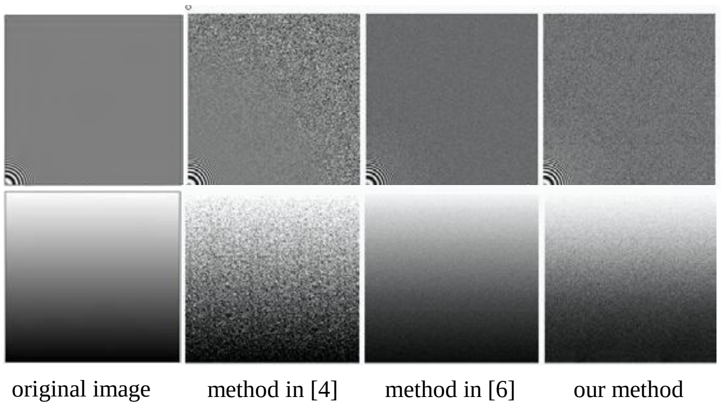


Fig. 5. Zone patterns (top) and grayscale wedges (bottom) produced by method in [4], method in [6], and our method

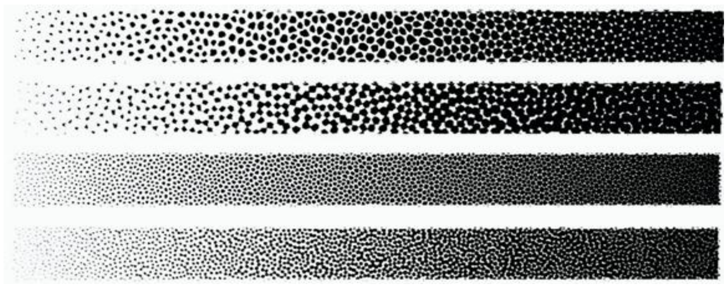


Fig. 6. Grayscale wedges. From top to bottom rendered respectively by the method in[6], r = 16; our method, r = 16;the method in[6], r = 8; our method, r = 8.

The grayscale wedges in Fig.6 are produced by method in [6] and our method, using two different radius (*r* = 8, and *r* = 16). From these we can see clearly that our method can get more perceived colors on display devices with a limited number of intensity levels.

In Fig.7, from the left to right, the binary images are produced by using method in [6], [4] and our method, respectively. In the right images produced by our method,

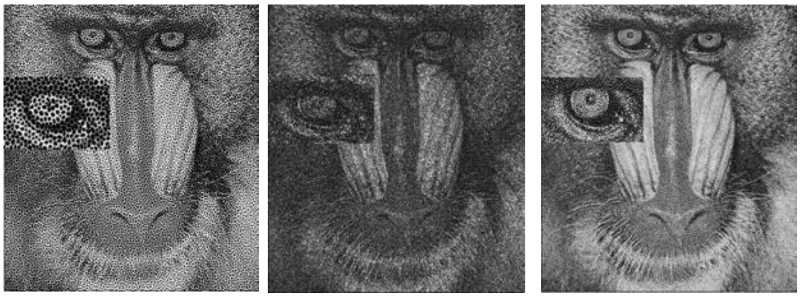


Fig. 7. a. Result images. From left to right, using method in [6], method in [4], and our method respectively.

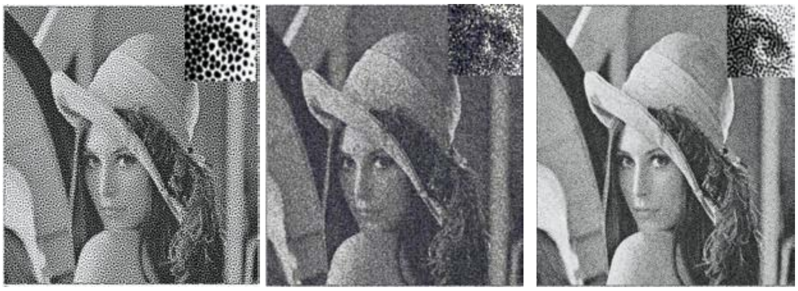


Fig. 7. b. Result images. From left to right, using method in [6], method in [4], and our method respectively.

the detail in eye's area, Lenna's hair, Lenna's hat texture, and Mandrill's fur, are better expressed. And in some highlight area, such as the Mandrill's bridge of nose, our method can produce better tone reproduction.

6 Conclusion and Future Work

For pattern reproduction in printing and dyeing, chinaware, weave, silk, washrag, carpet, brand and other industries, continuous tone image need to be transformed into halftone image before batch printing. Clustered-dot screens are usually used in this process. But it is usually difficult to process large image owing to speed and memory limit, and the generated halftone image often have periodic patterns, leading to poor tone reproduction.

In this paper, a GPU-based algorithm for building stochastic clustered-dot screens is proposed. Experiments show the method can generate screens faster and with less memory than traditional algorithms. Moreover, in a halftone image generated by our method, the details and highlight part can be better expressed. And the algorithm can be integrated into the *Integrated Computer Aided Patterns Design and Plate-making System* we have developed.

In this paper, we only consider linear interpolation of gray intensity levels, it may result in some linear discontinuity (jagging) in the screens (see Fig.3(d)), and some discontinuity (see Fig.8)of gray levels between the halftone image generated by our algorithm and the continuous tone image. In the future, we will try different interpolation methods such as quadratic and cubic interpolation to get better result.

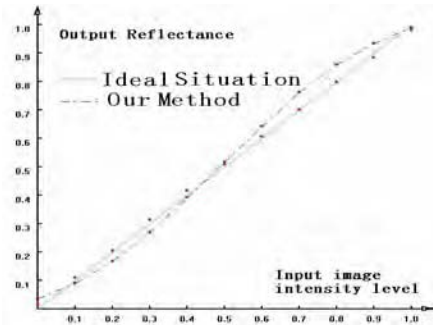


Fig. 8. Tone reproduction curves

Acknowledgement

We would like to acknowledge Dr. Rong Guodong for providing the codes of constructing Voronoi diagram based on GPU and giving some advice. And we also like to acknowledge the National Natural Science Foundation (60703028, 60473103, 60573181) and the National Key supported by Basic Research and Development Plan (2006CB303102).

References

1. Floyd, R.W., Steinberg, L.: An Adaptive Algorithm for SpatialGrey Scale. In: Proc. SID, vol. 17(2), pp. 75–77 (1976)
2. Ostromoukhov, V.: A Simple and Efficient Error-Diffusion Algorithm. In: Fiume, E. (ed.) Proceedings of SIGGRAPH 2001. Computer Graphics Proceedings, Annual Conference Series, pp. 567–572. ACM Press / ACM, SIGGRAPH, New York (2001)
3. Zhou, B., Fang, X.: Improving Mid-tone Quality of Variable-Coefficient Error Diffusion Using Threshold Modulation ACM SIGGRAPH 2003, Knuth, D.E.: Digital Halftones by Dot Diffusion. ACM Trans. on Graphics 6(4), 245-273 (1987)
4. Kopf, J., Cohen-Or, D., Deussen, O.: Dani Lischinski Recursive Wang tiles for real-time blue noise (siggraph 2006). ACM Trans. Graph. 25(3), 509–518 (2006)
5. Ostromoukhov, V., Hersch, R.D.: Stochastic Clustered-Dot Dithering. Color Imaging: Device-Independent Color, Color Hardeopy, and Graphic Arts IV 3648, 496–505 (1999)
6. Changhe, T., Rongjiang, P., Xiangxu, M.: An Algorithm for Building Stochastic Clustered-dot Screens. Journal of Computers 23(9), 938–942 (2000)
7. Rong, G., Tan, T.-S.: Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform. In: Proceedings of the 2006 symposium on Interactive 3D graphics and games, Redwood City, California,USA, March 14-17, 2006 (2006)

Optimized HLOD Refinement Driven by Hardware Occlusion Queries

Jean Pierre Charalambos[1,2], Jiří Bittner[3], Michael Wimmer[1], and Eduardo Romero[2]

[1] Vienna University of Technology
[2] National University of Colombia
[3] Czech Technical University in Prague

Abstract. We present a new method for integrating hierarchical levels of detail (HLOD) with occlusion culling. The algorithm refines the HLOD hierarchy using geometric criteria as well as the occlusion information. For the refinement we use a simple model which takes into account the possible distribution of the visible pixels. The traversal of the HLOD hierarchy is optimized by a new algorithm which uses spatial and temporal coherence of visibility. We predict the HLOD refinement condition for the current frame based on the results from the last frame. This allows an efficient update of the front of termination nodes as well as an efficient scheduling of hardware occlusion queries. Compared to previous approaches, the new method improves on speed as well as image quality. The results indicate that the method is very close to the optimal scheduling of occlussion queries for driving the HLOD refinement.

1 Introduction

Interactive visualization of complex models comprising millions of polygons is one of the fundamental problems in computer graphics. In order to achieve interactive rendering of such models, the amount of processed data has to be substantially reduced. Level-of-detail methods allow aggressive reduction of the amount of data sent to the GPU at the expense of sacrificing image quality. Particularly, hierarchical level-of-detail (HLOD) methods proved capable for interactive visualization of huge data sets by precomputing levels-of-detail at different levels of a spatial hierarchy. HLODs support out-of-core algorithms in a straightforward way, and allow an optimal balance between CPU and GPU load during rendering [1].

An orthogonal approach of reducing the amount of rendered primitives is occlusion culling [2]. Occlusion culling methods aim to quickly cull the invisible part of the model and render only its visible part. In order to achieve this task, most recent methods employ hardware occlusion queries (HOQs) [3,4].

The effects of HLODs and occlusion culling can be effectively combined [5,6]. Moreover, it was shown that HOQs can also be used to drive the HLOD refinement [7]. In this case, the occlusion queries allow more aggressive culling of the HLOD hierarchy, further reducing the amount of rendered primitives. However, due to the latency between issuing a HOQ and the availability of its result, the direct use of HOQs for refinement criteria causes CPU stalls and GPU starvation.

In this paper we introduce a novel traversal algorithm for HLOD refinement driven by HOQs. The algorithm minimizes CPU stalls and GPU starvation by predicting the HLOD refinement conditions using spatio-temporal coherence of visibility. As a result, it provides substantial speedup over previous methods while maintaining comparable image quality.

2 Related Work

Rendering very large models has received serious attention in recent years. Among other techniques, HLOD-based methods proved efficient for handling these datasets. HLOD systems either use polygonal representation of LODs [8], point-based representations [6], or a combination of both [9]. They commonly employ a *screen space error* (SSE) to drive the HLOD refinement [8,5]. As an alternative, Charalambos [7] proposed the *virtual multiresolution screen space error* (VMSSE) which also considers the degree of occlusion.

Occlusion culling methods are another popular technique for handling large models [2]. Recent methods employ HOQs mainly due to their efficiency and simplicity. The main goal of the recent techniques is to cope with the latency between issuing the query and availability of its results as well as reducing the number of issued queries [3,4].

The combination of occlusion culling and discrete LODs was addressed by Andújar *et al.* [10], who introduced the concept of *hardly visible sets*. In the context of view-dependent LOD El-Sana *et al.* [11] proposed *cell solidity values* with the same purpose in mind. Gobetti *et al.* [6] presented a method for integrating hardware occlusion queries into an HLOD-based system. This method copes with the query latency by using temporal coherence as proposed by Bittner *et al.* [3]. It does not, however, exploit the results of HOQs for driving the HLOD refinement.

Charalambos proposed a different method which also integrates occlusion culling into an HLOD-based system, and it additionally exploits the results of HOQs to drive HLOD refinement using the VMSSE [12]. In order to minimize the effect of latency of HOQs, this technique performs several simplifications, which may degrade the visual quality as well as the performance of the resulting algorithm. Moreover, these effects are more noticeable in the case of scenes with higher depth complexities [12]. In this paper we focus on these problems and propose an optimized method for efficiently scheduling the occlusion queries, which improves both the running time as well as the visual quality compared to the previous methods.

3 Integrating HLODs and HOQs

This section overviews the problem of integrating HLODs and HOQs, and briefly describes previous approaches. We first describe the visibility-based traversal of an HLOD hierarchy and then we focus on the computation of the visibility-based HLOD refinement criterion.

3.1 Visibility-Based HLOD Traversal

The HLOD algorithm recursively traverses the hierarchy starting at the root node. At every traversed node it evaluates a *refinement condition*. If this condition indicates that the node should be refined, the traversal continues with the children of the node. Otherwise, the traversal terminates and the LOD associated with the node is rendered. The set of nodes where the refinement stops forms the *front of termination nodes*. The refinement condition commonly uses the SSE, which corresponds to the projection of the model space error onto the screen [8,5]. The SSE for the given node is compared to a user-defined threshold given in pixels (τ), known as *pixels-of-error* [5].

The HLOD algorithm can be improved by integrating occlusion culling into the traversal of the hierarchy. Firstly, we can cull nodes which are completely invisible [6]. Secondly, for visible nodes we can use the results of HOQs in the refinement condition. This can be achieved by using the VMSSE [12,7], which modulates the SSE using the result of the HOQ.

By using the VMSSE we can reduce the number of rendered primitives. However, the HLOD refinement becomes dependent on the result of the HOQ, which is not readily available at the time when the refinement criterion is evaluated, due to the latency between issuing the HOQ and the availability of the result. Waiting for the result of the query would cause a CPU stall and in turn a GPU starvation. A way to cope with this problem is to predict the result of the HOQ and use the predicted value in the evaluation of the refinement condition. If the prediction were perfectly correct, we would only stop the refinement where it would have been stopped if the result of the query had already been available. A simple prediction method which uses the processing order of the nodes has been proposed by Charalambos [12]. However, this prediction is rather simplistic and inaccurate, leading to two main drawbacks: (1) When the prediction is too conservative, more nodes get refined and in turn more primitives are rendered, (2) when the prediction is too aggressive, the node is rendered, but the same node is then also refined when the result of the query is available. That means that the refined children are rendered together with their parent, which can cause visual artifacts.

In order to cope with this problem, our new algorithm predicts the refinement condition based on the criteria determined in the previous frame. The prediction is used to decide whether to immediately refine the node or stop the refinement, or delay the decision till the result of the query becomes available.

3.2 Virtual Multiresolution Screen Space Error

The classical HLOD refinement criterion is based on the SSE which bounds the error of projecting the simplified geometry onto the screen. The SSE only considers the distance of the object from the viewpoint, and disregards the occlusion of the object. However if the object is largely occluded, we most likely do not perceive subtle details in the remaining visible part of the object. This suggests that the computed SSE could be reduced if occlusion is considered [10,7].

In this paper we use a modification of the VMSSE proposed by Charalambos [7]. This method evaluates the *relative visibility* μ, which is the ratio of unoccluded pixels to the number of pixels an object projects to. The number of unoccluded pixels is computed by an HOQ, while the number of projected pixels is calculated analytically.

The method presented in [7] used the relative visibility μ to scale the SSE linearly. However a simple linear scaling of SSE might lead to visual artifacts because it does not take the possible distributions of visible pixels into account: For larger relative visibility, it is likely that larger *visibility windows* appear, which make errors due to decreased LOD levels more perceptible. A proper way for handling this issue would be to analyze the distribution of visible pixels in conjunction with visual masking analysis. However both techniques would be too costly for evaluating the HLOD refinement criterion in real time. A simple model which aims to reflect the possible clustering of visible pixels was proposed in [13]. It assumes that the likelihood of larger visibility windows to appear is proportional to relative visibility μ, and therefore modulates the SSE using a sigmoid *bias* function $B(\mu)$ (see Figure 1-a):

$$VMSSE = B_{s,t}(\mu) \cdot SSE, \tag{1}$$

where the bias $B_{s,t}(\mu)$ is a function of μ with user-defined parameters s and t. $t \in [0,1]$ is the *movable turn over* point and $s \in [0,1]$ is the *smoothness*. The closer the movable turn over point t is to 0, the more conservative the SSE modulation is, i.e. the SSE will be modulated strongly just if only a few unoccluded pixels exist. The *smoothness* parameter s linearly interpolates between μ and the unsmoothed sigmoid. Particularly, if $s = 0$ then the sigmoid function becomes equal to relative visibility (μ).

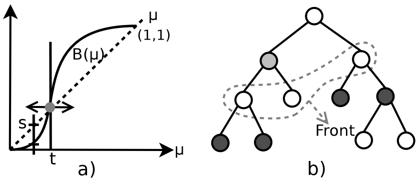


Fig. 1. a) Sigmoid bias function $B_{s,t}(\mu)$ used to compute the VMSSE. In the region defined by the halfspace $\mu \leq t$ the function provides aggressive scaling, while in the region defined by the halfspace $\mu \geq t$ the scaling is rather conservative. The s value is used to control the smoothness of the curve. b) Candidates for update of the front of termination nodes. In light grey we show nodes for which refinement has stopped for all its children. In dark grey we show nodes one level below the termination nodes.

4 Coherent HLOD Culling Algorithm

This section describes the proposed HLOD culling algorithm. We first describe the complete traversal algorithm and then we focus on its crucial part, which predicts the HLOD refinement condition by using temporal coherence.

4.1 Visibility-Based HLOD Traversal

The aim of our new algorithm is to perform efficient traversal of the HLOD hierarchy while using visibility information to drive the HLOD refinement. A naive algorithm would issue an occlusion query for every traversed node, wait for its result, compute the refinement condition, and decide whether to descend the hierarchy. Waiting for the result of occlusion is costly as it stalls the CPU and in turn causes GPU starvation. Our algorithm solves this problem by predicting the refinement criterion using temporal coherence of visibility. When proceeding from one frame to the next, it is most likely that refinement will stop at the same set of nodes where it has stopped in the previous frame. The exceptions are when refinement is shifted up or down from the current level of the node due to a change in its VMSSE.

Our coherent HLOD culling algorithm proceeds as follows: we start the traversal of the HLOD hierarchy at the root node. At each traversed node, we predict the refinement condition for the node based on the refinement conditions and the results of occlusion queries from the previous frame (the prediction will be described in detail in the next section). The prediction indicates one of the following actions: (1) *refine*, (2) *stop* refinement, or (3) *delay* the decision to the moment when the visibility of the node for the current frame is known.

In case (1), the children of the node are traversed by putting them in the priority queue. In case (2) and (3), we issue a HOQ for the node and put it in the query queue. In case (2), we also render the geometry associated with the node immediately and stop the refinement. In case (3), we delay the processing of the node until the result of the HOQ becomes available in the query queue. The decision is then made using the updated information about the visibility of the node: If the node is invisible, it is culled. If the VMSSE is lower than the threshold, refinement stops and the geometry of the node is rendered. Otherwise the refinement continues by inserting the children of the node in the priority queue.

The new traversal algorithm is outlined in Figure 2. Note that the differences to the traversal algorithm of Gobetti *et al.* [6] were underlined. Also note that the function $CalcSSE()$ computes the node SSE [8], which is a quick operation.

4.2 Predicting the HLOD Refinement Condition

The crucial part of the new method is the prediction of the refinement condition based on the relative visibility. As stated in the previous section, the prediction suggests either refine, stop, or delay. Let us first analyze the consequences of these actions for the traversal algorithm:

```
PriorityQueue.Enqueue(hierarchy.Root);
while ¬PriorityQueue.Empty() ∨ ¬QueryQueue.Empty() do
    while ¬QueryQueue.Empty() ∧ (ResultAvailable(QueryQueue.Front()) ∨
    PriorityQueue.Empty()) do
        node←QueryQueue.Dequeue();
        visiblePixels←GetOcclusionQueryResult(node);
        if visiblePixels> 0 then
            PullUpVisibility(node);
            $\underline{\mu \leftarrow visiblePixels/BBoxPixels(node)}$;
            $\underline{\text{node.bias} \leftarrow B_{s,t}(\mu)}$;
            $\underline{\text{VMSSE} \leftarrow \text{node.bias}*CalcSSE(node)}$;
            $\underline{\text{node.stopRefinement} \leftarrow \text{VMSSE} < \tau}$;
            Traverse(node);
    if ¬PriorityQueue.Empty() then
        node←PriorityQueue.Dequeue();
        if InsideViewFrustum(node) then
            stopMode←PredictRefinement(node);
            node.stopRefinement ← ¬(stopMode=Refine);
            node.visible ← false;
            node.lastVisited ← frameID;
            if node.stopRefinement then
                IssueOclussionQuery(node);
                QueryQueue.Enqueue(node);
            if ¬(stopMode=Delay) then
                Traverse(node);

Traverse(node);
if node.stopRefinement then
    Render(node);
else
    PriorityQueue.EnqueueChildren(node);
```

Fig. 2. Coherent HLOD Culling

- *Refine.* The children of the node are traversed immediately. No HOQ nor rendering is performed for the current node. If it turns out that the prediction was *too conservative* (it actually might have stopped for the current node) we end up rendering more geometry than necessary.
- *Stop.* An HOQ is issued and the node is rendered immediately. When the result of the query is available we compute the VMSEE of the current node. If the prediction was *too aggressive*, we have to continue by traversing the children of the node. Note that in this case we end up rendering geometry of (some) child nodes over the geometry of the parent node, which increases the rendering cost and can also lead to visual artifacts.
- *Delay.* In this case wait for the result of the query to decide on the refinement condition. Thus for a node which was delayed and for which refinement

should have stopped we have induced a latency in passing its geometry to the GPU.

From this analysis we designed a prediction technique which aims to minimize the number of incorrect predictions by assuming coherence of the front of termination nodes. It primarily aims to predict either *refine* or *stop* conditions with high accuracy. If we expect a stop condition, but with lower confidence, the predictor returns *delay*. We also return *delay* for nodes which have been previously invisible and thus we expect the refinement will terminate without rendering the geometry of these nodes. The main idea of the prediction is to estimate the VMSSE by combining the SSE of the current frame with the *cached* bias values from the previous frame:

$$VMSSE_i^{est} = SSE_i * bias_{i-1} \tag{2}$$

The prediction works as follows (see also the pseudocode in Figure 3):

- *Node was invisible in the previous frame.* In this case the prediction returns *delay.*
- *Refinement stopped for the node in the previous frame.* We calculate $VMSSE^{est}$, and if it is still below the threshold, the predictor returns *stop.* Otherwise, a significant change in the node visibility has occurred and the predictor returns *refine.*
- *The node was refined in the previous frame, but refinement stopped for all its child nodes.* In this case, the node is a good candidate for pulling up the termination front (see the light grey node in Figure 1-b). We verify this by first checking whether refinement for all children would still stop in the current frame based on their estimations $VMSSE^{est}$. If any of these indicates continue refinement, then the predictor returns *refine.* Otherwise, since the node itself doesn't have a cached bias value, we approximate $bias_{i-1}$ for the node by taking the average cached bias from all children. If the resulting $VMSSE^{est}$ is above the threshold, the predictor returns *refine.* Otherwise the predictor returns *delay.*

5 Results and Discussion

We implemented the proposed algorithm using C++ and OpenGL under Linux. The HLODs use an octree with a single discrete LOD per node consisting of about 2000 triangles. We used the quadric error metric [8,5] to construct the HLODs and to derive the model space errors. For efficient caching of the geometry on the GPU, we employed vertex buffer objects. The measurements were performed on two scenes with different depth complexities. For all tests we used a resolution of 640*480 pixels and the error threshold $\tau = 1$. The tests were evaluated on a PC with Intel-Core 2 Duo (2.4GHz) and nVidia GeForce 8800 GTX.

```
if ¬(node.lastVisited=frameID-1) ∨ node.visible=false then
    return Delay;
if ¬node.stopRefinement then
    candidateToShiftUp ← true;
    forall child ∈ node.children do
        if ¬child.stopRefinement ∨ ¬(child.bias*$CalcSSE$(child) ≤ $\tau$) then
            candidateToShiftUp ← false;
    if candidateToShiftUp then
        if AvgBias(node.children)*$CalcSSE$(node) ≤ $\tau$ then
            return Delay;
else if node.bias*$CalcSSE$(node) ≤ $\tau$ then
    return Stop;
return Refine;
```

Fig. 3. Function $PredictRefinement(node)$

5.1 Tests

We have used two scenes with middle and high depth complexities, respectively named as *scene 1* and *scene 2* (see Figure 5). For each scene we have designed a session representing typical inspection tasks. Depending on the traversal algorithm and the metric used to refine the hierarchy, we have evaluated the following scenarios:

- *Bool*: the hierarchy was traversed with the coherent culling algorithm version of Gobetti *et al.,* [6] (see Section 2). For this method we used the SSE metric for HLOD refinement, i.e., HOQs were used as if their result were boolean. Note that this configuration gives the ideal image quality for our tests.
- $SW(B)$: the hierarchy was traversed with the hierarchical *stop-and-wait* method referenced in Bittner *et al.* [3], using VMSSE to refine the hierarchy. (B) stands for using the $B_{s,t}(\mu)$ bias to compute VMSSE. Note that this configuration gives the ideal number of nodes to be drawn when using VMSSEs.
- $Simp(B)$: the hierarchy was traversed with the coherent HLOD culling algorithm proposed in [12] using VMSSE to refine the hierarchy.
- $Coh(B)$: the hierarchy was traversed with our new coherent HLOD culling algorithm (see Section 4) using VMSSE to refine the hierarchy.

5.2 Speedup

Figure 4 shows the whole sequence of drawn nodes together with the frame rates for the two scenes. All scene statistics have been summarized in Table 1.

It can be seen that our new prediction algorithm significantly reduces the number of drawn nodes compared to both *Bool* and $Simp(B)$, which also translates directly into higher framerates. The main reason for the reduction of drawn nodes

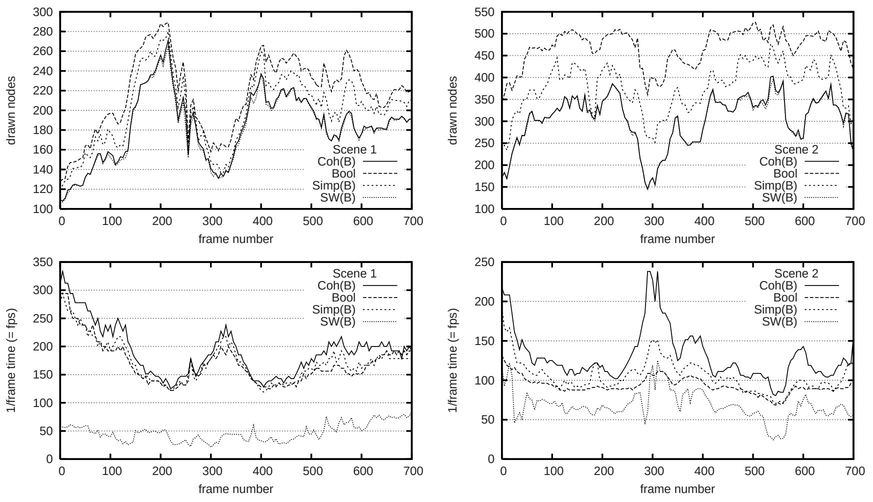


Fig. 4. Drawn nodes and frame rates for the test scenes

with respect to *Bool* was the use of VMSSE instead of SSE within refinement conditions. The speedup obtained by using the VMSSE is clearer perceived in scenes with higher depth complexity in which the savings in number of drawn nodes are greater. The reduction of number of drawn nodes with respect to $Simp(B)$ follows from the tighter approximation of the ideal number of nodes to be drawn, which relies on the method to approximate the VMSSE *bias*. The comparison to $SW(B)$ shows that our new approach is within 1% from the ideal number of nodes to be drawn, whereas the $Simp(B)$ method draws up to 25% more nodes. It is also worth noting that the precision of $Simp(B)$ depends on the scene depth complexity: it behaves poorly in scenes with higher depth complexities (scene 2), whereas the new method handles this types of scenes very well.

Table 1. Statistics for the test scenes. DN is the number of drawn nodes, RARN is the number of nodes that once rendered in a given frame need further refinement within the same frame and D is the number of nodes delayed for rendering. FPS is the number of frames per second, and Speedup is the relative speedup of $Coh(B)$ with respect to the given method. All presented values are averages over all frames.

Stats	*scene 1* *full resolution model* $\approx$ *5M* $\triangle'$*s* *number of HLOD nodes* $\approx$ *12k*					*scene 2* *full resolution model* $\approx$ *12M* $\triangle'$*s* *number of HLOD nodes* $\approx$ *21k*				
Scenario	DN	RARN	D	FPS	Speedup	DN	RARN	D	FPS	Speedup
Bool	217	-	-	169.9	1.13	468.5	-	-	92.9	1.40
$SW(B)$	181.4	-	-	47.3	4.04	302.1	-	-	68.2	1.91
$Simp(B)$	200	5	-	173.6	1.10	377.8	4.3	-	105.4	1.23
$Coh(B)$	183.3	0.1	4.7	191.6	-	303.9	0.4	4.2	130.2	-

The only source for visual artifacts inherent in the traversal algorithm (as opposed to the VMSSE calculation) is the case when there are some nodes that need to be refined even though they have already been rendered in the same frame (RARN). Fortunately, unlike for $Simp(B)$, for $Coh(B)$ we have found this value to be always negligible. The reason is that our delay strategy for the nodes where the stop refinement condition is predicted to be shifted up effectively minimizes RARN. Additionally, the RARN reduction is achieved without hindering performance: the average number of nodes that are delayed for rendering out of the total number of drawn nodes for the two scenes are only 2.56% and 1.39%, respectively.

5.3 Image Quality

We have measured the difference between the final ideal image obtained by $Bool$ and the one obtained by $Coh(B)$ by randomly selecting 20 frames of each inspection sequence and computing the peak signal-to-noise ratio difference ($psnr$). This measure has been traditionally used as an estimator of the distortion introduced by compression algorithms [14] and corresponds to the ratio of the power of a particular signal and the corrupting noise. The average and standard deviation $psnr$ values (luminance (l) and chrominance (C_b and C_r) components of the colors, repectively) for the 20 frames are: $l = 42.84 \pm 3.22$, $C_b = 67.04 \pm 3.68$ and $C_r = 56.2 \pm 3.65$ for scene 1; and $l = 35.51 \pm 1.4$, $C_b = 59.34 \pm 1.55$ and $C_r = 48.69 \pm 1.58$ for scene 2. The fact that $psnr > 30$ for all color components indicates that the proposed method practically does not alter the final image quality [14].

To emphasize the influence on image quality caused by $Coh(B)$, for each node in the front we have colored the geometry from dark to grey to light according to the severity level of the modulation introduced by the bias: dark represents regions of the model where the modulation is weak, grey represents regions where the modulation is moderate and light represents regions where the modulation is strong (see Figure 5). Note that the use of $Coh(B)$ attenuates the bias, i.e., the stronger the bias is, the less likely it is that the node is actually visible (see the last column in Figure 5). On the other hand, whilst in $Simp(B)$ the appearance of visual artifacts is common, in $Coh(B)$ we practically eliminated this problem (see the first two columns in Figure 5 and also the accompanying video).

5.4 Summary of Results

The results show that the proposed method is superior with respect to the previous state-of-the-art methods both in the framerate as well as in the image quality obtained. In comparison to the method of Gobetti et al. [6] ($Bool$), the reference solution for image quality measurements, we obtain a speedup of $1.13 - 1.4$, which is significant, while the visual quality of our method does not incur a perceivable penalty. In comparison to the method of Charalambos [7] ($Simp(B)$), the speedup is about $1.1 - 1.2$. However, that technique shows frequent visual artifacts which might not be acceptable in walkthrough or inspection

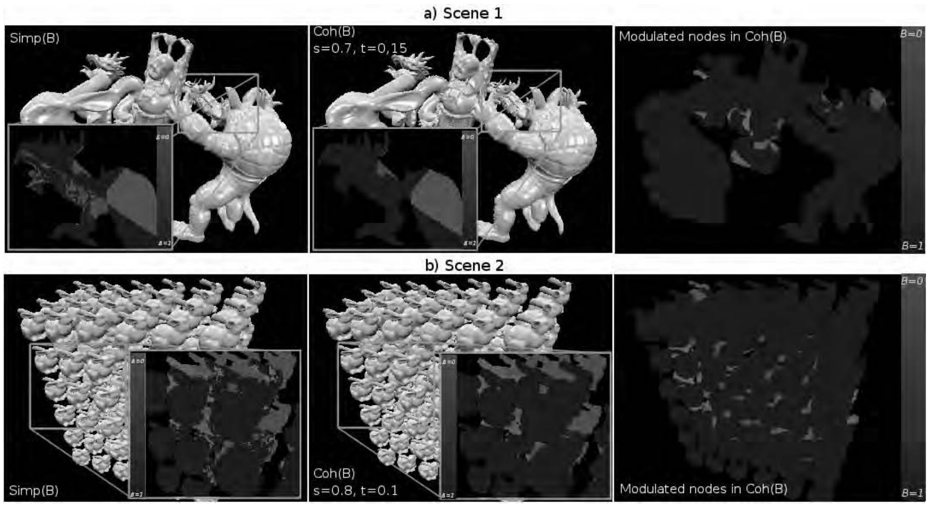


Fig. 5. Test scenes: selected frame of the visualization sequences when using $Simp(B)$ and $Coh(B)$. The last column corresponds to a visualization (from the user viewpoint) of the introduced modulation of the nodes selected to be drawn due to the VMSSE *bias* ($B_{s,t}(\mu)$) in $Coh(B)$. The small frames in the first two columns correspond to a detail in the scenes to show the possible appearance of visual artifacts due to RARN. Models courtesy of Standford Graphics Group.

applications, and which the new method avoids. Therefore, the proposed solution is qualitatively superior while still managing to be faster.

6 Conclusions

We have presented an algorithm to integrate HLOD and occlusion culling. The main contribution is that the algorithm closely approaches the optimal set of primitives to render while avoiding visual artifacts exhibited by previous methods. We demonstrate significantly improved performance when compared to previous approaches. The main idea is to exploit temporal coherence and make use of the visibility information returned by hardware occlusion queries to determine the simplification degree of nodes. The method also has a straightforward implementation.

Acknowledgments

This work has been supported by the following resources: an Ernst Mach grant from the Austrian Exchange Service – Agency for International Cooperation in Education and Research (ÖAD), Academic Cooperation and Mobility Unit (ACM); the EU project no. IST-2-004363 (GameTools); and by the Ministry of

Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics).

References

1. Guthe, M., Borodin, P., Balázs, Á., Klein, R.: Real-time appearance preserving out-of-core rendering with shadows. In: Rendering Techniques, pp. 69–80 (2004)
2. Cohen-Or, D., Chrysanthou, Y., Silva, C.T., Durand, F.: A survey of visibility for walkthrough applications. IEEE Transaction on Visualization and Computer Graphics (2002)
3. Bittner, J., Wimmer, M., Piringer, H., Purgathofer, W.: Coherent hierarchical culling: Hardware occlusion queries made useful. Computer Graphics Forum 23, 615–624 (2004)
4. Guthe, M., Balázs, Á., Klein, R.: Near optimal hierarchical culling: Performance driven use of hardware occlusion queries. In: Akenine-Möller, T., Heidrich, W. (eds.) Eurographics Symp. on Rendering 2006, The Eurographics Association (2006)
5. Yoon, S.-E., Salomon, B., Gayle, R., Manocha, D.: Quick-vdr: Interactive view-dependent rendering of massive models. In: VIS 2004. Conference Proc., Washington, DC, USA, pp. 131–138. IEEE Computer Society Press, Los Alamitos (2004)
6. Gobbetti, E., Marton, F.: Far Voxels – a multiresolution framework for interactive rendering of huge complex 3d models on commodity graphics platforms. ACM Trans. on Graphics 24, 878–885 (2005)
7. Charalambos, J.P.: Virtual multiresolution screen space errors: Hierarchical level-of-detail (hlod) refinement through hardware occlusion queries. In: GMAI, pp. 221–227. IEEE Computer Society Press, Los Alamitos (2006)
8. Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., Scopigno, R.: Adaptive TetraPuzzles – efficient out-of-core construction and visualization of gigantic polygonal models. ACM Trans. on Graphics 23(3), Proc. SIGGRAPH 2004 (2004)
9. Guthe, M., Borodin, P., Klein, R.: Efficient view-dependent out-of-core visualization. In: VRAI 2003. The 4th International Conference on Virtual Reality and its Application in Industry (2003)
10. Andújar, C., Saona-Vázquez, C., Navazo, I., Brunet, P.: Integrating occlusion culling with levels of detail through hardly-visible sets. In: Computer Graphics Forum (Proceedings of Eurographics 2000), pp. 499–506 (2000)
11. El-Sana, J., Sokolovsky, N., Silva, C.T.: Integrating occlusion culling with view-dependent rendering. In: VIS 2001. Conference Proc., pp. 371–378 (2001)
12. Charalambos, J.P.: Coherent hierarchical level-of-detail (hlod) refinement through hardware occlusion queries. In: SIACG 2006. Ibero-American Symp. on Computer Graphics, Univ. Santiago de Compostela, Eurographics Association, Spain (2006)
13. Anonymous: A visibility information-based metric for performing the refinement of a hlod-based system. Submitted to Computer Graphics Forum (2007)
14. Gonzalez, W.: Digital Image Processing, 2nd edn. Prentice-Hall, Englewood Cliffs (2001)

Autopolis: Allowing User Influence in the Automatic Creation of Realistic Cities

Soon Tee Teoh

Department of Computer Science, San Jose State University

Abstract. Given terrain information and user preferences such as desired city size, Autopolis generates commercial and industrial centers, and grows areas surrounding these centers. The program also creates streets and highways, and assigns a land-use for each parcel. Compared with previous methods, Autopolis generates cities with better realism, greater detail in layout, and more design variety.

1 Introduction

The computer gaming industry also has a strong demand for fast automatic creation of urban enviroments because games across different genres (such as driving, first-person shooter, role-playing, and strategy) take place in virtual city environments. Computer-generated cities are also used in many movies.

We present Autopolis, a system for the automatic creation of realistic cities. Given terrain information, the user first sets initial parameters such as the number of commercial and industrial centers, airports and seaports desired. The program generates these centers according to heuristics. The heuristics for the generation of centers and commercial and residential regions around the centers are loosely based on classical well-established urban development models such as Burgess' concentric zone model, Hoyt's sector model and Harris and Ullman's multiple nuclei model. These models are designed to be simplistic generalizations of urban growth patterns. Autopolis includes more advanced heuristics that incorporate more variables (such as terrain, center proximity, water and randomness) to better mimic real-world cities. For example, airports tend to be near to the coast, and industrial centers tend to be near to seaports, on flat land, and are a moderate distance from commercial centers.

Next, Autopolis grows areas surrounding these centers. These areas are set to one of the following primary land-uses: commercial, industrial, residential or park. The growth of these areas also depend on their target sizes set by the user. Here, the user can determine the desired size of the city and relative sizes of the industrial, commercial and residential areas. The program grows areas realistically, for example, residential areas tend to grow on the perimeter of commercial areas, and urban areas have a preference to grow on gentle terrain.

Autopolis then creates streets and highways, and assigns more detailed secondary land-uses. The result is a city that is realistic in layout, having shapes and patterns that are similar to real-world cities. By controlling parameters and also editting the output of the program at each step in the process, the user can also influence the appearance and design of the city. The program provides the full range of user control, from fully-automated city creation, to fully-manual city creation.

Compared to previous methods, Autopolis generates cities realism and detail, including highways, major and minor streets, and more detailed classification of land-use, and takes into consideration more heuristics and factors that influence city development and land-use. Autopolis also creates cities with more variety in appearance. Example 3D views of automatically-created cities are shown in Figures 1 and 2.

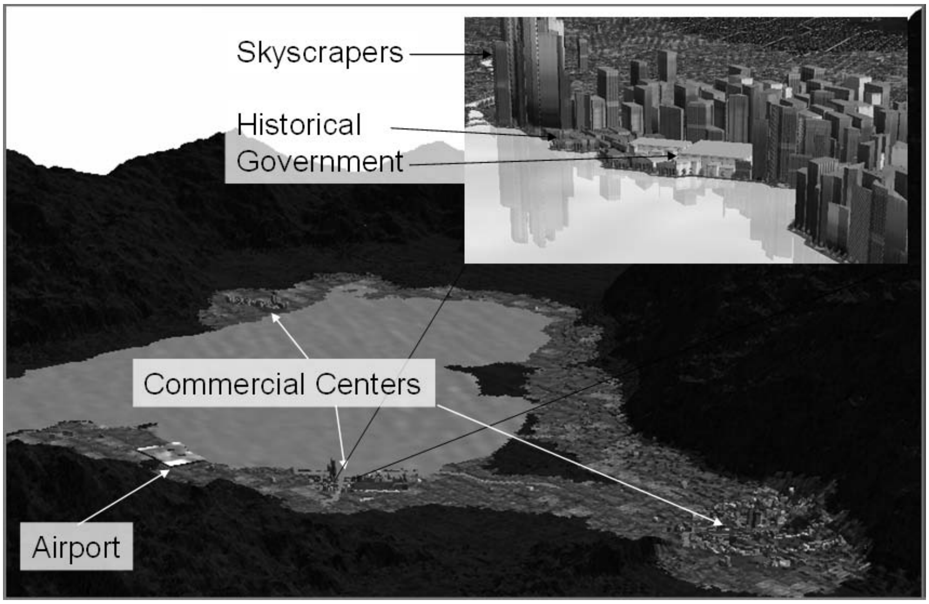


Fig. 1. Annotated bird's eye view of an automatically-created city

2 Related Works

Greuter et al. [1] have presented a procedural method to generate "pseudo-infinite" virtual cities in real-time. Their method focuses on generating buildings with different appearances. By generating a large number of these buildings in real-time, their method creates an entire scene full of office buildings. As the user pans across and looks at a different part of the scene, the method simply creates more random buildings in real time. This method differs from our work in that it only generates office buildings and does not generate realistic cities containing other common land-use types, such as residential and industrial areas.

Lechner et al. [2] proposes a method that uses agents to grow a city. Like our system, their system accepts terrain height and water as input and outputs

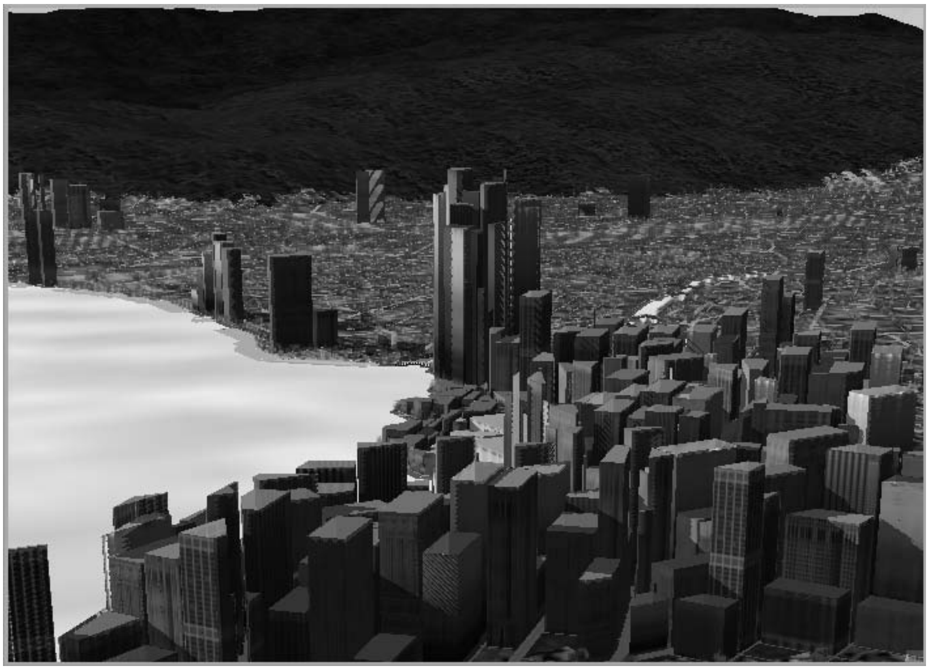


Fig. 2. Bird's eye views of an automatically-generated city

land use. They classify land use into three types: commercial, residential and industrial. Different types have different preferences, for example, residential developers prefer regions where roads are less congested, are close to water, and are far away from industrial areas. Our proposed process is able to generate cities that have much higher levels of complexity and detail, and therefore mimic real-world cities better. We also allow the user a lot more control over the parameters of the city design, so that the program can generate many different styles of cities.

Parish and Mueller's [3] system, called CityEngine, uses L-systems to generate streets and buildings. This method focuses on generating street patterns and does not determine the land-use type of different areas of the city. Sun et al. [4] also created a virtual road network. Their approach builds road patterns based on image-derived templates and rule-based systems. They accept an input map of the population density of the area, and then intelligently produce streets of different patterns based on geographical and urban restrictions, such as elevation and congestion.

DiLorenzo et al. [5]'s approach creates a hierarchical model based on zoning and other heuristics. These heuristics are based on sociological and empirical information from real-world cities. For example, the density is the highest in the Central Business District and decreases with distance from the center. Their system then runs a simulation of the city growth/decay over time. Our method considers additional factors in the development of a city.

3 Automated Centers

Given terrain information, the positions of commercial centers, seaports, airports and industrial centers can be automatically-generated by Autopolis. Terrain information consists a 2D matrix of the height map (height of each point on grid given) and water map (each point on grid set to Water or Land).

As a pre-processing step, for each cell with low elevation (with threshold defined by the user), an attribute value *distanceToElevated* is set, that indicates its distance to elevated land. Attributes *distanceToRiverMouth* and *distanceToWater* are also set. Below is the algorithm to set distance to elevated land. The algorithms to set distance to river mouth and water are similar.
Program to Set Distance from Elevated Land

```
program SetLowLand
  current_distance = 1;
  for each cell C in grid
    if (C is low and adjacent to elevated land)
      add C onto queue;
      C.distanceToElevated = current_distance;
  while not empty(queue)
    current_distance++;
    for each cell C from queue
      for each adjacent cell C2
        if not(C2.done)
          add C2 to queue;
          C2.distanceToElevated = current_distance;
end.
```

(Modified flood-fill algorithm)

First, Commercial Centers (also commonly known as "Downtowns," "Financial Districts," or "Central Business Districts") are generated. Random positions are generated for each Commercial Center. The position is rejected if it is closer than d from an existing center, where d is a threshold value set by the user. The position also has a probability p of being rejected if it is at a high elevation, where $p = h \times m$, where m is a parameter set by the user, a greater value of m is used to indicate preference for lower elevation, and h is the elevation of the cell ($0.0 < h < 1.0$, $0.0 < m < 1.0$).

Next, seaports are generated according to the following algorithm. First, all coastal positions are enumerated. Next, N random coastal positions are generated, and each position is given a score s. $s = k_1 \times d_1$ - $k_2 \times d_2$ where d_1 is the distance to elevated land and d_2 is the distance to a river mouth. d_1 and d_2 have been pre-computed for all cells in O(n) time (algorithm given above). k_1 and k_2 are parameters set by the user, a larger k_1 value is used to indicate preference for low land, and a larger k_2 value is used to indicate preference against proximity to river mouths, because river mouths have high level of sedimentation unsuitable for ports. The position is rejected if it is too close to an existing center. Among the N random positions, the one with the highest score is selected as the seaport.

Airports are generated similarly to seaports. The only difference is in the assignment of scores. To simulate real-world siting of airports, a position has a higher score if it is surrounded by a large area of flat land (for clear flight path) and if it is near to a lake or the sea (for easy future extension of additional runways).

Industrial centers are generated last because their positions depend on the positions of existing airports and seaports. Industrial centers are generated in a similar way to seaports and airports. The score of a position is higher if it is surrounded by a large area of flat land, and also if it is close to a seaport and an airport.

Figure 3 shows the positions of the commercial centers, industrial centers, airports and seaports generated in a given terrain. After the centers have been generated, they are displayed on the map. At this stage, the user has the option to intervene by selecting and moving any center to any desired position on the map.

4 Automated Roads

Roads are automatically generated in several different ways. First, highways can be created to connect secondary urban centers to the main city center. Then highways can also be created to connect centers with one another. Next, main roads are created, and finally, smaller roads are created.

Highways are created from the town centers, industrial centers, airports and seaports to the city center. A shortest path algorithm is used to create each road to the city center. Each tile on the map is a node. There are costs associated with going to a tile with a different elevation, and also going across water. These costs can be set by the user.

The shortest path algorithm is slightly modified to take advantage of existing roads (that is, the road that have already been created from other secondary centers to the city center). In such a case, when the shortest path search reaches a tile that is an existing road, it is assigned a cost, such that the cost of taking this existing road to the city is a fraction of its actual distance to the city. This fraction can be set by the user. The algorithm continues until the shortest path search finds the city center. Then, among all the paths that reach an existing road, or directly reaches the city center, the path that has the smallest cost is chosen.

Commercial land-use is also randomly generated along the highways to simulate real-life growth of settlements along transport routes. The frequency and size of these small commercial areas can also be chosen by the user. In Figure 3, you can see the highways that have been generated to connect the secondary centers with the main commercial center in the middle of the picture.

Next, main roads are created. To create a grid of roads, the user first defines the resolution of the grid in the x and y directions. The system then generates a road point in each of the grid cells that has more than a user-defined percentage of developed area. There are two methods of determining the position of this

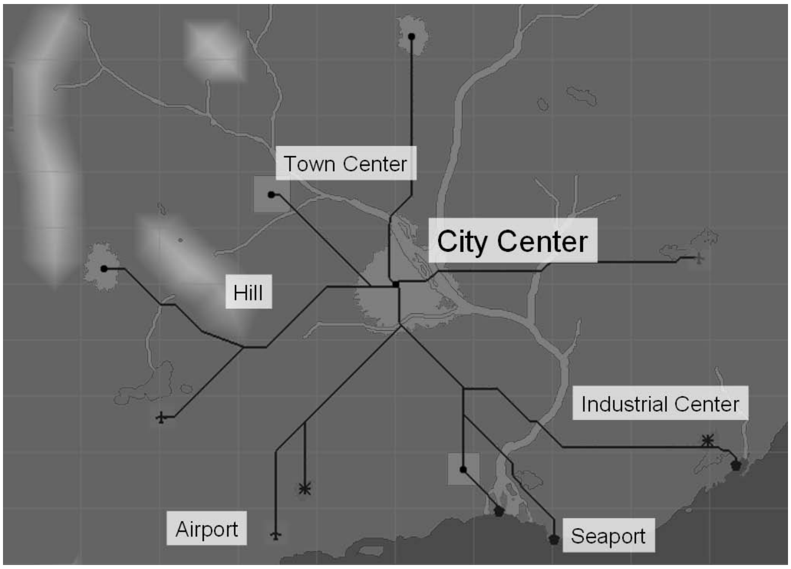


Fig. 3. Automatic urban centers, land-use areas and highways. Path-finding algorithm avoids hills and water.

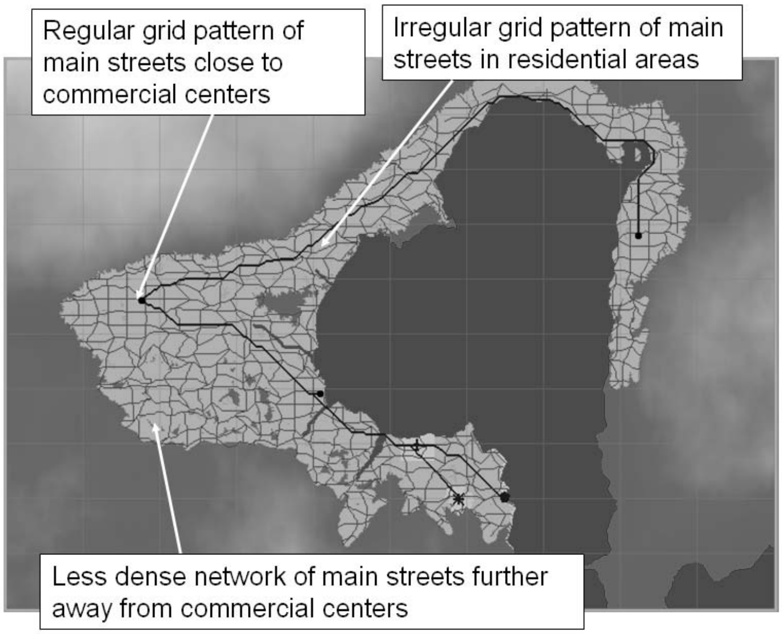

Regular grid pattern of main streets close to commercial centers
Irregular grid pattern of main streets in residential areas
Less dense network of main streets further away from commercial centers

Fig. 4. Hybrid method for main streets: regular hexagonal grid used near commercial centers, irregular patterns used elsewhere. Less dense network further from city center. Many major world cities have such a pattern.

road point. In the “irregular” mode, a random developed point in each grid cell is chosen as the road point. In the “regular” mode, the center of each grid cell is selected as the grid point.

After all the grid points are determined, roads are created for each pair of adjacent grid cells containing grid points. The shortest path algorithm is used to create the roads. The creation of grid roads should take place after the developed areas have been created. Many major U.S. cities, such as Chicago, have such a pattern of main roads throughout the city.

Next, the vector field method is used to create minor roads. First, a vector field is created. This vector field can be created based on any combination of the following factors: (1) the coastline, (2) the relief, and (3) the existing network of main roads.

To create a vector field according to the coastline, the cells containing the coastline are assigned a vector in the direction parallel to the coastline. To use the relief to influence the vector field, a cell that contains difference in height is assigned a vector parallel to the contour in the cell (in other words, perpendicular to the direction of steepest descent). To use the existing network of main roads, each cell containing a main road is assigned a vector parallel to the direction of the main road. After these cells have been assigned vectors, the method grows the vector field by duplicating the vector for each cell to any adjacent cell that does not yet have a vector assigned. In this way, every cell is assigned a vector.

Then, for each region surrounded by main streets, we find the vector at the center of the region. We then create parallel minor streets inside this region in the direction of the vector and also in the perpendicular direction. The streets are equal distance apart, and those in the perpendicular direction are a greater distance apart. An example is shown in Figure 5. The user is allowed to change the roads created by the automatic algorithms.

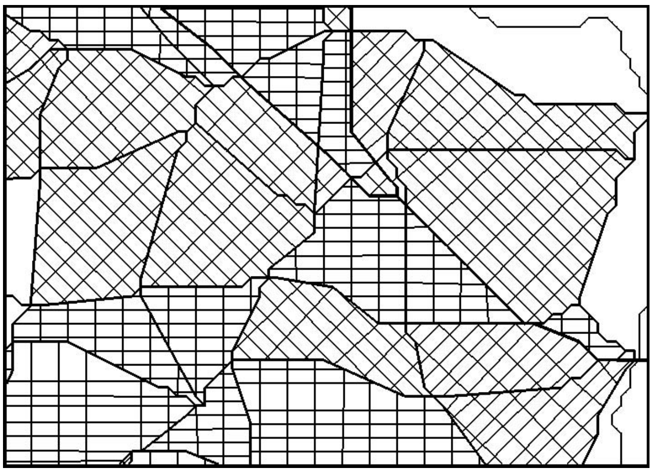


Fig. 5. A grid of minor streets is created in each region bounded by main streets

5 Growing Areas

Urban areas initially grow from the urban centers: Commercial centers, airport centers, seaport centers and industrial centers. We use various algorithms to grow these urban areas. Figure 3 shows an example. The user is allowed to set target sizes for each of the land-use types: commercial, industrial and residential. For each of the urban centers, the user is also allowed to set its target sizes. By setting different relative target sizes, the user can control the style of the resulting city. For example, increasing the size of the commercial and industrial areas will result in a more densely-packed city.

The square method assigns a square area surrounding the urban center to have the urban landuse. The size of the square is determined by the user. For example, if the user has chosen to use the square method for an airport center, and set the square size to 10 miles, then, the land-use of the 10x10 mile square surrounding the airport center will be set to *airport*. This simulates many real-life situations where city-planners have designated rectilinear areas for commercial land-use. For example, downtown San Jose, the area designated for commercial use, has a square shape.

Alternatively, the randomized method is used. he user sets a target number of cells to be set to the urban land-use. Urban areas tend to grown along the coast, and also they tend to avoid hills. Therefore, the user is also allowed to set two parameters: (1) Water Affinity W, and (2) Hill Affinity H, to control how the growth of this urban area is influenced by the presence of water and terrain. A higher value for W leads to a higher probability that the cell would be selected by the random algorithm, while a larger value for H causes the cell to be less likely to be selected. A modified randomized flood-fill algoritm taking into consideration W and H is used to grow urban regions.
Program to Grow Urban Region

```
nCells := 0;
program SetUrbanLand(Target)
  put Seed in queue
  while (nCells < Target)
    for each cell C in queue
      for each cell C2 adjacent to C
        if (C2.landuse is not Urban)
          if (rand()%(5-Wd+Hh)==0)
            C2.landuse := Urban;
            add C2 to queue;
            nCells := nCells + 1;
end.
```

(where d is distance from water, and h is difference in height from neighbors)

After the commercial and industrial areas have been set up, residential and park areas are grown from the boundaries of the commercial and industrial areas. All undeveloped cells adjacent to a commercial or industrial area are randomly set to have residential or park land-use and place in a queue and the areas are grown using the same algorithm to grow an urban area given above.

After the various land-use areas have been set by the program according to the parameters set by the user, the user has a chance to change the land-use simply by painting over the area.

6 Secondary Commercial Land-Use

Each primary land-use type (commercial, residential and industrial) can be further classified into secondary types. The secondary land-use types that influence the city skyline the most are the secondary commercial land-use types. Our program currently allows four different sub-types for commercial land-use: (1) regular commercial, (2) historical, (3) government, and (4) skyscraper. These are the four most common land-use types found within the downtown business district of major cities in the world. They are classified in this way because these different types all have a distinct style of buildings that affect the appearance of the city.

The program begins by letting the user choose the number of Skycraper, Historical and Government sub-centers for each commercial center. The program then generates these centers based on heuristics explained below. For example, government centers tend to be near the city center, and historical centers tend to be nearby government centers. The user is then allowed to adjust the positions of these sub-centers. After that, the program generates the city by growing these secondary areas from the sub-centers. An example map showing secondary commercial land-uses is shown in Figure 6.

Many major city centers in the world have a substantial amount of land used for government buildings. Especially in major capitals like London, Beijing and Washington D.C., government buildings have a significant impact on the appearance of the city. Government administrative buildings also tend to look different from other buildings because they tend to be shorter, and are often built with an ornamental classic design, using traditional architecture, even though they may have been built only recently. Streets in the government areas tend to be grand, and are often long and straight, designed to be suitable for holding parades, marches. Examples include the National Mall in Washington D.C. Large squares, such as Tiananmen Square in Beijing and the Red Square in Moscow are also commonly present in government areas. Large parcels of land are often dedicated to palatial government buildings. Capital cities built from scratch, including Canberra (Australia), Brasilia (Brazil), Putrajaya (Malaysia) and Washington D.C. (U.S.A.), often have grand avenues and a spokes street pattern connecting major government buildings and circles or squares containing monuments and statues, like Paris (France).

Autopolis assigns a random position to be the Government Center. In each urban commercial center, all cells are possibly chosen, however, more weight is given to cells that are near the city center. This is because historically, government centers tend to be near to the city center itself.

Once the government center is set, the user is given the opportunity to move it elsewhere. After this, a "government pattern" of streets is built centered on this government center. An example government pattern of streets built around a government center is shown in Figure 6. All the streets created in this pattern will be flagged as "government" streets, and all adjacent parcels are set to Government land-use.

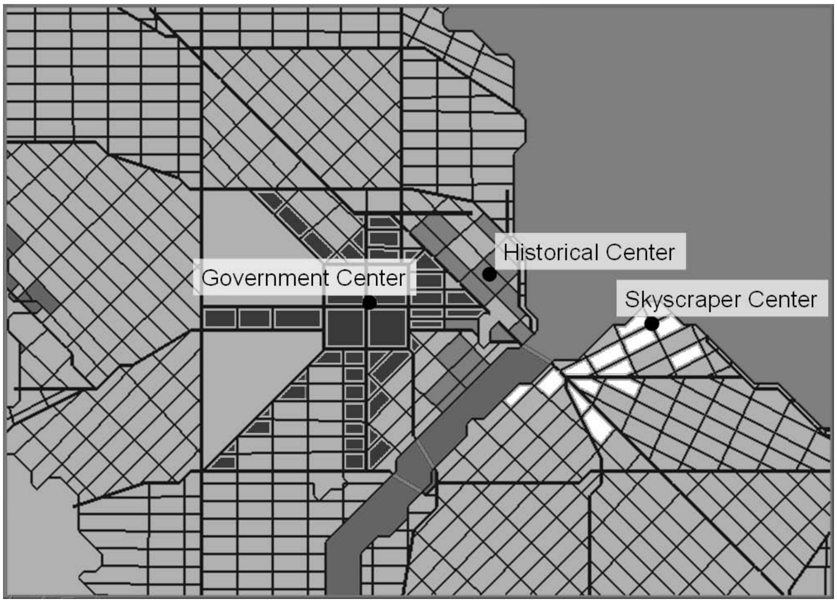


Fig. 6. Secondary Commercial areas grown from their respective centers

Historical areas are important because they have a distinctive appearance, and are often low in height compared to the skyscrapers in the financial district. The historical center is often rather near to the government center, because both were established when the settlement was still small. It also tends to be near water, because that is where the port is usually originally built. As usual, any cell in the commercial area is possible to be chosen as the historical center, but more weight is given to those cells that match the heuristics mentioned.

Many commercial centers have a section that has especially tall buildings. Examples include the La Defense business district in Paris and the Pudong financial district in Shanghai. This cluster of skyscrapers often define the skyline of the city. Most cities have one single skyscraper area, while some have multiple. For example, New York City has a cluster of skyscrapers in Downtown and another cluster in Midtown. Many skyscraper areas are in the middle of the city, growing near to the historical center of the city, while other cities preserve their historical centers and instead open up an entire new swath of previously undeveloped land to build a new financial center.

Also, the size of a skyscraper area compared to the total size of the city can vary widely. American cities tend to have a small concentrated downtown area with skyscrapers, surrounded by large low-lying suburbs. Other cities such as Sao Paolo (Brazil) appear to be entirely made of skyscrapers. Some cities also do not have skyscraper areas at all, because the building codes of the city impose a height limit of the buildings and therefore skyscrapers are legally prohibited from being constructed.

Therefore, the user is allowed to choose some parameters to control the way skyscraper areas are generated: (1) number of skyscaper areas for each commercial area, (2) the target size of a skyscraper area, and (3) the desired proximity of the skyscraper area(s) to the historical center. By choosing different parameters, the user can cause the program to create cities of different characteristics.

Given these parameters, the algorithm generates some suggested skyscraper centers. The user can move these centers. When the user is satisfied, the algorithm proceeds to assign parcels in these centers to be of the "skyscraper" secondary commercial land-use. The algorithm then grows these skyscraper areas by adding adjacent parcels to the skyscraper land-use until the target is reached.

7 Secondary Residential Land-Use

Residential areas in a city contain different types of buildings besides houses, including schools, shopping centers, libraries, high-rise apartment blocks and

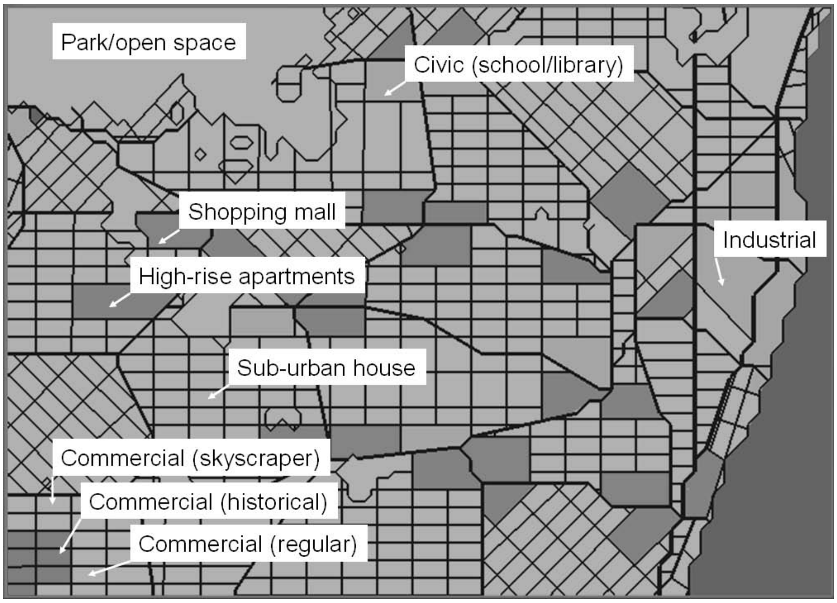


Fig. 7. Parcels around major intersections merged to form secondary Residential areas

hospitals. Most of the above buildings are usually situated along main streets, and shopping centers are often found at intersections of main streets. Autopolis merges parcels at intersections of main streets into larger parcels and assigns them to the Shopping Mall or other secondary residential land-uses. An example is shown in Figure 7.

8 Conclusion

Autopolis has several very attractive features that makes it effective for practical use. Compared to previous methods, Autopolis is able to quickly generate cities of unprecedented realism and detail, including highways, major and minor streets, and more detailed classification of land-use (skyscraper, historical, government, residential, civic, ports, industrial etc.). Compared to previous methods, Autopolis also takes into consideration more factors that influence city development, such as water proximity, slope, elevation, and optimal distance between industrial/commercial centers and ports.

Autopolis also allows the user a full range of controls, from a fully-automated city generation, to a fully manual city design. Because the process generates data randomly, the user is able to generate a variety of different cities using different seeds, or different parameters such as different number and size of commercial centers.

References

1. Greuter, S., Parker, J., Stewart, N., Leach, G.: Real-time procedural generation of 'pseudo infinite' cities. In: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, pp. 87–95 (2003)
2. Lechner, T., Watson, B., Wilensky, U., Felsen, M.: Procedural city modeling. In: Proceedings of the 1st Midwestern Graphics Conference (2003)
3. Parish, Y., Muller, P.: Procedural modeling of cities. In: Proceedings of ACM SIGGRAPH 2001, pp. 301–308. ACM Press, New York (2001)
4. Sun, J., Yu, X., Baciu, G., Green, M.: Template-based generation of road networks for virtual city modeling. In: VRST 2002. Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 33–40. ACM Press, New York (2002)
5. DiLorenzo, P., Zordan, V., Tran, D.: Interactive animation of cities over time. In: CASA 2004. Proceedings of the 17th International Conference on Computer Animation and Social Agents (2004)

Simulation of Flexible Tubes in VR

Florian Mannuß[1], André Hinkenjann[1], Gernot Göbbels[2], and Martin Göbel[2]

[1] *University of Applied Sciences Bonn-Rhein-Sieg*, Sankt Augustin, Germany
{Andre.Hinkenjann,Florian.Mannuss}@fh-brs.de
[2] *fleXilution GmbH*, Cologne, Germany
{Gernot.Goebbels,Martin.Goebel}@flexilution.de

Abstract. We present a Virtual Reality application enabling interactive, physically correct simulation of tube-like flexible objects. Our objective was to describe flexible objects by a set of parameters (length, diameter and material constants) instead of rigid geometry (triangle meshes) and to give the user the possibility to add, delete and manipulate those flexible objects in a stereo projected environment in real-time.

1 Introduction

CAD-Tools create flexible, tube-like objects as static geometries during the construction process, but the layout and behavior of these objects is not necessarily physically correct. A more favorable way is to describe all flexible objects using a predefined set of parameters and let a simulation return a correct triangle mesh. Complete cable-trees can be constructed by joining cables or single cables can be fixed at a cable tie. A positive result of such a VR simulation system is a reduction of development time, eg. in automotive industry, the domain of our application.

A simulation engine for tube-like objects must provide real-time frame rates of at least 15 fps. The simulation must coincide with the shape of the real tube within a small error. These two characteristics are fulfilled by the simulation package *fleXengine*. For details about the simulation software, see [1]. We use this software in our system without modifications.

2 Related Work

Two popular VR frameworks are VR Juggler [2] and AVANGO [3]. VR Juggler provides interfaces for a wide variety of graphical interfaces, but applications are bound to the chosen graphical interface. When the programmer needs to replace the renderer, the source code of the application has to be rewritten. The system can be extended through managers. User interaction is decoupled from the base system and is part of the application. AVANGO is very closely coupled with the OpenGL Performer library [4] and extends it with a VRML-like field interface. User interaction is again the task of the application programmer. Both systems do not know the concept of a generic "renderer" that works on scene data and produces some output (visual, auditory etc.). For the task presented here, a simulation renderer that works on tube data is needed. Our own framework allows the use of generic renderer plugins (cite ommited for review).

In [5], the authors mention the inclusion of the simulation of hoses and wires in their VR system without describing their integration approach. The flexEngine has recently been included into a commercial VE software package [6]. Unfortunately, the integration process is not published and cannot be compared at this stage.

Grégoire and Schömer [7] describe interactive simulation of flexible parts. Their paper however is focused on the simulation approach and the mathematical backround while this paper focuses more on the integration into a VR software. For an overview of other simulation approaches see their paper.

3 VR-Framework and Setup

The application was realized with our own framework. It contains a very small *kernel* with management functionality and abstract interfaces to *plugins* as seen in figure 1. All functionalities implemented in plugins are loaded at run time.

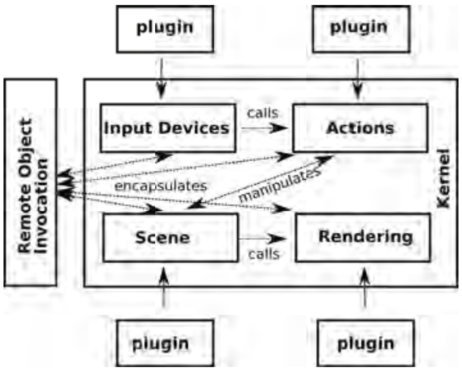


Fig. 1. Overview of the VR framework with plugin interfaces

Input device receives and preprocesses the data from a physical input device. An *action* is responsible for manipulating the virtual scene. The *scene* representation is separated from the *renderer*. Arbitrary renderers can be used for the same scene representation. A renderer is not limited to graphical output. It is a synonym for all tasks that work on the virtual scene (an audio renderer could process audio scene data, eg).

To achieve renderer independency, all scene data has to be encapsulated. Therefore, an abstract scene layer is needed through which all possible renderers can access the scene data. We call this abstraction layer *virtual objects* and *virtual object attributes*.

As a consequence of this design, we achieve independence of any third party scene graph, audio or physics library in the kernel.

Plugins are not a hard coded part of an application. All plugins that have to be loaded at start up time are specified and configured using a configuration file. In this configuration file virtual objects with their attributes can be specified, too.

The hardware setup of the system is shown in figure 2.

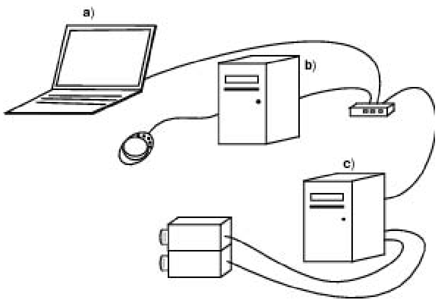


Fig. 2. Hardware setup for the application. **a**) is a tablet PC providing the user interface, **b**) is the input device server using a Space Mouse and **c**) is the application PC. Two projectors are connected to the appication PC to provide stereo output.

4 Tube Interaction and Parameter Extraction

In our application, a tube consists of tube parameters, like length, inner/outer diameter, minimal bending radius, material constants or torsion. In addition, the handles of the tube must be specified. Overall, there are three different types of handles as shown in figure 3: The handles marked with *(a)* are *normal handles*. If such a handle (except end-handles) is moved, the tube slides through the handle. A handle of type *(b)* is called a *split handle* where the tube is fixed at the handle. This handle is created by splitting the tube into two independent tubes at the split handle position and by joining the two newly created end-handles. *Join handles (c)* allow for the creation of cable trees. When moving such a handle it behaves like a normal handle. A splitted join handle behaves like a split handle when moving the handle.

In order to extend our VR framework for the simulation and interaction of flexible, tube-like objects, three new plugins had to be written (for the graphical representation and the rendering we used Open SceneGraph [8] based plugins).

fleXData (scene plugin) adds two new attributes for *single* parameters and for *handle* parameters. A *single* describes a tube object. This is the data the fleXRenderer needs.

fleXRenderer (renderer plugin) is the simulation component. Its output is a triangle mesh for every tube object in the scene. It is added as a triangle mesh attribute to the virtual objects. The simulation is done using the *fleXEngine* library.

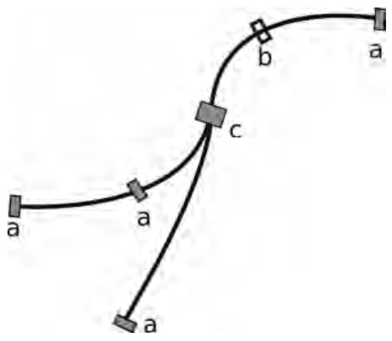


Fig. 3. Two tubes showing all possible handle types. **a** shows a normal handle, **b** a split handle and **c** a join handle.

FleXActions (action plugin) adds all actions needed to create, delete, modify and interact with the tube and handle objects.

Due to external constraints, all actions had to be usable with a one-button input device. The minimal requirements to the input device are to provide the position, orientation and one button state. Position data and orientation can either be absolute or relative. All actions are part of the *fleXActions* plugin and every activated action stays active until another action is activated.

We identified the following actions: *Create a single with handles; Add new handles at the beginning, end or in between an existing tube; Split/Unsplit a handle; Join/Disjoin; Delete a handle; Delete a single or handle.* Utility actions are: *Select single, handle or background.*

Create single produces a new single (tube object) and adds a new handle to the intersection point of the virtual input device representation and the background surface until the user clicks on the newly created single. After this intersection the creation process is finished and a new single can be created. In order to find the intersection point and the termination criteria the *select background* and *select single* actions are utilized.

The orientation of the handle is calculated using the surface normal $\boldsymbol{n}$ at the intersection point and the normalized vector $\boldsymbol{v}$ from the last handle position to the intersection point.

More complex actions are implemented as state-machines. The process of *adding one or more handles* at the front of a single is the following: First, the single has to be activated. On *button down* we check if a single was hit. If this single is still hit on *button up* we select this single, otherwise we jump to the final state. With this two phase mechanism a wrong selection process can be aborted.

Joining is done by selecting two handles. After the second handle was selected both handles are joined. The first selected handle is the master handle and the second is mapped onto it by maintaining its orientation.

Disjoin releases the joined handles of the selected handle. If more than two handles are joined all handles are released.

Every handle can be split except for the two end-handles of a single. When performing the *split* operation a new single will be created and all handles right of the split handle are transfered to the new single. For the split handle a new handle will be created. After that we have two singles and by joining the back-handle of the left and the front-handle of the right single the split handle is created. If the handle that has to be split is already a join-handle, then all joined singles have to be split and then joined again.

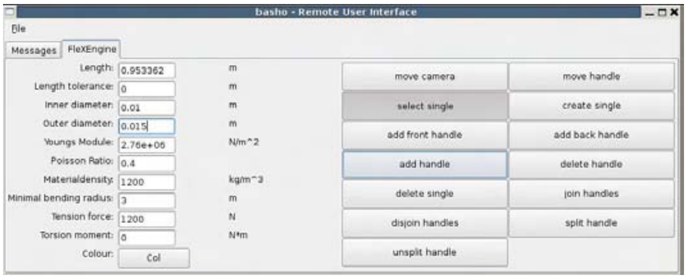
Unsplit merges two singles into one single. Again, if the split-handle is also joined we perform the unsplit operation for all joined handles that are splitted.

By using *delete single* all handles of a single and the single itself are deleted. If there is a split-handle, the connected single will be deleted either. Join-handles are not touched and only the handle of the single we want to delete is removed. When *deleting a handle* more work has to be done. The simple case is where we only delete a simple handle. No additional work has to be done, except if the single had only two handles left. In this case the whole single will be deleted. When deleting a split-handle, the two singles have to be joined again and in case of a join-handle all handles connected through this join have to be deleted, too.

Extraction of tube parameters
The initial input to our system is a CAD model of the tubes, consisting of triangulated surfaces. Unfortunately, the quality of the triangulation is sometimes poor and needs to be manipulated by hand to achieve good visualization and parameter extraction results. When extracting tube parameters three values can be obtained, the tube length, inner and outer diameter. All values must be extracted from the geometry. This leads to the constraint that every single must be represented by one geometric object. However, in practice sometimes more than one single is stored in one geometry node leading to the task to separate every single.

The algorithm to extract the tube parameters first finds the two end-points of the single. If the tube has only an outer diameter, all edges belonging only to one triangle are part of the border. For a single with an inner and outer diameter, if the surface normals satisfy $-\epsilon \leq \boldsymbol{n_1} \cdot \boldsymbol{n_2} \leq \epsilon$ with $\epsilon \approx 0.1$ and $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ being the normals of the triangles sharing an edge, the shared edge belongs to the border. After all border edges are determined, all connected edges must be grouped. Finding the inner/outer diameter is done by finding the two most distant vertices in those edge groups. For finding the length of the single a greedy algorithm is used that chooses the edge where $min(\boldsymbol{e_{act}} \cdot \boldsymbol{e_n})$ for all adjacent edges e_n of the actual edge e_{act}. In the next step the path vertices are projected along the surface normal of the actual vertex to the opposite side of the single. The mid-point of the line from the actual vertex to the projected vertex is one of points on the middle line of the single. As a last step the distances between all mid-points is summed. The result is the length of the single.

2D User Interface
Because of our goal to use all actions with one button a method to toggle between actions is needed. To provide the possibility to change the numerical values of a single we decided to create a 2D user interface (UI). Figure 4 shows a screen



Length:	0.953362	m
Length tolerance:	0	m
Inner diameter:	0.01	m
Outer diameter:	0.015	m
Youngs Module:	2.76e+06	N/m^2
Poisson Ratio:	0.4	
Materialdensity:	1200	kg/m^3
Minimal bending radius:	3	m
Tension force:	1200	N
Torsion moment:	0	N*m

Fig. 4. 2D user interface for changing single attributes and to activate an action

shot of the GUI. On the left side the attributes of a single are displayed and on the right all actions implemented for this application are shown as buttons. After a single has been selected its attributes are displayed in the 2D UI. Selecting a single can either be done through *select single* or *create single* and *add front/back/position*. Parameter changes are immediately executed on the single virtual object. By tapping an action button the action associated with this button is activated. The UI is implemented using the *remote object invocation* plugin to be able to run on a tablet PC.

5 Conclusion and Future Work

We presented a VR application for interactive cable and wire simulation. With this application the user is able to interact with flexible cables and hoses in real time. The system has been presented using a setup like that of photograph 5.

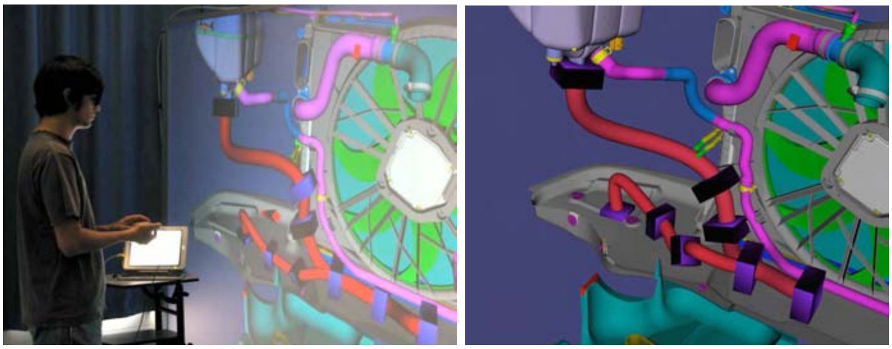


Fig. 5. Photo of the application installation. The red tubes are flexible objects. The user in front of the projection wall uses the tablet PC with the 2D user interface. Model courtesy DaimlerChrysler. To the right, the corresponding screen shot is shown.

Despite the use of stereoscopic output the localisation of objects can be improved. Very often it is hard to exactly locate objects in space. This can be improved by adding further visual clues like shadows or depth of field information.

Currently, we use a Space Mouse as input device. More intuitive input devices should be evaluated to enable natural interaction with the virtual world.

Another important extension is collision detection and handling. At the moment, no collisions are intercepted and the cables can intersect each other or the background model. However, correct collision handling is crucial for assembly simulation and digital mock up.

At the moment, the presented system is a stand alone application. For using the application in an industry production process, a seamless integration in the construction process is essential. When this is done, a detailed comparison with the traditional design process is mandatory.

Acknowledgements

This work was supported by the Ministry of Innovation, Science, Research and Technology of the State of North Rhine-Westphalia under grant number 80081505.

References

1. Goebbels, G., Hornung, N., Klein, U., Müller, S., Nikitin, I., Wienss, C.: flexengine: Highly accurate real-time simulation system for cables, hoses and wiring harnesses with contacts. In: 55th IWCS 2006, International Wire and Cable Symposium (2006)
2. Cruz-Neira, C., Bierbaum, A., Hartling, P., Just, C., Meinert, K.: Vr juggler – an open source platform for virtual reality applications. In: 40th AIAA Aerospace Sciences Meeting and Exhibit 2002, Reno, Nevada (2002)
3. Tramberend, H.: Avango: A Distributed Virtual Reality Framework. In: Proceedings of AFRIGRAPH 2001. 1st International Conference on Computer Graphics, Virtual Reality and Visualization in Africa, ACM Press, New York (2001)
4. Rohlf, J., Helman, J.: Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In: SIGGRAPH 1994. Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pp. 381–394. ACM Press, New York (1994)
5. de Sá, G., Zachmann, A.: Virtual reality as a tool for verification of assembly and maintenance processes. Computers & Graphics 23, 389–403 (1999)
6. ICIDO: Ido:Flexibles. Web page: produkte/ software/ idoflexibles (2007), `http://www.icido.de/`
7. Grégoire, M., Schömer, E.: Interactive simulation of one-dimensional flexible parts. In: Martin, R.R., Hu, S.M. (eds.) Symposium on Solid and Physical Modeling, pp. 95–103. ACM Press, New York (2006)
8. Burns, D., Osfield, R.: Open scene graph a: Introduction, b: Examples and applications. In: VR 2004. Proceedings of the IEEE Virtual Reality 2004, p. 265. IEEE Computer Society, Washington, DC, USA (2004)

Blur in Human Vision and Increased Visual Realism in Virtual Environments

Michael S. Bittermann, I. Sevil Sariyildiz, and Özer Ciftcioglu

Delft University of Technology

Abstract. A challenge for virtual reality (VR) applications is to increase the realism of an observer's visual experience. For this purpose the variation of the blur that an observer experiences in his/her vision, while he/she focuses on a particular location, can be mimicked by blurring the VR computer graphics based on a model of the blur. The blur in human vision is the result of a combination of optical and neural vision processes; namely optical refraction, non-uniform retinal sampling, and cortical magnification. Due to the complexity of the phenomenon, apparently no theoretical model of the blur has been published. In this work we model the combined effect by means of a probabilistic model of the human visual system. The results from the models match common vision experience verifying the validity of the underlying theoretical considerations. The implementation of the model for increased realism in virtual reality is illustrated by means of a rendering of a virtual reality scene, which is processed for two different acts of focusing.

1 Introduction

Traditional applications of virtual reality (VR) are medical surgery training, flight and driving simulation, as well as industrial and architectural design [1, 2]. In the applications usually a high degree of visual realism is desired for increased effectiveness. Visual realism concerns both, the experienced egocentric and exocentric distances, and the level of visual detail of the environment. A source reducing visual realism of VR experience is the fact that a virtual scene is usually rendered on a 2-dimensional display, so that an observer has some awareness of the display surface [3]. There is evidence that awareness of the display surface interferes with the perceived three-dimensionality of a scene displayed [4, 5]. This evidence appears to corroborate with experiments where absolute egocentric distances are found to be generally underestimated in VR [6, 7], while a number of possible reasons for this can be ruled out [8, 9], including the quality of the computer graphics as claimed by [10].

The visual experience of an object differs from the experience of the object's counterpart in VR. One difference is the experienced variation of visual detail on the object when we focus our vision to a certain location on it. This phenomenon we refer to as *blur* phenomenon. When we focus on a surface patch S_1 of an object, we experience the patch as visually *sharp*. When we keep focusing on S_1, we experience another patch S_2 located at some distance from S_1 as less sharp, i.e. *blur*. The degree of experienced blur depends on the geometric relation among S_1, S_2, and the location of observation. We can verify this by increasing the distance between S_2 and S_1, while we continue to focus on S_1. When we experience an environment in virtual reality we obtain the visual information looking at the surface of the display. While looking at a patch of the display surface the degree of blur we experience for another patch on the surface is clearly dependent on the geometry of the display surface, and not on the geometry of the virtual objects being displayed. Therefore it is clear that the variation of blur we experience viewing the VR rendering may differ significantly from what we would experience if the virtual scene were physically present. This mismatch may be part of the reason why comprehension of the geometry of virtual environments is usually not accurate.

In this work we endeavor to establish a model of the blur occurring in human visual experience of an environment. Based on this model renderings of virtual environments can be processed in such a way that visual experience of them matches more closely to the experience of the corresponding physical environment. The blur phenomenon in human vision is a combined effect of several processes that occur in the human vision system. The processes include refraction of light by cornea, lens, and vitreous, non-uniform retinal sampling, and cortical magnification. The optical refraction of the light yields a *gradient of focus* of the image reaching the retina. The dependence of the focal gradient on distance is well known [11]. The model of focal gradient involves a Gaussian point-spread function characterized by the standard deviation σ. It models the relation between the distance of a location in the environment and the degree of defocus the corresponding image has on the retina, given the lens properties of human eye. The image with its gradient of focus is then processed by the photoreceptors on the retina. The retina has a non-uniform distribution of photoreceptors, i.e. the amount of cone and ganglion cell density is reduced in the periphery of the retina [12] with respect to the central region termed fovea. The non-uniformly sampled retinal image is thereafter mapped to the striate cortex, where another non-linearity is introduced into the processing of the stimulus known as cortical magnification. Cortical magnification refers to the fact that with eccentricity from the center location of the retina, less and less portion of the striate cortex processes the corresponding retinal information [13, 14]. Non-uniform retinal sampling and cortical magnification are considered to cause the well-known sharp decrease of acuity with eccentricity [15], and it has been found that the decrease is proportional to the cortical magnification factor. Since the blur a human experiences in his/her vision is a combined effect of optical and neural processes, modeling the variation of blur is a challenging issue. This is because it is uncertain how the optical effects and neural effects interact precisely, so that the dependence of the blur experience on the geometry of the environment remains essentially unknown.

Due to the uncertainty mentioned above in this work we do not attempt to model each component of the visual system and thereafter combine them to develop a model of blur. In our approach we regard the visual system as a complex system, where the input is the visible environment and the output is the mental realization of the environment. In the approach we distinguish two types of human vision experience that are related. The first type is perception of an environment, where no particular item is intentionally looked at, yet an initial awareness of the visible environment is built up. We refer to this type of experience as perception during *early vision*. The second type of experience is a specific conditioning of the perception during early vision. The condition is that a certain item is gazed at, while the observer has attention for the other objects in view. Given that a certain location in view is gazed at, the attention for the other items in the visual scope takes a certain form. In this *conditional perception* the resulting human vision experience takes a different form compared to the early vision case, namely it yields the experience of blur in human vision. Since the model of the conditional perception is based on the model of perception during early vision, we start by modeling perception during early vision. This is described in section 2. The conditional perception and resulting blur is modeled in section 3. Thereafter an application of the model of blur is presented in section 4. This is followed by conclusions.

2 A Model of Perception During Early Vision

Our subjective impression is that we view our environment in sharp clear focus. In particular, when we assimilate an overall comprehension of a scene we are usually not aware of the variation of visual sharpness that accompanies an intentional act of looking someplace. The phase of initial formation of environmental awareness is known as early vision [16, 17]. Early vision involves eye saccades, optical, retinal and neural processes etc. while an observer is apparently not aware of each act of looking he/she exercises during this phase. This unawareness must be due to complex processing of retinal stimuli in the brain, where attention is selectively paid to environmental information in such a way that an overall awareness of the visible environment is assembled. This means in early vision an observer is viewing locations in the environment in such a way that the net result is equivalent to the event that the observer viewed the entire environment in his/her visual scope. Without prior information about the scene an observer is about to view, he/she pays attention to any direction in his/her visual scope equally. Please note that equally paying attention does not refer to uniform sampling of light stimulus or uniform cortical mapping. It refers the combined effect of the processes interacting in the visual system during early vision. The combined effect results in the phenomenon, that no differential angle in our visual scope has a greater chance to yield awareness of an item that is located within this angle during early vision. In this sense all angles in our view are equivalent at the first instance. Based on this consideration a probabilistic model of early vision is developed, where the *visual attention* paid with respect to vision angle is modeled by means of a uniform probability density function (pdf) [18]. This mathematical definition of attention substantiates the existing verbal definitions of the concept. Any point of a visible environment "receives" a particular degree of the attention paid. To compute the distribution of the observer's attention over the environment we consider a fundamental visual geometry shown in figure 1. In the figure an observer at the point *P* is viewing an infinite plane whose intersection with the plane of page is the line passing from two points designated as *O* and *O'*. *O* represents the origin. We compute the variation of attention along the *r* axis shown in figure 1a. The pdf $f_r(r)$ of this random variable is computed using the theorem on the function of random variable. The resulting pdf is [18]

$$f_r(r) = \frac{l_o}{\pi/2} \frac{\sin(\varphi)}{r^2 - 2l_o r\cos\varphi + l_o{}^2} \qquad \text{for} \qquad 0 < r < \frac{l_0}{\cos(\varphi)}. \tag{1}$$

Equation 1 gives the variation of visual attention during early vision along the axis *r*. To compare this variation for different angles of *φ* we obtain the variation of attention with respect to the direction parallel to the *y*-axis. Figure 1b is the magnified portion of figure 1a in the vicinity of the origin indicating the relationship between the infinitesimally small distances *dy* and *dr*.

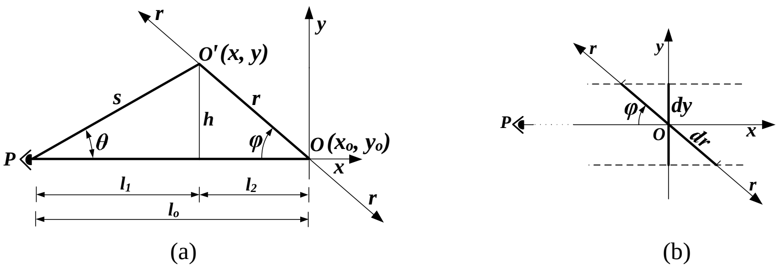


Fig. 1. The geometry of visual perception, where the observer has the position at point *P* with the orientation to the point *O*. The *x,y* coordinate system has the origin placed at *O* (a); the magnified region at the origin (b).

The relation between *dy* and *dr* is given by *dy =dr sin(φ)* or

$$dr=dy/\sin(\varphi) \tag{2}$$

Substitution of (2) into (1) yields

$$f_r^*(r) = \frac{l_o}{\pi/2} \frac{1}{r^2 - 2l_o r\cos\varphi + l_o{}^2} \tag{3}$$

The plot of $f_r^*(r)$ in Cartesian coordinates is shown in figure 2a. Based on the probabilistic expression of attention, perception is defined as the integral of attention over a certain geometric domain [18].

3 A Model of the Blur in Human Vision

Based on the result from the previous section we aim to model the situation when an observer gazes at an object. In this case there is still perception of other objects within the visual scope. Considering the probability density $f_r(r)$ given by (3), we can derive a conditional density conditioned by the event of seeing point *O* in figure 1. This means the observer is gazing at point *O* while he still has some attention for *O'*. This conditional attention can be calculated by means of Bayes's theorem yielding the probability density along the line *OO'* while the observer focuses at the point *O*. We note that this application of the Bayes' theorem differs significantly from the existing Bayesian approaches to perception. In contrast to the *probabilistic model* we develop in this paper, the Bayesian approach to perception in literature is a statistical inference approach carried out in terms of *probability* statements. The vast majority of probabilistic approaches in the literature for perception modeling is based on the Bayes' rule. The present probabilistic approach is a complementary endeavor for the Bayesian approach providing theoretical base for the prior density. Therefore the posterior density can be obtained more accurately in the Bayesian approach.

The computation of visual blur is accomplished as follows. The probability density $f_r^*(r)$ in (3) is also a function of the angle φ. Therefore we can consider that it is a function of two variables, which are r and φ, and thereby we write it in the form $f_r^*(r, \varphi)$, which is given by

$$f_r^*(r,\varphi) = \frac{l_o}{r^2 - 2l_o r\cos\varphi + l_o^2} \; . \tag{4}$$

Above the factor $2/\pi$ is omitted taking the angle θ as a unit angle, so that $f_\theta=1$ as compared to $f_\theta=1/(\pi/2)$ [18]. One can interpret that f_r is a marginal joint probability density of a two dimensional joint probability density. This means $f_r(r|\varphi)$ gives the probability density with respect to r at the angle φ. With this understanding, there is no objection to apply the Bayes' theorem for the joint probability density. Accordingly we write

$$f_r(r\,|\,\varphi) = \frac{f_r(r,\varphi)}{f(\varphi)} \tag{5}$$

where $f_r(r|\varphi)$ is the conditional probability density with respect to r; $f(\varphi)$ is the marginal probability of the joint probability density $f_r(r,\varphi)$ at the angle φ, and therefore the Bayes' theorem is applicable without any difficulty. It is noteworthy to mention that in the continuous case the formulation of conditional probability presents some difficulty since the theorem requires a probability at the denominator rather than a probability density, which is the case in the continuous case. This is described in [19]. From (5), we obtain $f(\varphi)$ as [18]

$$f(\varphi) = \left[\arctan\frac{\tan(\theta)\tan(\varphi)-1}{\tan(\theta)+\tan(\varphi)}\right] + \frac{\pi}{2} - \varphi \tag{6}$$

where $\tan(\theta)$ is

$$\tan(\theta) = \frac{r\sin(\varphi)}{l_o - r\cos(\varphi)} \tag{7}$$

Substitution of (6) and (2) into (4) yields

$$f_r^*(r\,|\,\varphi) = \frac{l_o}{f(\varphi)}\,\frac{1}{r^2 - 2l_o r\cos\varphi + l_o^2} \, . \tag{8}$$

The probability density $f_r^*(r|\varphi)$ gives the conditional attention, i.e. the attention conditioned on φ, while the observer focuses on the point O. This attention we define as sharpness. The inverse of sharpness we define as *blur*.

The visual attention $f_r^*(r,\varphi)$, that is perception per unit length, is given by (4) and presented in figure 2a. The variation of conditional perception per unit length $f_r^*(r|\varphi)$

given by (8) is presented in figure 2b. The latter probability density indicates the degree of attention for an object in the case where another object receives full attention. In other words, when we gaze at an object at *O*, which is located at the point *y=x=0* in figure 2b, we still have attention for other objects with a lesser degree, so that becoming aware of a second object located at *O'* is subject to probability. This probability quantifies the degree of seeing the object at *O'*.

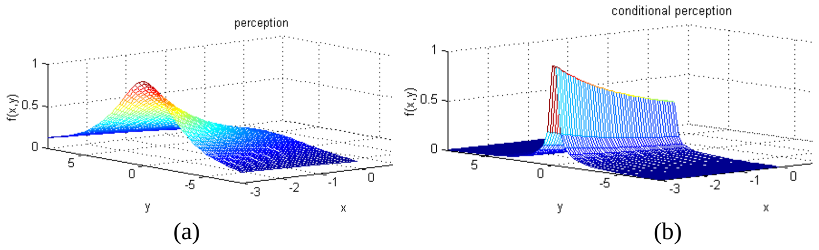


Fig. 2. Computed perception per unit length (a) and conditional perception per unit length (b)

Comparing figure 2a and 2b we note that in the early vision case the attention, which is the perception per unit length, is distributed more evenly throughout space compared to the conditional perception case. This means the awareness for objects in the vision periphery diminishes gradually in early vision compared to the case, where an item is gazed at, so that awareness diminishes sharply.

In the following we consider some implications of the pdf of conditional perception shown in figure 2b. As examples we consider two acts of focusing. These are shown in figure 3a. In the first case the focus is on point *O* shown in the figure and the conditional perception is considered along the line *OO'*. In the second case the focus is switched to the object *O'*, so that it becomes the new focus point labelled *(O)* in figure 3a, and the conditional perception is considered along the line between (*O)* and the other point labeled *(O')*. In the first case the angle $\varphi_1=3\pi/4$ and $l_o=8m$. In the second case $\varphi_2=\pi/25$ and $l_o=14m$. Figure 3b shows a vertical section of a scene subject to conditional perception, having the same geometry as shown in figure 3a.

The variation of sharpness with increasing distance *r* is obtained as the intersection of a vertical plane with the conditional perception surface presented in figure 2b, where the plane is oriented with φ and passing from the object *O*. Two plots of sharpness with respect to *r* for the two acts of focusing shown in figure 3a are shown in figure 4a and 4b. Figure 4a shows the sharpness along *r for* $\varphi=3\pi/4$, l_o=8m. Figure 4b shows the sharpness along *r* for $\varphi=\pi/25$, l_o=14m. In both cases the sharpness diminishes with the distance *r*, however in the first case it diminishes sharper than in the second case. In figure 4b we observe that for distances *r*>3m sharpness does not diminish as drastically as in figure 4a, but it remains practically constant at a certain level and even it increases as the distance increases towards $r=l_o$. This means, when we gaze at a nearby object, a second object located at a greater distance appears much more blur than in the reverse situation if we focus on the distant object. In the latter case the nearer object appears still relatively sharp. This is demonstrated in the application section.

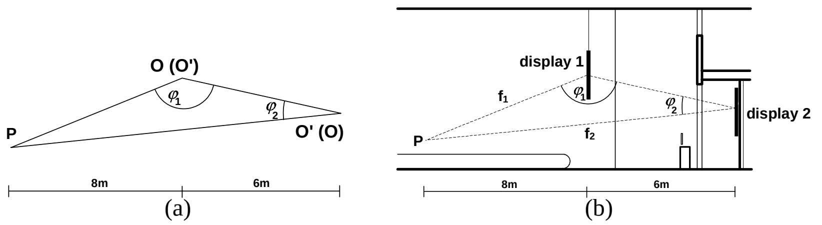


Fig. 3. (a) Sketch showing two acts of focusing: for $\varphi_1=3\pi/4$, $l_o=8m$ and $\varphi_2=\pi/25$, $l_o=14m$; (b) Vertical section of a scene subject to conditional perception, with the same geometry as in 3a

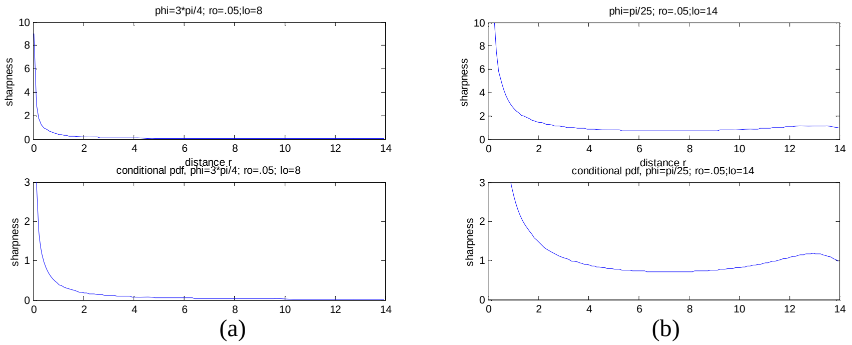


Fig. 4. (a) Plot of sharpness with respect to the distance r for $\varphi=3\pi/4$; l_o=8m, where r corresponds to the line OO' in figure 3a; r_o is the distance at which the plot starts; (b) Plot of sharpness with respect to r for $\varphi=\pi/25$; $l_o=14m$, where r corresponds to the line $(O)(O')$ in figure 3a

4 Application

The model of perception during early vision presented in section 2 has found application in robot navigation [20] and design [21] as reported earlier. In the following we investigate the application of the model of visual blur/sharpness presented in section 3 for VR environments with the objective to increase the visual realism of an observer's experience. The issue we are concerned about in the application is the influence blur has on depth perception. Virtual reality graphics are usually shown on a two dimensional display. An observer is looking at a particular pixel on the computer display at a given moment. This location can be obtained by means of eye tracking for example. Having this information, the degree of blur can be computed for every surface patch of a virtual scene, so that the rendered blur matches the stimulus an observer would obtain in case the scene the observer is experiencing were physically present. For each new act of looking the blur should be updated in real-time. To simulate the implementation we apply the results from the model of blur to a rendering of a virtual reality scene. The scene is the one shown in figure 3b, which belongs to the computations presented in figure 4. As shown in figure 3b we consider two objects of a scene as the targets of an observer's visual focus; the objects in our case are two flight schedule displays. These objects we take as an example for the blur phenomenon modelled. The size of the displays in the scene is equal. *Display 1* is located nearby to the observer, who is located at the place denoted by *P* as shown in figure 3b. *Display 2* is located beyond the first one. We compare two perception events of the observer: the first one is gazing at a place on display 1, and the second one gazing at a place on display 2. The gaze line belonging to the first event is shown in figure 3b by means of a dashed line labeled f_1. The gaze line belonging to the second event is shown by means of a dashed line labeled f_2. Figure 5 shows the rendering of the scene shown in figure 3b without blur. The rendering is then processed based on the result from the model presented in figure 4a and 4b and the result is shown in figures 6 and 7.

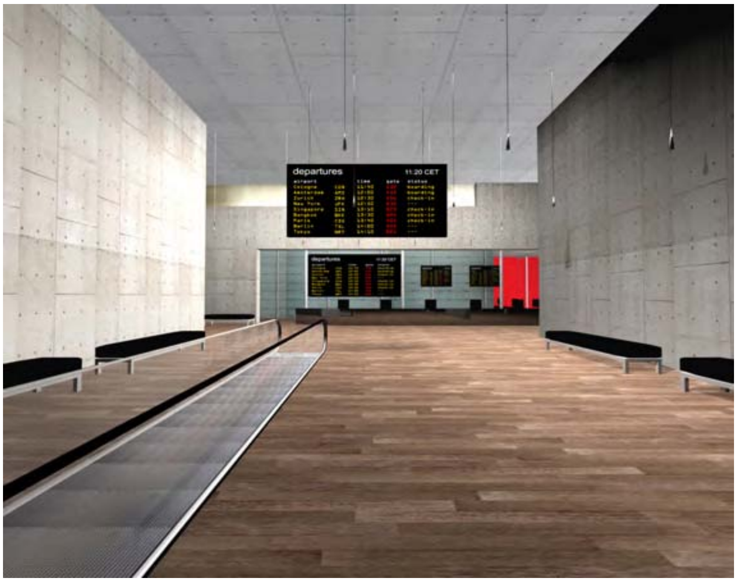


Fig. 5. Endering of the VR scene shown in figure 3b without perceptual blur

Figure 6 shows the event that the observer focuses on the plus sign on *display 1*. Figure 7 shows the event that the observer focuses on the plus sign on *display 2*.

Let us consider figure 6: When an observer is focusing on the plus sign on *display 1*, *display 2* is perceived with significant blur. This is because the angle φ_1 is quite large as shown in figure 3b, so that the sharpness quickly decreases with increasing distance, as shown in figure 4a. We can verify the increased realism by keeping our visual focus on the plus sign in figure 6 and at the same time perceiving *display 2*. Since the blur of *display 2* is excessive, we clearly get the impression that *display 1* is located nearer to us than *display 2*. Let us consider figure 7: When an observer is focusing on *display 2*, *display 1* is experienced as relatively sharp compared to *display 2* in the case shown in figure 6. This is because *display 1* is located nearer to the observer than *display 2*. The angle φ_2 shown in figure 3b is small, so that the sharpness of *display 1* is relatively high, as this is shown in figure 4b. This can be verified by keeping the visual focus on the plus sign in the figure and at the same time perceiving display 1. Again, one clearly has the impression that display 1 is nearer to our

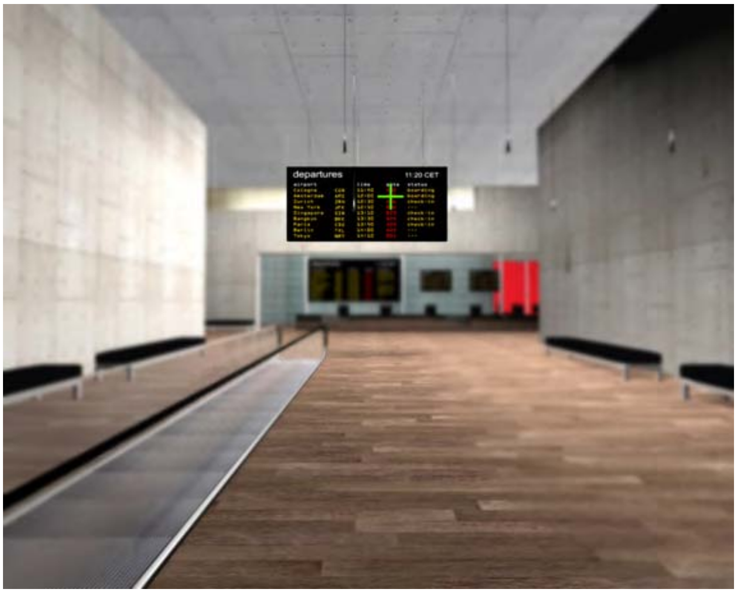


Fig. 6. Rendering of the VR scene shown in figure 3b blurred based on the result presented in figure 4a; the blur is according to the event of focusing on the plus sign on display 1

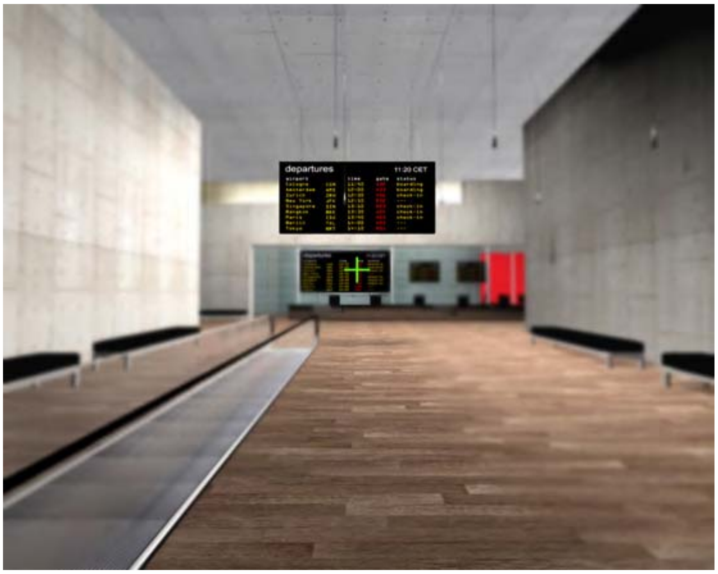


Fig. 7. Rendering of the VR scene shown in figure 3b blurred based on the result presented in figure 4b; the blur is according to the event of focusing on the plus sign on display 2

viewpoint than *display 2*. Let us consider the blur of the floor and of the ceiling in figures 6 and 7. We note that locations directly below or above the focus point are relatively sharp compared to locations at the side regions of the floor or ceiling. This is because the angle φ for the locations directly above/below is quite small compared to the places on the side. For illustrative purposes this effect is exaggerated in the figures. It is noteworthy to mention that when the reader focuses on the plus signs in figures 6 and 7 an additional blur component occurs that should be taken into consideration in the preparation of a perception blurred image. Namely, when we focus on the page of the paper on which the scene is displayed, naturally a blur is induced that depends on the geometry of the paper surface. This additional blur should be accounted for by means of convolution of this blur with the blur that occurs due to the geometry of the scene. In order to avoid interference of this additional blur with the blur induced by the geometry of the virtual scene, the images shown in the paper should be observed from a large distance.

The added value of the perception based image processing is that the observer gains a better understanding of the geometry of the virtual scene compared to the case he/she is viewing a conventional rendering without perceptual blur. This is because the stimulus presented via a perceptually blurred image matches more closely to the perception in a corresponding physical environment than the matching to gazing at an image that is equally sharp everywhere. In order to experience the improved realism in figures 6 and 7 we emphasize that the reader keeps focusing on the plus sign shown in the figures. Certainly blur is only one of the depth cues relevant for comprehension of environmental geometry. However we have reason to consider the variation of blur a significant source of visual information on an environment. Every act of looking to a certain location yields a particular degree of blur for every patch of the visible surface of the environment. As the variation of blur depends on the geometry of the environment, every act of focusing yields a unique blur pattern that encodes the geometric constitution of the environment. Since there are a great many visible environmental patches within our visual scope there is a significant amount of information available for the brain to attain comprehension of an environmental geometry with every act of looking. With successive saccades, where different locations in the environment are focused, the amount of information increases, so that the environmental geometry is usually comprehended unambiguously.

5 Conclusion

A model of blur human experiences in his/her vision is presented and its application for VR is exemplified. The blur in human vision is considered as a combined effect of a number of processes involved in vision. Therefore the visual system is modeled as a complex system with probabilistic properties. As the work provides a theoretical base for modeling the blur it is a significant step to understand the phenomenon in detail and to introduce the concept in VR. We note that the probabilistic approach to perception presented in this work is complementary to the existing Bayesian approaches to perception, which are essentially statistical, in contrast to the present work, which is without statistical considerations. We show that the variation of blur is depending on the geometric relation between focus point and the second point considered, and we quantify this dependency. The dependency has apparently not been described before. The model of blur reveals that objects located in the region between the observer and the place where he/she is looking at are perceived sharper than objects located beyond the focal point. The corroboration of the theoretical results with common vision experience verifies the validity of the theoretical base of the work. In particular we note that the axiomatic starting point of the model development is verified. The model is applied to process the rendering of a VR scene for two different gazing acts of an observer. Based on the model outcome the degree of perceptual blur an observer would experience if the scene were physically present is simulated. The resulting perceptually blurred rendering of a virtual reality provides increased visual realism. This is because the visual stimulus provided by the perceptually blurred rendering matches more closely to the experience a human has when he/she would be experiencing the corresponding real environment, compared to a rendering where every pixel is equally sharp. A uniformly sharp image of a scene is an unrealistic representation of its physical counterpart in the sense that it requires excessive memory to hold all the details of the scene in case of a human observer. Application of perceptual blur in VR may contribute to alleviating the phenomenon of distance underestimation in VR and may support the provision of a more realistic spatial experience in VR, in particular with respect to large objects or spaces. Interestingly, in the present implementation increased realism is achieved not by increasing the amount of visual information of an image stimulus, but in contrary, by selectively reducing the information, as it apparently occurs in human vision system during an intentional act of looking.

References

1. Hettinger, L.J., Haas, M.W.: Virtual and Adaptive Environments: Applications, Implications, and Human Performance Issues. Lawrence Erlbaum, Mahwah (2003)
2. Ong, S.K., Nee, A.Y.C.: Virtual Reality and Augmented Reality Applications in Manufacturing. Springer, London (2004)
3. Polanyi, M.: Personal Knowledge. Routledge and Kegan Paul, London (1978)
4. Pirenne, M.H.: Optics, Painting, and Photography. Cambridge University Press, Cambridge (1970)
5. Yang, T.L., Dixon, M.W., Proffitt, D.R.: Seeing big things: Overestimation of heights is greater for real objects than for objects in pictures. Perception 28, 445–467 (1999)
6. Loomis, J.M., Knapp, J.M.: Visual perception of egocentric distance in real and virtual environments. In: Hettinger, L.J., Haas, M.W. (eds.) Virtual and Adaptive Environments, pp. 21–46 (2003)
7. Plumert, J.M., Kearney, J.K., Cremer, J.F., Recker, K.: Distance perception in real and virtual environments. ACM Trans. Appl. Percept. 2, 216–233 (2005)
8. Knapp, J.M., Loomis, J.M.: Limited field of view of head-mounted displays is not the cause of distance underestimation in virtual environments. Presence: Teleoperators and Virtual Environments 13, 572–577 (2004)
9. Creem-Regehr, S.H., Willemsen, P., Gooch, A.A., Thompson, W.B.: The influence of restricted viewing conditions on egocentric distance perception: Implications for real and virtual environments. Perception 34, 191–204 (2005)

10. Thompson, W.B., Willemsen, P., Gooch, A.A., Creem-Regher, S.H., Loomis, J.M., Beall, A.C.: Does the quality of the computer graphics matter when judging distance in visually immersive environments? Presence: Teleoperators and Virtual Environments 13, 560–571 (2004)
11. Pentland, A.P.: A new sense of depth. IEEE Trans. on Pattern Analysis and Machine Intelligence 9, 523–531 (1987)
12. Hirsch, J., Curcio, C.A.: The spatial resolution capacity of human foveal retina. Vision Research 29, 1095–1101 (1989)
13. Cowey, A., Rolls, E.T.: Human cortical magnification factor and its relation to visual acuity. Experimental Brain Research 21, 447–454 (1974)
14. Sereno, M.I., Dale, A.M., Reppas, J.B., Kwong, K.K., Belliveau, J.W., Brady, T.J., Rosen, B.R., Tootell, R.B.: Borders of multiple visual areas in humans revealed by functional magnetic resonance imaging Science. 268, 889–893 (1995)
15. Wertheim, T.: Ueber die indirekte Sehschaerfe. Sinnesorg 7, 172–189 (1894)
16. Van Hemmen, L., Cowan, J., Domany, E.: Models of Neural Networks IV: Early Vision and Attention. Springer, Heidelberg (2001)
17. Adelson, E.H., Bergen, J.R.: The Plenoptic function and the elements of early vision. In: Landy, M., Movshon, J.A. (eds.) Computational Models of Visual Processing, pp. 3–20. MIT Press, Cambridge (1991)
18. Ciftcioglu, Ö., Bittermann, M.S., Sariyildiz, I.S.: Visual perception theory underlying perceptual navigation. In: Emerging Technologies, Robotics and Control Systems International Society for Advanced Research, pp. 139–153 (2007)
19. Meyer, P.L.: Introductory Probability and Statistical Applications. Addison-Wesley, Reading (1970)
20. Ciftcioglu, O., Bittermann, M.S., Sariyildiz, I.S.: Multiresolutional fusion of perceptions applied to robot navigation. Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII) 11, 688–700 (2007)
21. Bittermann, M.S., Sariyildiz, I.S., Ciftcioglu, Ö.: Visual perception in design and robotics. Integrated Computer-Aided Engineering 14, 73–91 (2007)

Embedded Multigrid Approach for Real-Time Volumetric Deformation

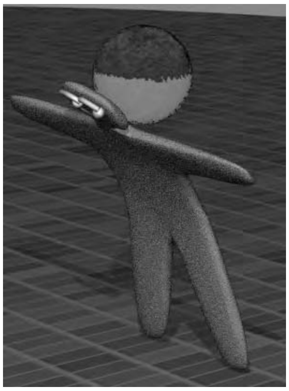
Guillaume Saupin[1], Christian Duriez[2], and Laurent Grisoni[2]

[1] CEA, LIST, 18 route du panorama, 92265 Fontenay aux roses, France
[2] LIFL, IRCICA / INRIA Futurs, USTL France
guillaume.saupin@cea.fr, christian.duriez@inria.fr, Laurent.Grisoni@lifl.fr

Abstract. Finding efficient and physically based methods to interactively simulate deformable objects is a challenging issue. The most promising methods addressing this issue are based on finite elements and multigrid solvers. However, these multigrid methods still suffer, when used to simulate large deformations, from two pitfalls, depending on the kind of grids hierarchy used. If embedded grids are used, approximating complex geometries becomes difficult, whereas when unstructured grids hierarchy is used, solving speed-up is reduced by the necessity to update coarser levels stiffness matrices. We propose a framework that combines embedded grids solving with fast remeshing. We introduce a new hierarchical mesh generator which can build a hierarchy of topologically embedded grids approximating a complex geometry. We also show how to take advantage of the knowledge of the stiffness matrix sparsity pattern to efficiently update coarse matrices. These methods are tested on interactive simulation of deformable solids undergoing large deformations.

1 Introduction

With the impressive CPU power growth over the last decade, computers finally meet hopes of the industrial and medical sectors. Thus, interactive simulations of deformable bodies, initially using naive deformable models chosen for their light computation cost, can now handle more and more realistic models in an interactive way.



However, non-linear models, real-time or interactive Finite Element Method (FEM) simulations are still limited to "small" deformable bodies, with few degrees of freedom, especially with implicit time integration. This limitation is the consequence of the solving process computation time.

This problem has been addressed by multigrid solvers, and significant speed-up has been achieved [1,2,3]. These solvers rely on multiple levels of grids to hierarchically refine a coarse solution. Two kinds of grids hierarchy can be used with these solvers: embedded and unstructured grids.

On one hand, embedded grids have the advantage that the topological link between two consecutive grids leads to fast update of the coarser matrices. Moreover, hierarchical basis functions used in embedded grids allow the construction of adaptive solver. However, this kind of hierarchy is difficult to use for complex geometry, as it demands a coherent 3D mesh structuration.

On the other hand, unstructured grids allows complex geometry approximation. However, coarse matrices updates become a bottleneck that significantly slows down the simulation.

In this paper, we propose a framework that combines a novel method for embedded multigrid solving with a simple, yet efficient, hierarchical remesher. We identify in the presented work two contributions:

– First, we introduce a simple hierarchical mesher which approximates complex geometries with a topologically embedded hierarchy of 3D meshes.
– Second, we show how the knowledge of the stiffness matrix sparcity pattern allows to overcome the main bottleneck of multigrid methods when unstructured grids are used.

We illustrate the efficiency of these two improvements in the case of corotational simulation of deformable bodies undergoing large deformations.

2 Previous Work

To address this need of fast solvers for accurate and physically plausible simulations, many methods have been proposed. For materials undergoing large strains, explicit temporal integrations of dynamics FEM/Boundary Element Method (BEM) objects have been used [4,5,6,7,8,9]. Although these methods are good schemes for soft materials, they could become time consuming for dynamic models that are physically stiff, incompressible or have a detailed discretization.

More precise results can be achieved using adaptative solvers [10,11,12,13,14, 15,16]. These solvers start with a coarse model, and add details using refinement functions on specific area of the model. Thus the number of degrees of freedom is reduced for the same precision. However, these methods need an heuristic, usually an activation threshold [10, 13] based on strain measurement, to find where to adapt the model. Moreover, when the model undergoes a large and global deformation, no speed up is achieved because all the refinement basis functions are activated. Finally, the different levels of the hierarchy are not used to improve the solver speed. They are only used to compute a more precise solution using usual solver like Conjugated Gradient.

On the contrary, multigrid solvers [17,1] take full advantage of the hierarchy of grids to reduce solving time. They also have the property of being independent of object stiffness and can then handle heterogeneous material with different stiffness [1]. An iterative process computes a correction at the coarser levels, taking into account the residual at the finest level. This correction is then transfered to the fine level. The transfers from fine to coarse and coarse to fine levels are done using respectively restriction and prolongation operators. If the mesh is embedded, average interpolators are used. If the mesh is unstructured, Georgii proposed in [1] to use barycentric interpolators. This kind of solver can also be used for geometry-based mesh deformation [2].

As stated in the introduction, both embedded and unstructured grids hierarchies have drawbacks. Embedded grids can not easily approximate complex geometries. Unstructured grids imply time consuming coarse matrices updates. In this paper, we propose to address the embedded hierarchy drawback by introducing a hierarchical mesher. The unstructured multigrid issue will be addressed by a fast coarse matrix update algorithm taking advantage of stiffness matrix sparsity knowledge.

3 Hierarchical Mesh Generator

3.1 Tetrahedra Subdivision

Embedded hierarchies of multigrid in 3D volume case are built from a coarse mesh Ω_0 obtained using a decimation tool. Then, for each tetrahedron T_m of the coarse mesh, a new node n_{new} is added in the middle of each edge of T_m and 8 new tetrahedra are built using this nodes. The same decomposition is used for the next levels of hierarchy. See Figure 1.

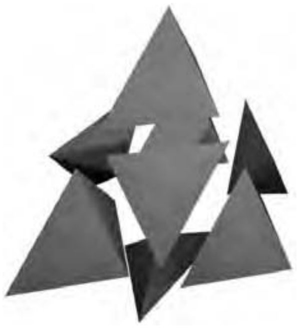


Fig. 1. Refined tetrahedron

3.2 Hierarchical Mesher

We use usual decimation tools to build the coarse approximation $\partial\Omega_0$ of the boundary target $\partial\Omega$. This initial coarse boundary mesh $\partial\Omega_0$ is then meshed using a volume mesh generator.

Due to the construction scheme above, the boundary $\partial\Omega_i$ at each level i is identical to $\partial\Omega_0$ since each new nodes n_{new} of $\partial\Omega_{i+1}$ already belongs to $\partial\Omega_i$.

Thus, we introduce a heuristic for a geometrical approximation step in the refinement scheme: each new node n_{new} that belongs to the boundary of the mesh is moved in order to fit a target boundary geometry $\partial\Omega$. See Algorithm 1.

3.3 Surface Nodes Heuristic

The proposed heuristic is based on very simple geometrical rules: for each node of the boundary, we find the three closest triangles. This gives us the point of each triangle which is the closest to the node. In convex cases, it is a simple projection on the triangles. In concave cases, this point can be on an edge or on a vertex of the triangles. Then, we move the node on the barycentric center of theses three points. This technique locally smooths the surface of the mesh while remaining close to the surface of origin.

Algorithm 1. Hierarchical Mesh Generator

```
Require: $\partial\Omega$, $\Omega_0$, $n_{level}$
Ensure: $\partial\Omega_{n_{level}-1}$ is a good approximation of $\partial\Omega$.
 1: for $i$ in $[0, n_{levels} - 1]$ do
 2:   for each tetrahedron $T$ in $\Omega_i$ do
 3:     Subdivide $T$ in $T_0, \ldots, T_7$.
 4:     for each new node $n_{new}$ added do
 5:       if two parent nodes where on $\partial\Omega_i$ then
 6:         $n_{new}$ is on $\partial\Omega_{i+1}$.
 7:       end if
 8:     end for
 9:   end for
10:   Set $\Omega_{i+1}$ as $\bigcup_{m=1}^{N_{tet}^{i+1}} T_m$, where $N_{tet}^{i+1} = 8N_{tet}^{i}$
11:   for each node $n_k$ in $\partial\Omega_{i+1}$ do
12:     Find the 3 closest triangles $t$ on $\partial\Omega$
13:     Find the closest points to $n_k$ on each 3 triangles.
14:     Move $n_k$ on the barycenter of theses 3 points.
15:   end for
16: end for
```

3.4 Examples

Our hierarchical mesh generator has been tested successfully on various meshes. See Figure 2. It has proved to always converged to the target $\partial\Omega$ under the condition that enough levels are used. However, a good initial appoximation $\partial\Omega_0$ of $\partial\Omega$ is necessary to converge in a minimal number of levels. Hence, it is sometime necessary to manually edit $\partial\Omega_0$ to have a good approximation of the salient points, like the ears of the kitten in Figure 2.

4 Multigrid Solver

Multigrid solvers are a well known class of solvers that use a grids hierarchy to efficiently and quickly solve differential equations. When used in a finite elements context, grids are referred as meshes. They use the fact that relaxation scheme, like the Jacobi or Gauss-Seidel methods, even though they have a low convergence rate, can efficiently solve an equation provided a good initial guess is used. This initial guess can be computed easily and efficiently by solving the problem at a coarser level.

Restriction and prolongation operators R and P, which allow respectively to switch from fine to coarse resolution and inversely, are used to build the problem at a coarser resolution, and to transfer result from one level to the other.

For instance, if we try to solve $A_i x_i = b_i$ on the fine level, we can compute an initial guess x_i^0 by solving $A_{i-1}x_{i-1} = b_{i-1}$ on the coarse level and then transfer the solution to the fine level: $x_i^0 = Px_{i-1}$, where $A_{i-1} = RA_iP$.

x_i can then be computed by Gauss-Seidel relaxation of x_i^0.

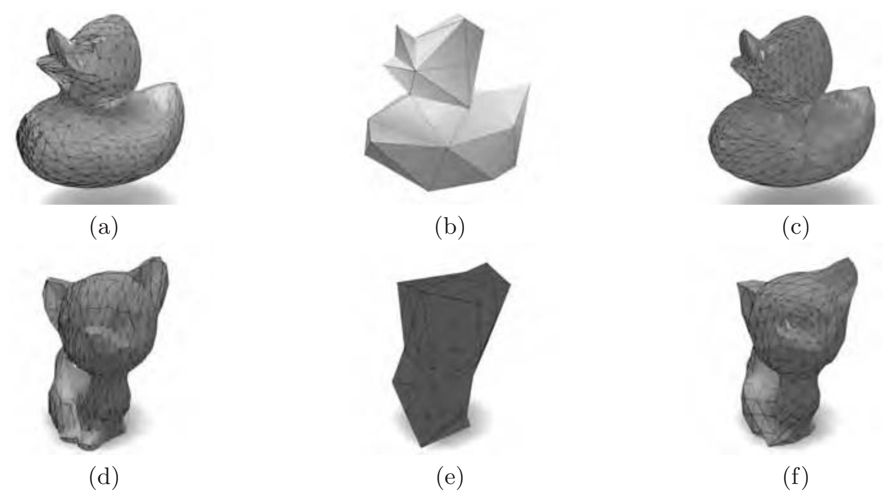


Fig. 2. (a): Original mesh; (b): Coarsest mesh; (c): Hierarchically remeshed. (d): Original mesh; (e): Coarsest mesh. (f): Hierarchically remeshed.

However, this method is sub-optimal, since it has been shown [3] that relaxation methods are improved when applied on an non-smooth signal. Hence it is more efficient to use the following method:

1. Compute residual: $r_i = A_i x_i^0 - b_i$
2. transfer residual to coarse mesh: $r_{i-1} = Rr_i$.
3. Find v_{i-1} so that $A_{i-1}v_{i-1} = r_{i-1}$
4. Transfer correction to fine mesh: $v_i = Pv_{i-1}$.
5. Apply correction: $x_i = x_i^0 + v_i$. (x_i is non-smooth as the residual r_i wasn't)
6. Smooth $A_i x_i = b_i$ using α Gauss-Seidel relaxations.

We can then derive the well-known full V-Cycle algorithm for multigrid as shown in Algorithm 2.

The prolongation operator P is usually an interpolating operator. If the meshes were embedded, the mean interpolation operator would be used, *i.e.* each new node of the fine mesh Ω_i would be the average of his two neighbors nodes. If the meshes were unstructured, we would compute the barycentric coordinates $w^k_{1,2,3,4}$ of each node n^k of the fine mesh Ω_i in the coarse mesh Ω_{i-1}. Thus, the value at these nodes is $x_i^k = \sum_{l=1}^{4} w_l^k x_{i-1,l}^k$.

The restriction operator is usually $R = P^T$.

5 Fast Coarse Matrices Update

The stumbling point in Algorithm 2 is the computation of the coarse matrix $A_{i-1} = RA_iP$. The models used in simulations usually have thousands tetrahedra which implies huge A_i matrices. Therefore the product RA_iP is time consuming, even thought R, P and A_i are highly sparse.

Algorithm 2. V-Cycle Solver

```
Require: $A_i, b_i, x_i^0, R, P$
Ensure: $x^i$ such that $A_i x^i = b_i$
 1: if Mesh is the coarsest one then
 2:    Solve $A_i x_i = b_i$.
 3: else
 4:    Compute coarse matrix $A_{i-1} = R A_i P$.
 5:    for $j$ in $[0, maxIter]$ do
 6:       Smooth $\alpha_1$ times $A_i x_i^0 = b_i$ using Gauss Seidel.
 7:       Compute residual: $r_i = A_i x_i^0 - b_i$
 8:       Transfer residual to coarse mesh: $r_{i-1} = R r_i$.
 9:       Recursively solve $A_{i-1} v_{i-1} = r_{i-1}$ using V-Cycle.
10:       Update $x_i$: $x_i = x_i + P v_{i-1}$.
11:       Smooth $\alpha_2$ times $A_i x_i = b_i$ using $\alpha$ Gauss Seidel.
12:    end for
13: end if
```

This is mainly due to the fact that multiplying two sparse matrices, when they are stored in the usual Compressed Sparse Row (CSR) format, is very inefficient.

This problem is recurrent in every multigrid solver, like the barycentric one proposed in [1], [2].

However, the P, R and A_i have the remarkable property that their sparsity pattern is the same for the whole simulation. And this important property allows some precomputations which will improve the computation time.

We propose to compute this product in two steps:

$$C = A_i \star P \tag{1}$$

$$A_{i-1} = R \star C \tag{2}$$

where the $\star$ operator, our optimized matrix product, works as follows:

1. Once the sparcity patterns of P and A^i are known, we compute the *intersection* for Equation (1) as follows: we find for each element $c_{k,n}$ of the matrix C the list of pairs $p = (index(p_{k,l}), index(a^i_{m,n}))$ such that $p_{k,l} \neq 0$ and $a^i_{m,n} \neq 0$. These lists of pairs are stored in an array to compute the $c_{k,n}$ elements. In addition to each list, we store the index of the corresponding $c_{k,n}$ in the C matrix.
2. Each time we need to compute C, we move through the array of list of pairs, do the necessary products, and store the result at the given index.
3. We repeat steps 1 and 2 for Equation 2.

The speed up observed on a bi Dual Core 2 Duo 5130 using our method is presented in Table 1, and compared with the naive methods using Intel MKL sparse matrix product, and the results obtained in [1]. We observed a 2.0 to 3.0 speed up factor compared to [1].

Table 1. Benchmark - 2 Dual Core 2.0 Ghz

#Tetra	#levels	MKL update (ms)	Georgii [1] update (ms)	optimized update (ms)
432	2	9		4
1148	2	151	∼ 25	14
3456	2	464		37
7168	3	2377	∼ 170	76

6 Deformable Solid Simulation

We can now use the methods presented above to efficiently simulate deformable bodies undergoing large displacements.

6.1 Elasticity Theory

The motion of the solid Ω subject to the force f and to constraints on its boundary is described by the system of equations below:

$$\begin{cases} \rho\frac{\partial^2 u}{\partial t^2} + \nabla\cdot\sigma + b\frac{\partial u}{\partial t} + f = 0 \text{ on } \Omega \\ \sigma = C : \epsilon \\ u = u_{constrained} \text{ on } \partial\Omega_C \\ \sigma(u)n = \sigma_{constrained} \text{ on } \partial\Omega_N \end{cases} \tag{3}$$

where C is the well known fourth rank elasticity tensor, *i.e.* the expression of Hooke's law, ϵ is the strain tensor, σ is the internal stress tensor, ρ is the density, b is the damping factor, u is the displacement and $\partial\Omega$ is the boundary of Ω.

The first equation gives the dynamics of the material. The second is the constitutive law for elastic material. The third one expresses displacement constraint on $\partial\Omega_C$, and the fourth corresponds to the stress applied on $\partial\Omega_N$. See Figure 3.

We derive the weak form of this equation using the test function v:

$$\int_\Omega v\left(\rho\frac{\partial^2 u}{\partial t^2} + \nabla\sigma + b\frac{\partial u}{\partial t} + f\right)d\Omega = 0, \forall v \tag{4}$$

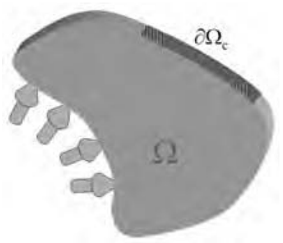


Fig. 3. The domain Ω and its constraints

Using the divergence theorem, we get:

$$\begin{aligned} &\int_\Omega v\rho\frac{\partial^2 u}{\partial t^2}d\Omega + \int_{\partial\Omega} v\sigma n d\partial\Omega - \\ &\int_\Omega \nabla v : \sigma d\Omega + \int_\Omega vb\frac{\partial u}{\partial t}d\Omega + \\ &\int_\Omega vfd\Omega = 0 \end{aligned} \tag{5}$$

Small Displacement. When the object undergoes only small displacements, the Cauchy strain tensor can be used:

$$\epsilon^C = \frac{1}{2} \cdot (\nabla u^T + \nabla u) \tag{6}$$

6.2 Discretization

We can discretize Equation 5 using finite element method. The discrete approximation of domain Ω is built using our hierarchical mesh generator. Then, we build the volume mesh $\Omega_i = \bigcup_{m=1}^{N_{tet}} T_m$.

Then, we can use Ω_i to integrate Equation 5, getting the usual equation of motion:

$$M \cdot \ddot{u} + B \cdot \dot{u} + K \cdot u = f \tag{7}$$

$$\text{Where} \begin{cases} M = (m_{i,j})\,, m_{i,j} = \int_{T_m} \varphi_i^T \cdot \rho \cdot \varphi_j \cdot dT_m \\ K \cdot u = (k_{i,j})\,, k_{i,j} = K_e = \int_{T_m} \nabla\varphi_i^T : \sigma_j \cdot dT_m \text{ and} \\ B = \alpha \cdot M + \beta \cdot K \end{cases} \tag{8}$$

where φ_i is the shape function associated with element i, and B is given by the Rayleigh damping.

Using these equations, we use the Euler implicit integration scheme described in [18].

6.3 Large Displacement

If the model undergoes large displacements, the tetrahedral elements of the mesh are subject to rotation, however the cauchy tensor (6) is not invariant to rotation. The matrix K is a linearization of the problem around the initial position. Therefore, using K constraints the use of the model to simulate small displacements only.

Hence we use a corotational model to remove the displacement due to rotation from the total displacement. This model ensure that we keep the deformational displacement only. The method we use, proposed in [19], consists in the decomposition of the displacement gradient ∇u using a polar decomposition for each tetrahedron:

$$\nabla u = QS \tag{9}$$

where Q is an estimation of the element rotation matrix, whereas S is a positive-semidefinite Hermitian matrix. Using this rotation Q, each element stiffness matrix K_e can be updated in order to remove the rotational part of the displacement:

$$K_e^{new} = Q^T \cdot K_e^{old} \cdot Q \tag{10}$$

7 Results

All experiments were implemented on a bi Dual Core 2 Duo 5130. The Figure 4 illustrates the efficiency of theses methods in the case of deformable bodies colliding with a rigid sphere. The time spend to solve Equation 7, *i.e.* coarse matrices updates plus V-Cycle, is presented in Table 2. The stars indicate that an unstructured grids hierarchy has been used.

Table 2. Solving time - 2 Dual Core 5130 2.0 Ghz

Model	#Tetra	#levels	Solving time (ms)
			mat. update + iter.
Kitten	448	2	0.8 + 6.2
Kitten*	432	2	4 + 6.0
Duck*	1148	2	14 + 11.3
Duck	2048	2	8 + 27
Camel	2752	2	24 + 50
Duck*	3456	2	37 + 35
Kitten	3584	3	17 + 37
Duck_2	6080	3	85 + 37

7.1 Discussion

In the case of hierarchically embedded grids, the performances are similar to results in literature [1], with the main difference that due to our hierarchical mesh generator, the geometry of solids simulated is much closer to the desired one. Therefore, the deformations obtained are more realistic and correspond to those of the desired object. Concerning the unstructured grids case, compared to previous methods, solving times are shorter since the coarse matrix updates have been sped-up using our improved method. We can note that the performance gain is more important for bigger meshes. See Table 1. However, with big meshes, more time is spent updating coarse matrix than effectively solving the problem. See Table 2.

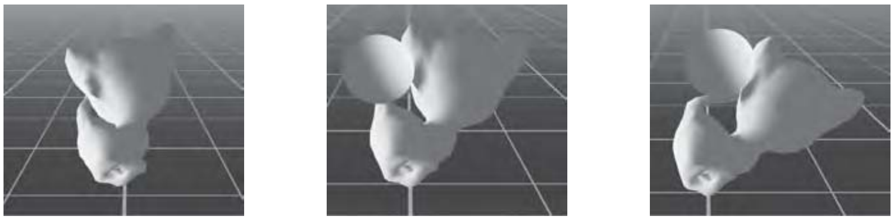

(a) (b) (c)

Fig. 4. (a): Kitten undeformed; (b): Kitten under collision; (c): Kitten under collision

8 Conclusions and Future Work

In this work, we have address two major pitfalls of multigrid solvers, *i.e.* the difficulty for embedded grids to approximate complex geometries, and the slow-down introduced by coarse matrix updates when unstructured grids are used. These issues have been tackled respectively by introducing a novel hierarchical mesh generator and a new method to reduce computation time for matrix updates. These two improvements have been successfully tested on deformable solids simulation using the corotational model.

We have used our hierarchical mesh generator to approximate complex geometries using topologically embedded grid. These grids, allowing fast update of the coarser matrices, have allowed efficient simulation of highly deformable bodies in interactive time.

A 2.0 to 3.0 speed-up factor (See Table 1) has been achieved using our new coarse matrices update method, allowing to interactively simulate deformable solids with thousand tetrahedra using corotational model.

The mesh generator currently manage to build meshes whose geometry is very closed to the target one. However, no warranty concerning the quality of the tetrahedral mesh is provided. Therefore we plan to add constraints on tetrahedra volume and shape to avoid flat tetrahedron. These constraints will be handled using quadratic programming.

References

1. Georgii, J., Westermann, R.: A multigrid framework for real-time simulation of deformable bodies. Computers & Graphics 30, 408–415 (2006)
2. Shi, L., Yu, Y., Bell, N., Feng, W.W.: A fast multigrid algorithm for mesh deformation. In: SIGGRAPH 2006, pp. 1108–1117. ACM Press, New York (2006)
3. Briggs, W.L., Henson, V.E., McCormick, S.F.: A multigrid tutorial. In: Society for Industrial and Applied Mathematics, 2nd edn., Philadelphia, PA, USA (2000)
4. James, D.L., Pai, D.K.: Artdefo: accurate real time deformable objects. In: SIGGRAPH 1999, pp. 65–72. ACM Press/Addison-Wesley Publishing Co., New York (1999)
5. Zhuang, Y., Canny, J.: Real-time and physically realistic simulation of global deformation. In: SIGGRAPH 1999, p. 270. ACM Press, New York (1999)
6. Hauth, M., Etzmuß, O.: A high performance solver for the animation of deformable objects using advanced numerical methods. In: Chalmers, A., Rhyne, T.M. (eds.) Proc. Eurographics 2001. Computer Graphics Forum, vol. 20(3), pp. 319–328 (2001)
7. Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B.: Stable real-time deformations. In: SCA 2002, pp. 49–54. ACM Press, New York (2002)
8. Teran, J., Sifakis, E., Irving, G., Fedkiw, R.: Robust quasistatic finite elements and flesh simulation. In: SCA 2005, pp. 181–190. ACM Press, New York (2005)
9. Hauth, M., Gross, J., Strasser, W.: Interactive physically based solid dynamics. In: SCA 2003, pp. 17–27. Eurographics Association, Aire-la-Ville, Switzerland (2003)
10. Grinspun, E., Krysl, P., Schröder, P.: Charms: a simple framework for adaptive simulation. In: SIGGRAPH 2002, pp. 281–290. ACM Press, New York (2002)

11. Debunne, G., Desbrun, M., Cani, M.P., Barr, A.H.: Dynamic real-time deformations using space & time adaptive sampling. In: SIGGRAPH 2001, pp. 31–36. ACM Press, New York (2001)
12. Wu, X., Downes, M.S., Goktekin, T., Tendick, F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. Comput. Graph. Forum 20 (2001)
13. Capell, S., Green, S., Curless, B., Duchamp, T., Popović, Z.: A multiresolution framework for dynamic deformations. In: SCA 2002, pp. 41–47. ACM Press, New York (2002)
14. Cotin, S., Delingette, H., Ayache, N.: Real-time elastic deformations of soft tissues for surgery simulation. IEEE Transactions on Visualization and Computer Graphics 5, 62–73 (1999)
15. Teran, J., Molino, N., Fedkiw, R., Bridson, R.: Adaptive physics based tetrahedral mesh generation using level sets. Eng. Comput (Lond.) 21, 2–18 (2005)
16. Barbič, J., James, D.L.: Real-time subspace integration for st. venant-kirchhoff deformable models. In: SIGGRAPH 2005, pp. 982–990. ACM Press, New York (2005)
17. Harrison, L.A., Gordon, R.K.: The use of wavelet-like basis functions in the finite element analysis of heterogeneous one dimensional regions. southeastcon 1996 (1996)
18. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: SIGGRAPH 1998, pp. 43–54. ACM Press, New York (1998)
19. Hauth, M., Straßer, W.: Corotational simulation of deformable solids. In: WSCG 2004, pp. 137–145 (2004)

A Novel Optical Tracking Algorithm for Point-Based Projective Invariant Marker Patterns

Manuel Loaiza, Alberto Raposo, and Marcelo Gattass

Tecgraf - Computer Graphics Technology Group
Computer Science Department
Pontifical Catholic University of Rio de Janeiro
R. M. S. Vicente, 225 – Gávea, Rio de Janeiro, RJ - Brazil
{manuel,abraposo,mgattass}@tecgraf.puc-rio.br

Abstract. In this paper, we describe a novel algorithm to group, label, identify and perform optical tracking of marker sets, which are grouped into two specific configurations, and whose projective invariant properties will allow obtaining a unique identification for each predefined marker pattern. These configurations are formed by 4 collinear and 5 coplanar markers. This unique identification is used to correctly recognize various and different marker patterns inside the same tracking area, in real time. The algorithm only needs image coordinates of markers to perform the identification of marker patterns. For grouping the dispersed markers that appear in the image, the algorithm uses a "divide and conquer" strategy to segment the image and give some neighborhood reference among markers.

Keywords: Point-based Feature, Point Set Matching, Projective Invariants, Optical Tracking.

1 Introduction

For many years, tracking systems have been essential and intensively used tools in the implementation of virtual and augmented reality applications. Their purpose is to track different objects or markers, either individually or as a group, within a predefined area. Among the diverse technologies used to develop these tracking devices, optical technologies have been one of the mostly used due to some advantages they have in relation to others, such as: their sensors require no wires, they are less sensitive to noise, and they allow incrementing simultaneously the number of markers tracked within an area without significantly affecting the system, especially hardware.

Optical tracking systems often have a well structured and standardized hardware and software architecture that can be summarized as follows:

Concerning hardware, this type of tracking is usually composed of video capturing devices and different types of markers, which can range from small retro-reflective spheres, IR (infrared) leds, up to features of the scene itself, such as corners, lines and other characteristic details the system might use as tracking markers.

Regarding software, optical tracking systems present a more standardized operation flow based on the implementation of different computer vision techniques with the purpose of extracting, recognizing and tracking markers. Normally image processing is the first technique implemented and has the objective of extracting the 2D coordinates that represent markers' positions within the image captured from the tracking area. Such coordinates constitute the principal information on which the system will operate.

The following stages are the stereo matching of the marker sets appearing in the images (when more than one camera is used) and the 3D reconstruction of the markers based on information about the cameras and the 2D coordinates provided by image processing. After 3D reconstruction, identification of markers generates expensive combinatory operations due to wrong results during matching stage.

The matching stage is usually made with stereo correspondence and epipolar geometry. Due to the lack of a previous standardized identification of markers included in 2D coordinates within the image, false matches and subsequent wrongly reconstructed 3D points are generated. To deal with this problem, other heuristics are required to discard these false markers, such as using 3D metrics to find the correct combination of markers that belong to a specific pattern. These metrics may be implemented based on distances, angles or graphs representing markers distribution.

It is precisely in this stage prior to matching that the algorithm proposed in this paper will be used to reach a way of individually grouping and identifying in 2D the markers sets that form a specific pattern to be tracked. The objective of the proposed algorithm is to be a support tool to reduce the number of cases of false markers generated in the matching stage and subsequently in the 3D reconstruction stage.

This paper is organized as follows: in the next section previous work is discussed. In section 3 the configuration of the hardware system used is presented. In section 4 the theoretical part of computer vision techniques used to implement the algorithm is described, as well as the algorithm's process flow. In section 5 some results are presented, and in section 6 we draw conclusions and future work.

2 Related Work

The marker matching and pattern identification process has been widely investigated in the area of pattern recognition while being used in computer vision. Often, robust and well developed techniques in the pattern recognition area are "exported" to be used in computer vision. In the case of the proposed algorithm, we have used two techniques employed in pattern recognition, here applied in the implementation of marker pattern matching in an optical tracking system. These two techniques on which the algorithm is based are the theory about projective invariant properties described in [3],[8],[10] and the "divide and conquer" strategy exemplified by the use of a quadtree to segment an image where the leaves will be the image coordinates of the markers, which are spread over the image.

The use and efficiency of these techniques have been presented in separate works. The implementation of projective invariant properties for pattern recognition was presented in [11],[12]. Image segmentation based on the "divide and conquer" strategy was presented in [6],[7]. In these works one can see that both techniques have good individual performance, but so far no work presented an integration of these techniques. This was a further motivation behind the present work, which intends to show that these techniques can yield good results by working together.

3 System Setup

To run and test the algorithm, the implementation of a basic optical tracking system was required. It is based on the standard architecture for optical tracking systems, i.e. an architecture based on cameras with infrared light spots and filters that irradiate a scene containing markers covered with retro-reflective material, with the purpose of highlighting the markers in relation to other objects present in the scene. The idea behind this architecture was used in our system, which also uses cameras with infrared filters but without the light spots – instead, the markers are small 5mm and 2mm incandescent lights powered by a set of batteries for each pattern. As was previously defined, our tracking patterns will have a predefined format with the following configurations:

- Pattern I: formed by 4 markers placed in a collinear manner.
- Pattern II: formed by 5 markers placed in a coplanar manner, where one marker will be surrounded by the 4 other markers.

These formats can be seen in Fig. 1 and will be further explained in section 4.3.

4 The Proposed Algorithm

In this section we will briefly summarize the theory behind the techniques used in the implementation of the proposed algorithm, and present a general description of it.

The algorithm's purpose is to group, label, individually identify each pattern, and track it optically in real time, having as operational data only the image coordinates of the markers that compose the tracking patterns. To achieve this goal, the algorithm has two steps: firstly an offline step used for training and generation of a unique identifier to each pattern to be tracked, and secondly an online step that will be the basic optical tracking system, running in real time, used to test the algorithm.

The main techniques used in each step will be presented as follows. Then, the process flow of each step of the algorithm will be presented.

4.1 Image Processing

The image processing used in the algorithm has as main goal analyzing and extracting a 2D representation for each marker displayed on the video images captured by the tracking system cameras. This 2D representation indicates the position of the markers in image coordinates. This sub-process is composed by a set of techniques that analyze the image sequentially, according to the following action flow:

- Capture the video images with the devices used (cameras).
- Convert each image to a single grayscale channel and apply a threshold filter to make it binary.

- Apply a connected-component algorithm to identify circular areas that will be the representations projected on the marker images.
- Extract the center of each connected area as the image coordinate for the marker candidate inside the image.

To implement these techniques in the proposed algorithm, the OpenCV [4] library was employed. This technique is widely used in diverse implementations of optical tracking systems, both commercial [1] and academic [5].

4.2 Quadtree

The quadtree implementation has the purpose of grouping, in quadrants, the total set of markers spread throughout the image. This grouping is based on the principle that markers belonging to the same pattern should be very close to one another. The key step to take advantage of this marker quasi-grouping is reading and traversing the quadtree to extract very close subgroups of markers. This way, candidate subgroups of markers will be formed and tested, and they will contain the tracking patterns inside the tracked image.

The process of traversing the quadtree and generating the test candidate pattern groups follows this action flow:

- Generate a quadtree for each group of markers detected in the image processing stage for each frame captured by the camera.
- Traverse each branch of the quadtree to find pattern I and then pattern II.
    - If the tested node doesn't have a child with less than 4 (or 5 depending of what pattern we are testing) markers then
        - Generate test patterns by combining markers as C^4_n (or C^5_n), where "n" is the number of leaf nodes the parent has, provided that "n" >=4 (or "n" >=5) correspondingly.
    - Else, move down through the 4 leaves of the node running the previous step recursively
- After generating the combinations to compose the marker sets, the test of correspondence between each of these sets against the patterns defined in the training stage is done.
- If some set is correctly matched to a given pattern, then the markers forming the matching group are removed from the quadtree structure.
- The unmatched markers return to the parent node identified as unmatched, and will be part of the same matching process with other unmatched children nodes of the same parent.

As can be seen, matching is a recursive process made at each branch, firstly top-down but becoming bottom-up as the markers are matched in each branch analyzed.

The role of the quadtree is merely as a tool that implicitly provides a neighborhood reference among the markers. An example of this implementation in our system is showed in Fig. 1.

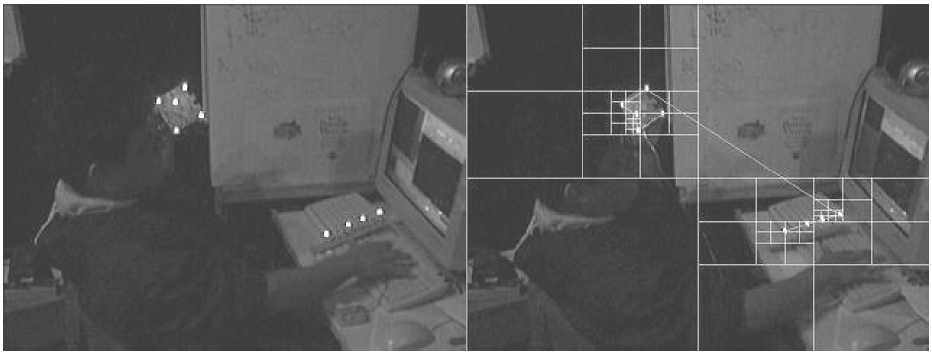


Fig. 1. Grouping for image coordinates of our patterns done with a quadtree

4.3 Projective Invariants

The implementation of projective invariant properties has been discussed especially in the area of pattern recognition [3],[10]. It is based on the invariant property of cross ratio, particularly in the case of perspective projection. The cross ratio property states that if we have 4 collinear points (A,B,C,D) we can define a cross ratio value based on these points' distances according to the following relationship:

$$\text{Cross ratio (A,B,C,D)} = \frac{|AC|/|BC|}{|AD|/|BD|} \qquad (1)$$

This property has been expanded in order to cover not only collinear patterns but also patterns with coplanar markers. In this case, the cross ratio is obtained based on the areas of the triangles generated using combinations of the vertices that constitute a coplanar pattern, as shown in [9]. This extended variation of cross ratio projective invariant properties applied to coplanar points was developed until it became invariant to possible changes in the labels that compose coplanar patterns [2].

In these works, the identifier of the projective invariant for a set of 5 coplanar points is defined as a vector with 5 ranges, with a minimum and maximum value for each vector position. This vector is obtained by applying the P^2-Invariant technique [2] over the 5-point sample, where each vector position is related to each marker in the set. According to [3] this allows not only group matching but also individual matching for each marker in the 5-point set. The only drawback is the need to generate the whole 5-range vector for each candidate group of markers to be matched to a given pattern. In [10] an improvement in relation to [3] was presented, especially to the case of 5 coplanar points. It was demonstrated that the 5-range vector can be reduced to a 2-range vector. In fact, this unique 2-range vector corresponds to the values of the base projective invariants for an especially 5-coplanar pattern that has the following restriction: in the tracked 5-coplanar pattern a specific marker from the group must be well recognized in all views while the pattern is being tracked. This is required because the value of the projective invariant vector is generated based on the cross ratio of the triangles formed by the combination of the vertices, where that specific marker is part of the 4 triangles used to compute the cross ratio. Another overall restriction is that no 3-marker set in a group of 5 be collinear, as this would generate a null triangle area.

Finally, based on the results of the works described above, we have defined the configuration of the patterns used in the algorithm proposed here. Pattern I has two invariant features to perspective projection: collinearity and cross ratio. Pattern II has 4 points around a 5th point, with the need to identify a specific marker in the 5-marker set – in this case, the central marker. These characteristics not only allow computing the value(s) of the properties that are invariant to projection and permutation, but also this specific configuration can be used as filter to quickly discard false candidates.

4.4 Auxiliary Methods

Some auxiliary techniques were implemented to further restrict the selection of marker sets candidate for tracking patterns. These techniques are used as filters that were implemented based on specific characteristics of each type of patterns.

The first one is collinearity test of the 2D coordinates of the markers to be analyzed in a given candidate set. As was already described, each time the algorithm moves down the quadtree branches it will stop at branches containing 4 or more markers (leaves). In the case of pattern I, the algorithm will test first the collinearity of the markers that compound a candidate to be matched. This quick test discards in advance false candidates for pattern I. A second filtering is the matching using the projective invariant property value.

In the case of pattern II, the subgroup of 5 candidate coordinates will be evaluated through the generation of a convex hull. This test is made by checking if 4 coordinates form a convex hull enclosing a fifth coordinate. This restriction is based on the shape defined by the configuration of pattern II, as previously discussed, and used like the first filter to recognize candidates for pattern II.

A second filtering is optional and can be added before matching the projective invariant properties. It consists in creating an oriented and expanded bounding area around each correctly identified pattern in a frame “t”. This tool can be used as a simple way to predict and restrict the area where a well recognized pattern may appear, based on information from frame “t” in frame “t+1”. This second filtering can only be executed after a pattern is identified by matching the projective invariant properties in a previous frame.

4.5 The Algorithm

The proposed algorithm is divided in two stages: training and tracking. In both stages the system presents process flows integrating the basic techniques described in previous sections.

The training stage executes the following flow:

1. Process the image to generate the 2D coordinates representing the markers’ positions in the image.
2. Generate a line in the case of pattern I or a convex hull around the center point in the case of pattern II.

3. Generate the unique identifier defined by the range containing a minimum and maximum value for pattern I, and the identifying vector composed by 2 ranges with minimum and maximum values for pattern II.

The tracking stage flow is composed of:

1. Process the image to generate the 2D coordinates representing the markers' positions in the image.
2. Generate the quadtree for each frame using the 2D coordinates of the markers as data.
3. Create sets of 4 markers using the concept of neighborhood provided by the quadtree based on the position of the 2D coordinates of the markers. These sets will be candidates for patterns of the tracking system.
4. Discard some candidate sets by running the collinearity test over the 4-marker sets.
5. For each marker set, generate the respective identifier, for pattern I.
6. Compare the values of the identifier generated for each candidate set tested against the pattern I values predefined in the training stage.
7. Remove from the quadtree each marker set, which is correctly matched with some specific pattern I.
8. With the remaining markers, create sets of 5 markers using again the concept of neighborhood provided by the quadtree structure.
9. Discard some candidates' sets by running the first filter to coplanar pattern.
10. For each marker set, generate the respective identifier, for pattern II.
11. Compare the values of the identifier generated for each candidate set tested against the pattern II values predefined in the training stage.
12. Label the matched and recognized pattern sets as one of the training patterns, in order to track them frame by frame.

Once the tracking stage is under execution identifying each pattern individually, we can insert two sub-processes to help the process of discarding false candidates:

- In step 12, once the pattern is identified, we can generate a 2D extended bounding box around markers that compose each pattern. This is helpful to predict where the pattern will appear in the next frame.
- In steps 4 and 9, test whether the candidates' sets that satisfy the collinearity tests are inside the 2D bounding boxes created in a previous frame for each respective pattern.

5 Results

To test the algorithm, we used the basic tracking system with a single camera and used the algorithm to track a set of 2 collinear patterns of type I, and 2 coplanar patterns of type II (Fig. 2). The goal of the test was to measure the efficiency of the algorithm in a continuous tracking of the patterns and to show its robustness in the discard of false candidates during the matching stage.

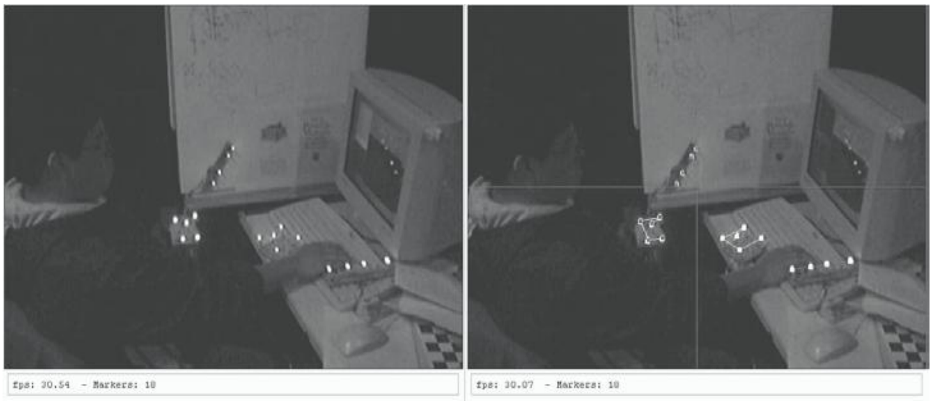


Fig. 2. Optical tracking test with two patterns of each type in the image

The first step was to make the individual training for each pattern to generate a unique identification, based on the projective and permutations invariant values calculated for each pattern. This training was done moving each pattern, one per time, in front of the camera for no more than 2 minutes. For each frame captured, the projective and permutation invariant values of the pattern were calculated and stored. After these 2 minutes we had a sample of approximately 2x60x30 = 3600 values. Later, this sample was analyzed to get a range of values for the projective invariant values for pattern I and two ranges for pattern II (Table 1).

Table 1. Projective invariant values for each pattern

Pattern	Type	Minimum Value	Maximum Value
1st-Pattern	I	[2.048]	[2.085]
2nd-Patern	I	[2.368]	[2.460]
3rd-Pattern	II	[0.06 , 0.06]	[0.14 , 0.13]
4th-Pattern	II	[0.16 , 0.16]	[0.23 , 0.24]

We can observe that the range of values that have the same pattern type are disjoints, meaning the patterns with same configuration will be identified separately. In Tables 2 and 3 we show some values produced in the process of recognizing and tracking the patterns in the case showed in Fig. 2.

Table 2. Test in each quadrant for getting candidates for pattern I

Quadrant	Nº Markers	Nº Candidates for Pattern I
Left – Up	4	$C^4_4 = 1$
Left – Down	5	$C^4_5 = 5$
Right – Up	0	0
Right – Down	9	$(C^4_6 =)$ 15 + $(C^4_9 =)$ 126 = 141
Total:	18	147

Table 3. Filters applied over candidates for pattern I

Quadrant	Candidates Pattern I	Collinearity Test	Projective Inv. Match	Pattern ID	Unmatched Markers
Left – Up	1	1	1	1st-Pattern	0
Left – Down	5	0	0	0	5
Right – Up	0	0	0	0	0
Right – Down	141	0 + 1 = 1	1	2nd-Patern	5
Total:	147	2	2	2 (recognized)	10

In Tables 4 and 5 we show the following process of recognizing coplanar patterns.

Table 4. Test in each quadrant for getting candidates for pattern II

Quadrant	Nº Markers	Nº Candidates for Pattern II
Left – Up	0	0
Left – Down	5	$C^5_5 = 1$
Right – Up	0	0
Right – Down	5	$C^5_5 = 1$
Total:	10	2

Table 5. Filters applied over candidates for pattern II

Quadrant	Candidates Pattern II	Convex Hull Test	Projective Inv. Match	Pattern ID	Unmatched Markers
Left – Up	0	0	0	4th-Pattern	0
Left – Down	1	1	1	0	0
Right – Up	0	0	0	0	0
Right – Down	1	1	1	3rd-Pattern	0
Total:	2	2	2	2 (recognized)	0

The strategy to remove from the quadtree the markers that were correctly matched with pattern I reduces the number of combination cases to test in the next process of recognizing the coplanar patterns. A key point in the execution of the algorithm was the implementation of an efficient routine to make the collinearity test of the candidates. For example, the case of quadrant Right-Down in Table 3 where 141 candidates are reduced to only 1 valid candidate.

Other important point is the robust sample used to train the patterns, which helped us to get a range of projective and permutation invariant values that are very flexible and good to discard false candidates after the collinear and convex hull filters. Normally, these false candidates appear as a consequence of the excessive number of possibilities generated by the random combination of markers that appear in a specific branch of our quadtree.

Finally, the tracking system was executed in a PC Pentium IV, 2.5 GHz, 2GB RAM, using a simple webcam at 30fps. In the execution of the system using the algorithm, the average frame rate was 28±2 fps.

6 Conclusions and Future Works

In this paper a novel algorithm for optical tracking of points-based patterns was presented, with the goal of making a fast matching and individual identification of predefined patterns. An interesting point of the matching process used is that all processes are implemented in a processing 2D stage only, differently from other optical tracking systems, which generally need to reconstruct the 3D position of the markers set to provide a correct matching.

As future projects, there is still space for new optimizations in the filters to discard false candidates, especially in the case of coplanar patterns. Another application area is the use of the tracking algorithm for markerless tracking. In this case, specific points, lines or textures of real objects are used as tracking markers. Since these features can have projection invariant characteristics, there is room for the use of the proposed algorithm.

References

1. Advanced Real Time Tracking GmbH, A.R.T. System. (2006), http://www.artracking.de/
2. Meer, P., Lenz, R., Ramakrishna, S.: Correspondence of Coplanar Features Through p2 Invariant Representations. In: Applications of Invariance in Computer Vision, pp. 473–492. Springer, Heidelberg (1993)
3. Meer, P., Lenz, R., Ramakrishna, S.: Efficient Invariant Representations. International Journal of Computer Vision 26, 137–152 (1998)
4. Intel Open Source Computer Vision Library, http://www.intel.com/technology/computing/opencv/
5. Ribo, M., Pinz, A., Fuhrmann, A.: A New Optical Tracking System for Virtual and Augmented Reality Applications. In: Proceedings of the IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary, vol. 3, pp. 1932–1936 (2001)
6. Santos, P., Stork, A., Buaes, A., Jorge, J.: Innovative Geometric Pose Reconstruction for Marker-based Single Camera Tracking. In: Proc. of ACM SIGGRAPH Int. Conf. on Virtual Reality Continuum and its Applications, pp. 237–244. ACM Press, New York (2006)
7. Santos, P., Stork, A., Buaes, A., Jorge, J.: PTrack: Introducing a Novel Iterative Geometric Pose Estimation for a Marker-based Single Camera Tracking System. In: Proceedings of IEEE Virtual Reality, pp. 143–150. IEEE Press, California, USA (2006)
8. Smit, F.A., Van Rhijn, A.J, Van Liere, R.: A Topology Projection Invariant Optical Tracker. In: Proceedings of the Eurographics Symposium on Virtual Environments, Lisbon, Portugal, pp. 63–70 (2006)
9. Suk, T., Flusser, J.: The features for recognition of projectively deformed point sets. In: Proc. of the IEEE International Conference on Image Processing, pp. 348–351. IEEE Press, Washington (1995)
10. Suk, T., Flusser, J.: Point-based projective invariants. Pattern. Recognition (33), 251–261 (2000)
11. Tsonis, V.S., Chandrinos, K.V., Trahanias, P.E.: Landmark-based navigation using projective invariants. In: Proc. of International Conference Intelligent Robots and Systems, pp. 342–347 (1998)
12. Van Liere, R., Mulder, J.D.: Optical Tracking Using Projective Invariant Marker Pattern Properties. In: Proc. of IEEE Virtual Reality, pp. 191–198. IEEE Press, Los Angeles (2003)

Automatic Subcortical Structure Segmentation Using Probabilistic Atlas

Jundong Liu[1], David Chelberg[1], Charles Smith[2], and Hima Chebrolu[2]

[1] School of Elec. Eng. & Comp. Sci.
Ohio University
Athens, OH
[2] Department of Neurology
University of Kentucky
Lexington KY

Abstract. Automatic segmentation of sub-cortical structures has great use in studying various neurodegentative diseases. In this paper, we propose a fully automatic solution to this problem through the utilization of a distribution atlas built from a set of training MR images. Our model consists of two major components: a local likelihood based active contour (LLAC) model and a guiding probabilistic atlas. The former has a very strong ability in standing out the structures that are in low contrast with the surrounding tissues. The latter has the functionality of defining and leading the segmentation procedure to capture the structure of interest. Formulated under the maximum a posterior framework, probabilistic atlas for the structure of interest, e.g. caudate, putamen, can be seamlessly integrated into the level set evolution procedure, and no thresholding step is needed for capturing the target.

1 Introduction

To accurately measure the volumetric and morphological changes of the subcortical structures is a very crucial step in diagnosing various neurodegenerative diseases, including Alzheimer's Disease (AD), Parkinson's Disease (PD) and Multiple Sclerosis (MS). However, to automatically segment those structures is a challenging task. Many undesired image characteristics, including intensity overlapping among different structures, partial volume effects and image noise, all contribute to the difficulty of this process.

A number of methods have been proposed in recent years for sub-cortical structures segmentation. Sonka *et. al.* [14] use a deformable model, where segmentation is achieve by deforming a template to match the structure of interest. This method, however, is sensitive to the initialization and usually requires expert intervention and guidance. The solution proposed Fischl *et. al.* [8] utilizes a spatially varying Gaussian mixture model to integrate prior information into the classification procedure. A probabilistic atlas that obtained from a set of manually labeled training images helps define the structure of interest. Shape-based prior has been explored in several works [9,5,17], and great robustness has been shown due to the use of PCA dimension reduction.

Another group of solutions [19,11,16] involve combining certain registration technique with an atlas. The final segmentation is obtained by aligning the previously segmented template with the subject. Atlases, mainly probabilistic distribution-based, play a crucial role in securing the robustness and accuracy of the segmentation results. However, under these atlas-registration-based models, the segmentation performance is often largely influenced by the accuracy of the non-rigid registration mechanism employed. Despite all these efforts, the subcortical structure segmentation problem is still far from being solved. There is a need to future investigate into novel techniques that can automatically detect the subcortical structures more accurately.

This paper proposes a fully automatic subcortical structure segmentation solution that utilizes a distribution atlas built from a set of training MR images. The models consists of two major components: a local likelihood based active contour (LLAC) model and a guiding probabilistic atlas. The former shows [10] a very strong ability in standing out the structures in low contrast with the surrounding tissues. The latter, which is created from a set of manually labeled training image, has the functionality of defining and leading the segmentation procedure to capture the structure of interest.

Formulated under Bayesian a posterior probability framework, our LLAC model can seamlessly integrate the probabilistic atlas information into the level set evolution procedure, and no thresholding step, as in many other sub-cortical structure segmentation methods, is required to generate the final segmentation. In addition, it relaxes the global Gaussian assumption in many region-based actively models (piecewise constant is the degenerate case) from "global" to "local", and *local means* are used as the area representatives. Being able to better account for intensity inhomogeneity, the LLAC model can stand out the subcortical structures that have low contrast with the surrounding tissues.

2 Methods

For simplicity, discussions in this section will be conducted with 2D notation, but all the conclusions hold for 3D cases. Let C be an evolving curve in Ω. C_{in} denotes the region enclosed by C and C_{out} denotes the region outside of C. The Chan-Vese (two-phase) piecewise-constant model is to minimize the energy functional

$$F(c_1, c_2, C) = \mu \cdot \text{Length}(C) + \lambda_1 \int_{C_{in}} |u_0 - c_1|^2 dxdy + \lambda_2 \int_{C_{out}} |u_0 - c_2|^2 dxdy$$

where c_1 and c_2 are the averages of u_0 inside C and outside C respectively.

This model, together with many region based active contour models, uses Gaussian distribution to model the intensity values within a same class. However, global Gaussian is rarely an accurate depiction of local image profile for many medical images, especially for the gray matter in the subcortical area. Negligence of local information would often result in undesired segmentations.

Piecewise-smooth models [2,18] provide a solution for the intensity variability problem. Gradual intensity changes can be handled with [2,18], however, high computational cost and being sensitive to curve initialization pose a barrier for practical applications.

2.1 Our Local Likelihood-Based Model

Let $S = \{in, out\}$ be the two classes for a two-phase model. The probability of the pixel (x, y) belonging to in and out is denoted by $P(in|(x, y))$ and $P(out|(x, y))$ respectively. Let $Pr(in)$ and $Pr(out)$ be the class prior probabilities at (x, y). Then,

$$P(in|(x,y)) = \frac{Pr(in_{(x,y)})P(u_0(x,y)|in)}{P(B)} \quad P(out|(x,y)) = \frac{Pr(out_{(x,y)})P(u_0(x,y)|out)}{P(B)} \tag{1}$$

where $P(u_0(x, y)|in)$ is the likelihood of a voxel in class in has the intensity of $u_0(x, y)$. $P(B)$ is a constant. The maximum a posterior segmentation would be achieved only if the multiplication of $P(in|(x; y))$ and $P(out|(x; y))$ all over the entire image domain is maximized. Taking a logarithm, the maximization can be reduced to the minimization of the following energy:

$$\begin{aligned} F(C) = \mu \cdot \text{Length}(C) - \int_{C_{in}} \log(Pr(in)P(u_0(x,y)|in))dxdy \\ - \int_{C_{out}} \log(Pr(out)P(u_0(x,y)|in))dxdy \end{aligned}$$

Note that our overall model is similar to [12,13], but the setup of the likelihood term is different, which will be explained next.

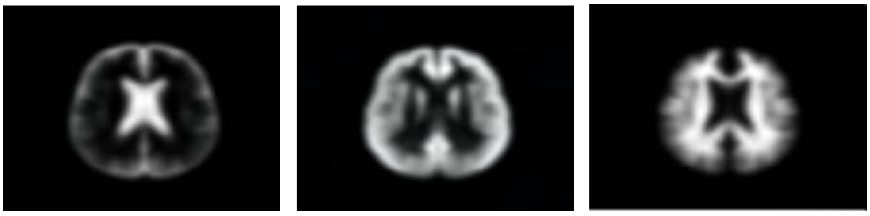


Fig. 1. Spatial prior probability images of CSF, GM and WM

Spatial distribution priors for whole brain segmentation: $Pr(in)$ and $Pr(out)$: In this paper, the distribution prior image we use is provided by the Montreal Neurological Institute (MNI) [6]. MNI prior is made of three probability images, corresponding to GM, WM and CSF respectively. Each image contains values in the range of zero to one, representing the prior probability of a voxel being either GM, WM or CSF after an image has been normalized to the same space (see Figure 1). In this paper, the sub-cortical structures we try to extract belong to GM, therefore we take the GM and WM prior images as $Pr(in)$

and $Pr(out)$ respectively. For these prior images to be applied, a registration is need to align the prior and the input image. We used the affine registration routine provided by SPM [15] in all the 3D experiments of this paper.

Spatial distribution prior for Caudate segmentation: The distribution prior used in this paper is constructed with 18 T1-weighted MR image data downloaded from internet brain segmentation repository (IBSR) at Mass General Hospital. Each data set contains a whole brain MRI together with an expert manual segmentation of 43 individual structures (1.5mm slice thickness). In this paper, we choose **caudate** as the structure of interest, and the proposed method would, in principle, work for other sub-cortical structures as well.

Out of the 18 data sets, the first 9 have been used for constructing the distribution atlas. The other 9 are used as testing cases to evaluate the accuracy of our s segmentation model. To construct a distribution atlas, the 9 caudate segmentations need to put into a standard space. The template brain provided by SPM2 [15], obtained based on 152 brains from Montreal Neurological Institute, has been used as the standard space.

Let $f_i (1 \leq i \leq N = 9)$ be one of the training images, and the extracted caudate segmentation is denoted as $s_i (1 \leq i \leq N)$. Let r denote the standard temple. The probabilistic caudate atlas was constructed as follows

1. For each training image f_i in the IBSR data sets, map it to the standard template r using SPM2's normalization routine. A 12-parameters affine transformation is estimated first, followed by a nonlinear warping based on a linear combination of discrete cosine transform (DCT) basis functions. The resulting transformation is denoted as T_i.
2. Apply T_i to s_i to get a transformed caudate segmentation s'_i.
3. Sum up s'_i under the standard space to get ss'.
4. The prior distribution is then obtained: $Pr(caudate) = \frac{ss'}{N}$, and $Pr(non\text{-}caudate) = 1 - P(caudate)$.

When we try to segment the caudate of a testing image k, an affine transformation from the standard template r to k is estimated using SPM2. Then the obtained transformation is applied to the distribution atlas $Pr'(Caudate)$ and $Pr'(nonCaudate)$ to put the prior images on the aligned space with the testing image k.

Spatial distribution prior – (Caudate + Putamen) atlas: To explore the effect and compare the results of segmenting individual vs. multiple structures together, we also construct the distribution prior for the combination of caudate and putamen, using the same routine presented above. Figure 2 shows a zoom-in version of the atlas (distribution prior image) constructed from the 9 IBSR data sets, viewing from three different axes.

Likelihood terms $P(u_0(x, y)|in)$ and $P(u_0(x, y)|out)$: global Gaussians are commonly assumed in many region-based active contour models to model the

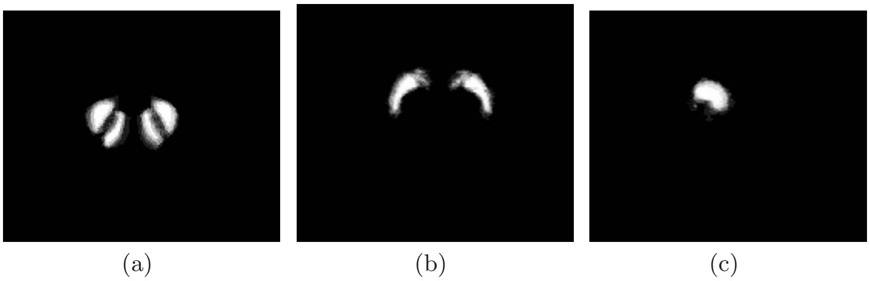


Fig. 2. Caudate + Putamen atalas: viewing from three axes. This atlas is constructed from 9 IBSR data sets.

intensity distribution, but they are often not an accurate description of the local image profile, especially when intensity inhomogeneity is present. A remedy is to relax the global Gaussian mixture assumption and take local intensity variations into consideration. More specifically, local Gaussians (piecewise-constant is the degenerate case) should be used as a better approximation to model the vicinity of each voxel.

In the Chan-Vese model, two global means c_1 and c_2 are computed for C_{in} and C_{out}. In our approach, we introduce two functions $v_1(x, y)$ and $v_2(x, y)$, both defined on the image domain, to represent the mean values of the *local* pixels inside and outside the moving curve. By *Local*, we mean that only neighboring pixels will be considered. A simple implementation of the "neighborhood" is to introduce a rectangular window $W(x, y)$ with size of $2k + 1$ by $2k + 1$, where k is a constant integer. Therefore,

$$v_1(x, y) = \text{mean}(u_0 \in (C_{in} \cap W(x, y)))$$
$$v_2(x, y) = \text{mean}(u_0 \in (C_{out} \cap W(x, y)))$$

With the new setup, our segmentation model can then be updated as a minimization of the following energy:

$$F(v_1, v_2, C) = \mu \cdot \text{Length}(C) - \int_{C_{in}} \left(\log(Pr(in)) - \log(\sigma_1) - \frac{(u_0 - v_1)^2}{2\sigma_1^2} \right) dxdy - \int_{C_{out}} \left(\log(Pr(out)) - \log(\sigma_2) - \frac{(u_0 - v_2)^2}{2\sigma_2^2} \right) dxdy$$

The variances σ_1 and σ_2 should also be defined and estimated locally. However, due to the fact that local variance estimation tends to be very unstable, we use global variances (for the pixels in C_{in} and C_{out}) as uniform approximation.

2.2 Level Set Framework and Gradient Flow

Using the Heavside function H and the one-dimensional Dirac measure δ [1], the energy function $F(v_1, v_2, C)$ can be minimized under the level set framework, where the update will be conducted on the level set function ϕ. Let $PrC = Pr'(Caudate)$ and $PrNC = Pr'(non - Caudate)$. Parameterizing the descent direction by an artificial time $t \geq 0$, the gradient flow for $\phi(t, x, y)$ is given from the associated Euler-Lagrange equation as

$$\frac{\partial \phi}{\partial t} = \text{sign}(v_1 - v_2) \cdot \delta(\phi) \left[\mu \text{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right) - \log \frac{PrC}{PrNC} + \log \frac{\sigma_1}{\sigma_2} - \left(\frac{(u_0 - v_1)^2}{2\sigma_1^2} - \frac{(u_0 - v_2)^2}{2\sigma_2^2} \right) \right] \quad (2)$$

where ϕ_0 is the level set function of the initial contour. This gradient flow is the evolution equation of the level set function of our proposed method.

Correspondingly, v_1 and v_2 are computed with

$$v_1 = \frac{(u_0 * H(\phi)) \otimes W}{H(\phi) \otimes W} \qquad v_2 = \frac{(u_0 * (1 - H(\phi))) \otimes W}{(1 - H(\phi)) \otimes W} \quad (3)$$

where $\otimes$ is the convolution operator. One should note that, Chan-Vese model can be regarded as a special case of our model — when the window W is set to infinitely large.

The sign(v_1 - v_2) term in Eqn.2 is designed to avoid the occurrence of an undesired curve evolution phenomenon we named *local twist*. For more details, we refer the readers to [10].

In practice, the Heaviside function H and Dirac function δ in eqn. 2 have to be approximated by smoothed versions. We adopt the $H_{2,\epsilon}$ and $\delta_{2,\epsilon}$ used in [1]. For all the experiments conducted in this paper, we set the size of the window W as 21×21.

3 Results and Discussions

Two groups of experiments were conducted using the IBSR data sets. The first 9 segmentations were used as the training sets, and the structures of interest in group one and group two are caudate, and caudate + putamen respectively.

The first group of experiments were to capture the cauduates of the rest 9 IBSR data sets. To assess the performance of our algorithm we computed the Dice coefficients between the segmentation obtained from our algorithm and that of the ground truth. This metric measures the similarity of two sets and ranges from 0 for sets that are disjoint to 1 for sets that are identical. This index is a special case of the kappa index that is used for comparing set similarity. The Dice coefficient is defined as:

$$K(S_1, S_2) = \frac{2 \times |S_1 \cap S_2|}{|S_1 \cup S_2|} \quad (4)$$

Table 1 shows the results of the Dice coefficients for all test cases. The results are rather stable across the 9 data sets, with an average Dice value of 0.7466. The accuracy is comparable to the results reported in [19].

Table 1. Caudate segmentation results: Dice coefficients for all the 9 test cases. The summary rows at the end of the table display the overall average.

IBSR Datasets	Dice Coefficient
Patient 10	0.7611
Patient 11	0.7929
Patient 12	0.7039
Patient 13	0.6928
Patient 14	0.8019
Patient 15	0.7342
Patient 16	0.7503
Patient 17	0.6622
Patient 18	0.8201
Average	0.7466

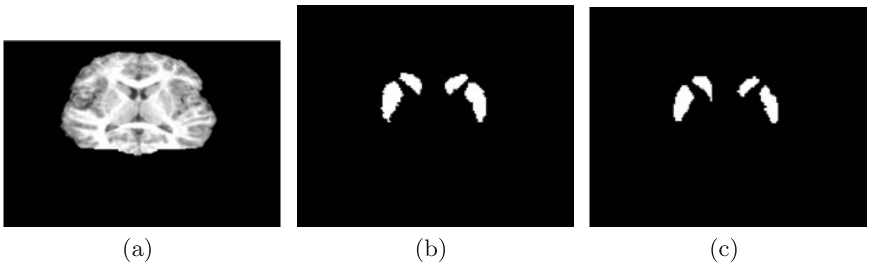


Fig. 3. A (Caudate + Putamen) segmentation example. (a) is an intensity image; (b) is the ground truth manual segmentation for Caudate + Putamen; (c) the segmentation result from this our method.

The second group of experiments aim to explore the benefit of segmenting multiple structures together. Caudate and putamen are connected at the bottom of the structures. Treating them as a single structure is expected to boost the segmentation accuracy. The segmentation results obtained from our experiments provide support to this notion. The same 9 IBSR images have been used, and the Dice coefficients obtained show a constant increase over all data sets. The average Dice value is 0.7868.

Program running time: The algorithm presented in this paper is implemented with Matlab 7 R14, under an IBM Thinkpad T60 laptop (Intel Core Duo T2300 / 1.66 GHz processor, 2.0GB of RAM). The average running time for BWH data is around 15 minutes per set. The level set update routine is impemented purely using Matlab m-files so far. We expect a significant reduce in the whole running time after the core level set part is implemented with mex-C.

Table 2. Caudate + putamen segmentation results: Dice coefficients for the 9 IBSR test cases

IBSR Datasets	Dice Coefficient
Patient 10	0.7923
Patient 11	0.8243
Patient 12	0.7502
Patient 13	0.7394
Patient 14	0.8423
Patient 15	0.7793
Patient 16	0.7930
Patient 17	0.7201
Patient 18	0.8405
Average	0.7868

4 Conclusions

In this paper, we propose a subcortical structure segmentation algorithm based on a local likelihood oriented active contour model. The LLAC model has the advantage of being able to stand out the brain structures that are with low contrast with the surrounding tissues. The probabilistic atlas essentially works as a mask to capture the structure of interest, where no thresholding step and value are needed. The accuracy of our model may be further boosted if shape-based atlas, constructed through PCA, is integrated into the level set framework.

References

1. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. on Image Processing 10(2), 266–277 (2001)
2. Chan, T.F., Vese, L.A.: A level set algorithm for minimizing the Mumford-Shah functional in image processing. In: 1st IEEE Workshop on Variational and Level Set Methods in Computer Vision, pp. 161–168 (2001)
3. Cocosco, C.A., et al.: BrainWeb: Online interface to a 3D MRI simulated brain database, Neuroimage, 5(4), (part 2/4 S245) (1997)
4. Cremers, D., Rousson, M., Deriche, R.: A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. IJCV (to appear, 2006)
5. Yang, J., Tagare, H., Staib, L.H., Duncan, J.S.: Segmentation of 3D Deformable Objects with Level Set Based Prior Models. ISBI, 85–88 (2004)
6. Evans, A.C., Collins, D.L., Milner, B.: An MRI-based stereotactic atlas from 250 young normal subjects. Society of Neuroscience Abstrasts 18, 408 (1992)
7. Zhang, Y., Brady, M., Smith, S.: Segmentation of brain MR images through a hidden Markov random field model and the expectation maximization algorithm. IEEE Trans. on Medical Imaging 20(1), 45–57 (2001)
8. Fischel, B., et al.: Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. Neuron 33, 341–355 (2002)

9. Leventon, M., Grimson, W., Faugeras, O.: Statistical shape influence in geodesic active contours. CVPR 2000 1, 316–323 (2000)
10. Liu, J., Chelberg, D., Smith, C., Chebrolu, H.: Distribution-based Level Set Model for Medical Image Segmentation. In: BMVC 2007. British Machine Vision Conference, Warwick, UK, 10-13 September (2007)
11. Pohl, K., Bouix, S., Kikinis, R., Grimson, W.: Anatomical guided segmentation with non-stationary tissue class distributions in an expectation-maximization framework. In: ISBI 2004, pp. 81–84 (2004)
12. Paragios, N., Deriche, R.: Coupled Geodesic Active Regions for Image Segmentation: A Level Set Approach. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 224–240. Springer, Heidelberg (2000)
13. Rousson, M., Deriche, R.: A Variational Framework for Active and Adaptative Segmentation of Vector Valued Images, INRIA Technical Report (2002)
14. Sonka, M., Fitzpatrick, J.M.: Handbook of medical imaging, vol. 1,2, pp. 69–211. SPIE Press (2000)
15. Mechelli, A., Price, C.J., Friston, K.J., Ashburner, J.: Voxel-Based Morphometry of the Human Brain: Methods and Applications. In: Current Medical Imaging Reviews, pp. 105–113 (2005)
16. Gouttard, S., Styner, M., Joshi, S., Smith, R.G., Cody, H., Gerig, G.: Subcortical Structure Segmentation using probabilistic Atlas Priors. In: SPIE 2007, vol. 6512, p. 65122J (2007)
17. Tsai, A., Yezzi, A., Wells, W., Tempany, C., Tucker, D., Fan, A., Grimson, E., Willsky, A.: A Shape-Based Approach to Curve Evolution for Segmentation of Medical Imagery. IEEE TMI 22(2), 137–154 (2003)
18. Tsai, A., Yezzi, A., Willsky, A.: A Curve Evolution Approach to Smoothing and Segmentation Using the Mumford-Shah Functional. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2000)
19. Zhou, J., Rajapakse, J.C.: Segmentation of subcortical brain structures using fuzzy templates. NeuroImage 28, 915–924 (2005)

Integrative Geometric-Hashing Approaches to Binding Site Modeling and Ligand-Protein Interaction Prediction

Joanna Lipinski-Kruszka[1] and Rahul Singh[2]

[1] Department of Biology
[2] Department of Computer Science, San Francisco State University,
San Francisco, CA 94132

Abstract. The function of a protein is dependent on whether and how it can interact with various ligands. Therefore, an accurate prediction of protein-ligand interactions is paramount to understanding proteins' biological mechanisms and hence to the development of therapeutic agents. A ligand is most likely to bind in the largest pocket on the surface of the protein. Moreover, it requires that the pocket meets certain structural and geometric criteria that allow the ligand to "anchor" in place by forming stabilizing interactions with the protein. Based on this logic, many geometry-based algorithms have been developed to predict protein-ligand interactions. Here we investigate a geometric-hashing based algorithm – to see how well it distinguishes proteins that do and do not bind a ligand, and propose enhancements that improve its robustness. We also introduce an alternative way of integrating geometric and biochemical properties of multiple binding mechanisms into a single representation.

1 Introduction

Delivering a drug to market is a very lengthy process [3], much of which is spent in labs experimenting to find compounds that have good efficacy for the targeted protein. If this screening process could be expedited, drug development cost and time to market could be significantly reduced. This need has fueled the development of new computational approaches aimed at performing accurate virtual screening. A critical problem in this context is to correctly dock a ligand onto a receptor surface and involves both geometric and physicochemical reasoning.

Computational solutions to this problem are based on the postulate that one of the key requirements for binding is that the ligand has to be "anchored" in a small pocket on the surface of the protein, called the active site, through hydrogen or covalent bonds, or interactions such as hydrophobic or hydrophilic ones. Figure 1(c) illustrates an example of stabilization via hydrogen bonds. These stabilizing bonds and interactions can be formed only with the nearest atoms at specific locations. For this reason, the spatial distribution of atoms is thought to play a key role in ligand-protein binding, and hence suggests that the geometric definition of an active site is a critical and necessary [12] component in determining the protein's affinity for a ligand.

At the state-of-the-art, techniques that use geometry to screen for similarities between active sites can be broadly divided into two categories [12]: those that characterize general properties of an active site, and those that find sites that resemble an

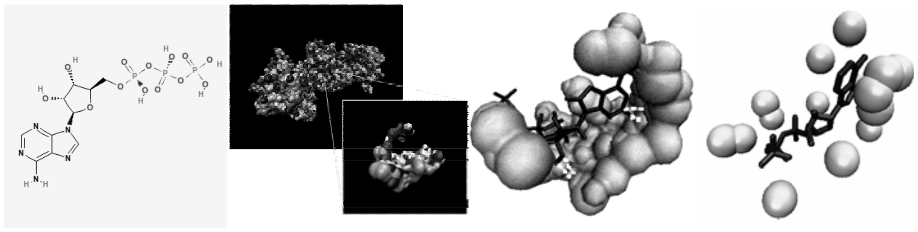


Fig. 1. (a) Structure of Adenosine Triphosphate (ATP). **(b)** A protein (PDB ID: 2HIX) and its active site. The close-up shows only atoms that are within 5 Å of the ligand (ligand not shown). **(c)** Example of ligand-protein interactions. ATP shown in black, active site of a protein in gray, some of the stabilizing hydrogen-bonds are shown as white, broken lines. **(d)** Model of ATP active site. Spheres represent atoms that make-up the model; ATP is shown as thick, dark lines.

already known active site. Here we focus on the latter, by considering approaches based on geometric hashing, an algorithm adopted from the field of computer vision, that has found significant applicability in this context.

Geometric hashing uses a model of a known active site to search for similar ones in other target proteins. One of the strengths of this approach is its ability to deal with noisy or incomplete data. One of its major drawbacks is its over-specificity due to a strong bias towards the geometry and physical properties of a single protein underlying the model. To overcome this limitation, we present an alternative that is based on robust, example-driven characterization of the binding site geometry. The proposed method starts by analyzing structurally diverse molecules which share their ability to bind a specific ligand but might differ in their mechanisms of interaction with it. Next, a voting-mechanism is used to extract from their binding sites characteristics that are common to multiple such proteins. Lastly, these features are integrated into a single arrangement of atoms representing all of the inspected binding mechanisms. This method not only limits the model's favoritism for any particular binding mechanism but, also, ensures that the pocket definition is deeply grounded and takes into account implicit geometric, physical, and chemical factors involved in the binding.

With this improved model, we then move on to investigate whether we can further improve on the robustness of geometric hashing for prediction of ligand-protein interactions. Because this approach can produce many possible solutions, careful scoring of "goodness" of matching is of key importance. Scoring is very sensitive to whether more emphasis is placed on the number of good matches or on their quality. How to choose the balance between these metrics depends on the data set. The former approach, which gives more weight to tightness of the fit, is more feasible if one wants to find partial regions that have an exact or very close match between them since it does a good job dealing with partial data; the latter approach is better if one wants to find the largest possible areas of similarity. Here we describe the pros and cons of a few approaches to finding the best possible balance between the two, and their biological significance.

2 Proposed Methods

Based on the geometric hashing algorithm, we created a model for an active site of Adenosine Triphosphate (ATP) (figure 1.a), a ubiquitous molecule playing a key role in intracellular transfer of energy. ATP was chosen because of the availability of structural data of proteins that are known to bind it at known locations [9].

We used unrelated, ATP-binding proteins to create our model and to evaluate our scoring methods. Using the Visual Molecular Dynamics (VMD) [11] tool, each of these proteins was stripped down to atoms within 3-5 Å from the bound ligand (figure 1.b). The rest of the atoms were ignored because they were assumed to be too far from the active site to significantly impact binding.

2.1 Base Creation and Transformation

Following the first step of geometric hashing, feature points were extracted; in this case, atoms of an active site were used for this purpose. Then, similarity between two binding sites – a model and a target – was investigated by comparing all possible transformations in 3-dimenssional space of one to that of the other. To do that, all possible permutations of three distinct atoms were found and used to create triplets to form triangular bases in 3D space. For each of the bases, transformation was done so that its vertices were placed: at the origin (0,0,0), on the X axis (x, 0, 0), and in the XY plane (x,y,0). The transform was then applied to all other atoms of the active site.

Every base of one protein was tested against all bases of another to find their best-fitting spatial confirmations. First, bases were tested for match. Two bases matched if after transformation onto the coordinate system they had: (1) the same atom types at the corresponding vertices, and (2) spatial location of their corresponding vertices within some specified threshold (set to 1 Å). If these conditions were met, transformations of the two proteins with respect to these bases were compared. Whenever atoms of the same type from the two proteins translated to approximately the same spatial location (also set to 1Å from each other) these atoms were considered matching. These matches were then used to evaluate the quality of the fit between the two bases.

2.2 Finding Best Fitting Bases

We evaluated four methods of scoring the fit between bases:

Overall Root Mean Square Deviation (RMSD): In this approach, after both proteins have been transformed, for every atom of the model protein a closest matching atom of the target protein was found. The distance between each atom pairs was used to compute the overall RMSD.

Top 80% RMSD: Our second approach, similarly to overall RMSD, found for every atom of one protein a closest atom in the other. This time, however, instead of taking all distances to compute the score, we only considered the top 80% best atom pairings. The worst scoring 20% were ignored.

RMSD of Matched Atoms: The third method computed RMSD for just atoms that matched. If a transformation found a close fit (noise threshold 1Å) between two atoms of the two proteins, the atom pair was included in the calculation of RMSD. Atoms that did not have a match within the allowable threshold were ignored. This method also required a minimum of 5 matching atoms.

Most Matched Atoms: Our last evaluation method gave best scores to transformations that resulted in the highest number of matched atoms and which had a global RMSD under 1.

2.3 Computation of the Binding Site Model

We used nine different ATP-binding proteins (2HIX, 2J9L, 2OOY, 2HF4, 2EWW, 2HVY, 2F43, 1Y64, and 1MO8) to build our model, which was created by finding the best fit of atoms of the active site of the template protein (2HIX) and of each of the other eight target proteins. Each best fit would cast a vote on atoms of the template that were matched to atoms of the target. In order to avoid bias towards any of the base selection algorithms, this process was repeated six times, using a different technique each time – one of the above described ones, plus two others that were only used for model creation but not for base fitting evaluation. These additional methods were (1) a modification of method *RMSD of Matched Atoms* which did not enforce an atom match minimum; and (2) a modification of *Most Matched Atoms* which removed the RMSD threshold constraint.

After performing comparisons using each of the techniques and with all of the proteins in the data set (this resulted in 48 alignments), votes were tallied. Atoms of the 2HIX active site that had at least 5 votes from at least two different evaluation methods were used as part of a model.

2.4 Evaluation of Scoring Methods

In order to evaluate our base matching techniques we tested them against two different data sets. For the first set, we selected active sites of thirteen known ATP-binding and -nonbinding proteins (table 2). The second data set was designed to test how well each technique aligned our model, which, as described above, consisted of a small subset of atoms from the active site of 2HIX, with a larger portion of the same active site. A total of 157 atoms were selected from the active site; each selection was done based on distance from ATP. The resulting pool of atoms was a superset of the model. Then, one by one, we removed from the set 10 atoms that were also present in the model (the model was unchanged, however). In this way, we created 11 new test data sets: 2HIX_all, which contained all 157 atoms, and 2HIX_less1 – 2HIX_less10, each of which had one to ten atoms removed (respective atom counts: 156 to 147). Together, these and the 13 shown in table 2 (total 24) were used as our test data.

In each evaluation of model-target pairs, two metrics of best fit were gathered: the number of atoms matched (N) and depending on the method of evaluation RMSD of either all, top 80%, or just the matched atoms (R). The final score was the product of N and the reciprocal of R (i.e., $S = N * 1/R$).

Table 1. Atoms making up the model. Only atoms that received at least 5 votes using at least two different scoring schemes were used. For spatial representation of the model see figure 1.d.

Summary of Votes						
# Votes	# Methods	Residue	x	y	z	Symbol
10	4	LYS	11.351	29.733	18.933	C
5	2	TYR	12.16	26.491	15.053	N
7	2	ARG	19.336	25.785	21.914	C
13	5	ARG	18.899	28.507	26.98	C
5	2	GLU	15.775	24.236	18.813	O
20	5	PHE	20.149	25.464	14.586	C
28	5	PHE	21.012	26.236	15.354	C
20	5	PHE	20.919	26.261	16.744	C
11	3	PHE	19.916	25.505	17.387	C
8	3	MET	15.763	31.245	15.317	C
6	3	LYS	19.976	29.852	13.22	N
23	6	ARG	17.982	34.78	25.432	C
10	4	ARG	17.428	33.79	24.751	N
19	6	LYS	14.799	34.08	21.473	C
9	3	LYS	14.381	32.963	22.409	N

Table 2. Proteins used to evaluate base-fitting methods. Relation refers to structural similarity to 2HIX, an ATP-dependent DNA ligase from S. Solfataricus [7].

Relation	Symbol	Description
neighbor, ATP	1A0I	ATP-dependent DNA ligase from Bacteriophage T7.
neighbor, no ATP	1VS0	Component of Mycobacterium DNA ligase D
unrelated, ATP	1XMJ	Human deltaf508 Nbd1 domain
unrelated, ATP	1V1B	2-keto-3-deoxygluconate kinase from thermus thermophilus
unrelated, ATP	2J9L	Cytoplasmic domain of the human chloride transporter
unrelated, ATP	1MO8	ATPase
unrelated, ATP	200Y	Adenylate sensor from AMP-activated protein kinase
unrelated, ATP	2HF4	Monomeric actin
unrelated, no ATP	1C97	Isocitrate complex of aconitase
unrelated, no ATP	2B3X	Orthorhombic crystal form of human cytosolic aconitase
unrelated, no ATP	2IPY	Iron regulatory protein
unrelated, no ATP	1Q5O	Hcn2j 443-645
unrelated, no ATP	1LB2	E. Coli Alpha C-Terminal Domain Of RNA Polymerase.

3 Results and Experimental Evaluations

We created a model of an active site for ATP docking based on several different, known ATP-binding proteins. We then used this model to evaluate four different alignment scoring techniques, each of which was based on geometric hashing.

3.1 Model

Because we choose random ATP-binding proteins to build our model, we expected that they would vary in their interactions and binding mechanism with their ligand. Visual inspection verified our prediction. The portions of proteins that were extracted around their ligands were of different shapes and sizes. Active sites of some proteins interacted mostly with the base (the part of ATP consisting of two rings; see figure 1.a), others with phosphate group (the linear “tail” of ATP; see figure 1.a), but most with both parts (figure 2). The ligand was also positioned in each of these active sites in different poses (figure 3). These two factors contributed to a vast diversity of 3-dimentional conformations – virtually no two active sites were the same. Therefore, extraction of features common to all, and compilation of them into a single model was one of the key challenges. To address this, we utilized our geometric hashing-based voting approach and several unrelated proteins. In this way, we were able to combine into one model implicit geometric, physical, and chemical factors involved in a variety of mechanisms. The final model consisted of 15 atoms, which are listed in table 1 and rendered in figure 1.d.

3.2 Evaluation of Scoring Methods

Most biological applications that use geometric hashing for finding the best alignment between either atoms or feature points of two proteins evaluate the “goodness” of the match based on the overall RMSD [12]. In order to see if we could do better than RMSD, we evaluated each of the four here described methods. We tested how well each found the best base alignment and how well each filtered out active sites that did and did not bind ATP.

Overall RMS: First we evaluated how well the overall RMSD method picked the best base alignment of two segments of the same protein but of different sizes. To do this, we performed base pair alignment of our model and each of the following targets: 2HIX_all and 2HIX_less1 through 2HIX_less10. We expected that with all atoms present (2HIX_all) this method would be able to find the perfect alignment but that as atoms were removed and the number of overlapping points diminished, it would eventually fail. Our results verified our hypothesis. This method was able to find the perfect alignment with up to 7 atoms removed (47% removed).

Secondly, we evaluated this method based on how well it filtered out ATP-binding proteins from a pool of ATP-binding and -nonbinding targets. After aligning each of the 23 protein targets to the model, scores were computed and then used to rank the proteins (table 3.A). Not surprisingly, the best score was obtained by targets 2HIX_all through 2HIX_less7 (in that particular order), followed by 2HIX_less8 even though this method failed to find this protein’s correct base alignment. The next best scoring protein was a 1A0I, which is structurally very closely related to 2HIX and which binds ATP. However, the scores of the remaining 12 proteins did not seem to follow any predictable pattern. Targets built based on 2HIX that had more than eight atoms removed scored lower than some of the unrelated proteins. Similarly, about half of the ATP-nonbinding proteins scored better than their ATP-binding counterparts (figure 4). These data indicate that this method does well only with highly conserved structures which most likely bind ligands in similar poses. Figure 3(a) shows an example of confirmations of ligands of the model and a closely related protein which was scored very well by this method.

Top 80% RMSD: We hypothesized that this approach would handle noisy data, such as missing data and imperfect fit, better and more consistently than the overall

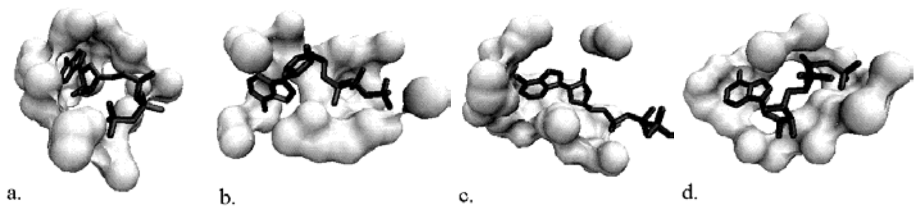


Fig. 2. Diversity of ligand interactions with active sites of proteins used to build and evaluate the model. Each shown active site includes atoms within 4Å of the ligand, ATP, which is shown as black, thick lines. **(a)** 2HIX – protein based on which the model was built; **(b)** 2EWW – used in voting to create model; **(c)** 1MO8 and **(d)** 2OOY were used for both model creation and method evaluation.

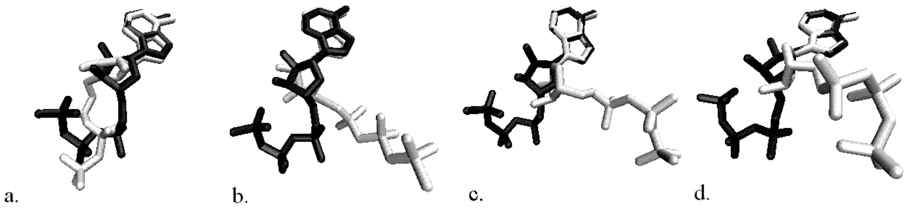


Fig. 3. Diversity of ligand binding conformations. The base portions of ligands were aligned; dark colored is ATP in its confirmation when docked in the active site of 2HIX, a protein based on which the model was created; light color shows ligands in their pose in active sites of proteins: **(a)** 1A0I – structurally closely related to 2HIX (note the similarity in ligand conformation), **(b)** 1V1B, **(c)** 2EWW, and **(d)** 2OOY. Proteins (a) and (b) were used only in evaluation; protein (c) just to build model; (d) used for both.

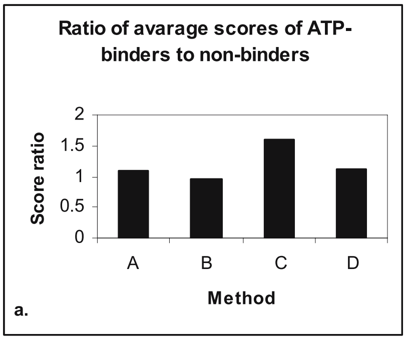
RMSD method. Surprisingly, it had the same threshold (7 atoms removed from the target) for finding the correct alignment. Similarly, it did equally well at filtering out structurally closely related proteins. It gave these proteins better scores than it gave to the 2HIX targets with more than 7 atoms removed. This suggests that it performs similarly well with missing data as it does with noisy data, but only if the model and target have high similarity (table 3.B). As was the case with overall RMSD, this method failed to distinguish the remaining ATP-binding from nonbinding targets. The rankings of the scores received by each target protein did not suggest that there is consistent preference for ATP-binding proteins (figure 4).

RMSD of Matched Atoms: This method was developed based on the assumption that not all atoms of the model were always required for binding. This was drawn from the fact that our model was built from a wide variety of ATP-binding proteins, and therefore, contained information relevant to several mechanisms. Because each ATP-binding protein had a somewhat different way of interacting with the ligand and hence anchoring via differently distributed atoms, the atoms contained in the model were unlikely to all be used simultaneously for binding. Based on this, we decided to remove the penalty for unmatched atoms that both the overall and top 80% RMSD methods had. The penalty originated from the inclusion in scoring, by both of these methods, of atoms with no close fit. Each time a base alignment contained one or more such atoms, the total RMSD was negatively impacted. In order to avoid this we decided to compute RMSD only for atoms that did have a match. In order to ensure that the match is of significant length we added a constraint that at least a third of the atoms of the model have to have a close mate in the target.

We hypothesized that this method would perform well in finding partial regions with the tightest fit possible. And indeed, it did do a good job at performing perfect alignments between the model and the 2HIX targets with removed atoms. It was able to reliably find perfect fit for all of them, even the ones that contained as few as 5 of the atoms that were also contained in the models.

Moreover, this method did significantly better at filtering ATP-binding and -non-binding proteins. Out of the 24 target proteins that we tested, most that did use ATP as a ligand scored better than those that did not. There were two exceptions – one, 2OOY, an ATP-binding protein, obtained the lowest score of all of the test targets. The second exception was an ATP-nonbinding protein, 1Q5O. It obtained a score better than three of the ATP-binding proteins. It was also interesting to observe that it did not rank the structurally closely related proteins as highly as the other methods did. It did not give these proteins a score that was highest right after that of self-superset targets (2HIX-x) (table 3.C, figure 4). Investigation of ligand poses of proteins that scored well using this method revealed that it was able to handle a diversity of 3-dimentional confirmations. Figures 3(a) and 3(b) show poses of ligands of the two best scoring proteins.

Most Matched Atoms: Our last approach concentrated on evaluating bases based on the number of atoms that were matched. Our test verified our hypothesis that the tendencies of this approach were to pick alignments that found a "good enough" match for as many atoms as possible; it lacked sensitivity for shorter regions with very close match. This method turned out to perform quite similarly to overall RMDS. It was able to correctly align the model to the target with up to 8 of the atoms missing, at which point it ranked the target on par with other ATP-binding and -nonbinding proteins (table 3.D). Ranking of the scores did not result in any noticeably meaningful pattern and did not separate binders from non-binders (figure 4).



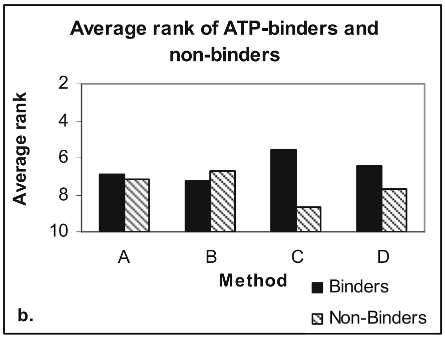


Fig. 4. Scores of ATP binders and nonbinders using method A: *Overall RMSD*, method B: *Top 80% RMSD*, C: *RMSD of Matched Atom*, and *D: Most Matched Atoms*. **(a)** Ratios of average scores obtained by ATP-binding proteins to those of ATP-nonbinding proteins. Ratio of 1 indicates that a binder cannot be distinguished from a nonbinder. **(b)** Average rank of ATP-binding and -nonbinding proteins.

Table 3. Results of evaluation of the four methods of base alignment scoring. Each sub-table contains ranked results of the matching between the model and each of the proteins listed in the left hand columns. Shaded rows contain proteins that do not bind ATP; clear rows contain proteins that do. Because scores in each of the sub-tables were computed using a different scoring technique, they should not be cross-compared. The cross-table scores by themselves are not informative enough to evaluate which method obtained a better alignment. This can only be done by further inspection. However, it is expected, in all categories, that proteins that do not bind ATP would score lower than those that do and, therefore, sorting by score would filter them down to the bottom of the table. Method (C), the *RMSD of Matched Atoms*, performs best.

A. Overall RMSD		**B. Top 80% RMSD**		**C. RMSD of Matched**		**D. Longest Match**	
protein	**score**	**Protein**	**score**	**protein**	**score**	**protein**	**score**
2HIX_less4	**27.99**	2HIX_less4	**93.22**	2HIX_less4	∞	2HIX_less4	∞
2HIX_less5	**20.24**	2HIX_less5	**42.02**	2HIX_less5	∞	2HIX_less5	∞
2HIX_less7	**11.763**	2HIX_less7	**16.994**	2HIX_less7	∞	2HIX_less7	∞
2HIX_less8	**9.86**	1A0I	**11.54**	2HIX_less8	∞	2J9L	**25.86**
1A0I	**7.99**	2HIX_less8	**10.62**	2HIX_less9	∞	1A0I	**18.32**
2HF4	**6.22**	2HIX_less9	**10.62**	1V1B	**70.42**	1C97	**17.86**
2HIX_less9	**6.08**	1V1B	**9.23**	1A0I	**58.77**	2HIX_less8	**16.81**
1VS0	**5.43**	2IPY	**8.97**	1XMJ	**41.67**	2HIX_less9	**16.81**
1V1B	**4.96**	2B3X	**7.61**	2J9L	**36.76**	1VS0	**16.16**
2IPY	**4.79**	1VS0	**7.60**	1Q5O	**34.48**	1Q5O	**15.90**
2B3X	**4.64**	1Q5O	**7.54**	1MO8	**30.49**	1V1B	**14.93**
1LB2	**4.07**	1XMJ	**7.07**	2HF4	**30.49**	2HF4	**14.44**
1XMJ	**3.48**	1MO8	**6.09**	1VS0	**27.37**	200Y	**13.55**
2J9L	**3.34**	2HF4	**5.28**	2IPY	**25.80**	1MO8	**13.49**
1Q5O	**3.27**	1LB2	**3.60**	2B3X	**22.73**	2B3X	**12.97**
1MO8	**2.41**	2J9L	**2.77**	1C97	**17.86**	1XMJ	**12.90**
1C97	**1.30**	200Y	**2.58**	1LB2	**17.21**	2IPY	**12.22**
200Y	**1.28**	1C97	**1.76**	200Y	**13.55**	1LB2	**10.99**

4 Conclusions

In this paper, we proposed and investigated multiple improvements to the robustness of geometric hashing based approaches to predicting ligand-protein interactions. We considered four methods of measuring the “goodness” of fit between a model and an active site. Our investigations demonstrate that the current most commonly used scoring technique – an *Overall RMSD* of the match – performs well with partial data and does well finding similarities between structurally closely related proteins. However, it performs poorly in terms of distinguishing between unrelated proteins that bind and those that do not bind ATP. Two of the other evaluation methods: *Top 80% RMSD* and *Most Matched Atoms*, also have similar problems. They were unable to predict which proteins did bind the ligand. The *RMSD of Matched Atoms* technique performed notably better. Ranking of scores received by various proteins revealed that this method performed better in distinguishing between binders and nonbinders.

The presented model of an active site was built utilizing a novel data-driven approach. Because it was built based on active sites of several, unrelated proteins it reflected properties of a diverse range of binding mechanisms. We postulate that the proposed approach and binding-site models derived from it present important advantages in comparison to other methods in terms of their robustness to missing data, and their ability to handle variations in the 3D poses of ligands. Furthermore, being derived from a number of structurally diverse binding mechanisms, it provides a more general binding site definition than techniques that only analyze a pair of interacting molecules. Our future work is directed at comparing the quality of binding sites derived using the proposed method with those obtained using other techniques.

References

1. Brakoulias, A., Jackson, R.M.: Towards a structural classification of phosphate binding sites in protein-nucleotide complexes: an automated all-against-all structural comparison using geometric matching. Proteins 56, 250–260
2. Comoglu, O., Kahveci, T., Singh, A.K.: Towards Index-based similarity search for protein structure databases. In: CSB 2003. Proceedings of the Computational Systems Bioinformatics (2003) 0-7695-2000-6/03
3. Coupez, B., Lewis, R.A.: Docking and Scoring – Theoretically Easy, Practically Impossible? Cur. Medicinal Chemistry 13, 2995–3003 (2006)
4. Edelsbrunner, H., Facello, M., Liang, J.: On the Definition and the Construction of Pockets in Macromolecules. Discrete Applied Mathematics 88, 83–102 (1998)
5. Laurie, A.T., Jackson, R.M.: Methods for the Prediction of Protein-Ligand Binding Sites for Structure-Based Drug Design and Virtual Screening. Current Protein and Peptide Science 7, 395–406 (2006)
6. Richard Jackson's group – ligand binding sites software, http://www.modelling.leeds.ac.uk/sb/
7. NCBI Protein Structure Database, ATP binding proteins: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?dopt=pccompound_structure&db=pccompound&cmd=Display&from_uid=5957
8. NCBI PubChem, ATP: http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=5957
9. PDB, http://www.rcsb.org/pdb/
10. Wang, J.T., Marr, T.G., Shasha, D., Shapiro, A.B., Chirn, G.-W.: Discovering active motifs in sets of related protein sequences and using them for classification. Nucleic Acids Res. 22(14), 2769–2775 (1994)
11. Visual Molecular Dynamics, Theoretical and Computational Biophysics Group, University or Illinois at Urbana-Champaign, http://www.ks.uiuc.edu/Research/vmd/
12. Rosen, M., Lin, S.L., Wolfson, H., Nussinov, R.: Molecular shape comparison in searches for active sites and functional similarity. Protein Engineering 11(4), 263–277 (1998)
13. Wolfson, H.: Geometric Hashing: an Overview. IEEE Computational Science and Engineering, 1070–9924 (1997)

4D Ventricular Segmentation and Wall Motion Estimation Using Efficient Discrete Optimization

Ahmed Besbes[1], Nikos Komodakis[1], Ben Glocker[2], Georgios Tziritas[3], and Nikos Paragios[1]

[1] GALEN Group, Laboratoire MAS, Ecole Centrale de Paris
{ahmed.besbes,nikos.komodakis,nikos.paragios}@ecp.fr
[2] Chair for Computer Aided Medical Procedures (CAMP)
Technische Universität München
glocker@in.tum.de
[3] University of Crete, Computer Science Department
tziritas@csd.uoc.gr

Abstract. In this paper we propose a novel approach to ventricular motion estimation and segmentation. Our method is based on a MRF formulation where an optimal intensity-based separation between the endocardium and the rest of the cardiac volume is to be determined. Such a term is defined in the spatiotemporal domain, where the ventricular wall motion is introduced to account for correspondences between the consecutive segmentation maps. The estimation of the deformations is done through a continuous deformation field (FFD) where the displacements of the control points are determined using discrete labeling approach. Principles from linear programming and in particular the Primal/Dual Schema is used to recover the optimal solution in both spaces. Promising experimental results obtained on 13 MR spatiotemporal data sets demonstrate the potentials of our method.

1 Introduction

The segmentation of the left ventricle has been a problem well addressed in medical imaging. Prior art either refers to model-free approaches or model-based. Model-free methods do not make an explicit assumption on the form/geometric properties as well as the appearance of the ventricle. MRFs [1], snakes [2], level sets [3], shortest path [4] have been considered in this context. On the other hand, model-based methods often consider certain geometric priors for the ventricle which could range from simple 2D shapes [4] and 3D models which also encode local variations [1] to complex biomechanical cardiac models [5].

Ventricular wall motion estimation was often addressed through the use of MR-Tagging [6] [7] techniques that consist of introducing a rectangular pattern on the acquisition. Direct 3D motion estimation in MR is a more challenging problem since it is known that the left ventricle undergoes a rather complex deformation within the cardiac cycle. In order to account for the ill-posedness of the problem, the use of shape models towards establishing visual correspondences and tracking was often considered [8] or 4D models have been constructed with spatial and temporal deformations being encoded [9]. Voxel-based methods often explore the visual preservation assumption [10] while being constrained to provide a smooth deformation map. More complex models use biomechanical constraints to determine such a deformation [11], an approach which might fail when processing diseased data.

In most of the cases, these methods do not relate segmentation with ventricular motion estimation. Furthermore, one can claim that they are sensitive to the initial conditions either because of the non-convexity of the designed cost function or due to the sub-optimal optimization approach. In this paper, we propose a novel approach to address both segmentation and ventricular motion estimation. We overcome the ill-posedness of the motion estimation problem through the use of interpolation techniques with higher order polynomials, while we introduce temporal segmentation consistency through the use of deformations field. In order to efficiently recover the optimal solution to the problem, we re-formulate the cost function in a fully discrete domain where the latest developments of linear programming are considered to determine the lowest potential of the cost function. Very promising results and comparisons with manual segmentation from physicians demonstrate the potentials of our approach.

2 Ventricular Segmentation and Wall Motion Estimation

2.1 Spatiotemporal Segmentation

Let us consider a spatiotemporal volume $\mathcal{V}(\mathbf{x};t) : \Omega \times [0..\tau] \rightarrow \mathcal{R}$, with Ω being the volume domain. The task of segmenting the endocardium can be reformulated using a labeling approach, or assigning a label $\phi(\mathbf{x};t) : \Omega \times [0..\tau] \rightarrow \{0,1\}$. Here, label 0 corresponds to the foreground (i.e., the ventricle), whereas label 1 corresponds to the background. Without loss of generality, let us assume that certain statistical properties on the intensities of the left ventricle $p(\mathcal{V}|\phi = 0)$, as well as on the intensities of the background $p(\mathcal{V}|\phi = 1)$ are available or can be determined on the fly. Let us also assume that we have a prior left ventricle closed surface $(S^t)_{t=0}^{\tau}$ defined as:

$$\xi_S(\mathbf{x};t) = \begin{cases} 0 & \text{if } \mathbf{x} \in S^t \\ -\mathcal{D}(\mathbf{x},\mathcal{S}) & \text{if } \mathbf{x} \in S^t_{in} \\ \mathcal{D}(\mathbf{x},\mathcal{S}) & \text{if } \mathbf{x} \in S^t_{out} \end{cases} . \tag{1}$$

with $\mathcal{D}$ being the Euclidean distance between a given voxel and the surface, and$(S^t, S^t_{in}, S^t_{out})$ being the partition of Ω defined by S^t, $\forall t \in [0..\tau]$. We define a penalization function $p_\epsilon(\xi;\phi) : \mathcal{R} \times \{0,1\} \rightarrow \mathcal{R}$, with $\epsilon > 0$ as a decreasing (respectively increasing) function of ξ if $\phi = 0$ (respectively $\phi = 1$), and equal to identity for $\xi < \epsilon$.

In such a case, the optimal labeling should refer to the maximum conditional posterior between the decisions and the data support. If spatial and temporal independence are assumed between voxels, that labeling can then be recovered through the minimum of:

$$\begin{aligned} E_{seg,dt}(\phi) &= \sum_{t=0}^{\tau} \sum_{\mathbf{x} \in \Omega} -\log \left[p \left(\mathcal{V}(\mathbf{x};t) | \phi(\mathbf{x};t) \right) . p_{\epsilon} \left(\xi_S(\mathbf{x};t); \phi(\mathbf{x};t) \right) \right] \\ &= \sum_{t=0}^{\tau} \sum_{\mathbf{x} \in \Omega} V_{dt}^{p}(\phi(\mathbf{x};t)) \ . \end{aligned} \tag{2}$$

which is equivalent to assigning to each voxel the label which is optimally supported from the observation. Such a simplistic formulation could produce sub-optimal results due the presence of noise and therefore one should introduce additional smoothness constraints on the label space, which aims to enforce regularity on the decisions, or:

$$E_{seg,sp}(\phi) = \sum_{t=0}^{\tau} \sum_{\mathbf{x} \in \Omega} \left(\sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} \psi(\phi(\mathbf{x};t), \phi(\mathbf{y};t)) \right) = \sum_{t=0}^{\tau} \sum_{\substack{\mathbf{x} \in \Omega \\ \mathbf{y} \in \mathcal{N}(\mathbf{x})}} V_{sp}(\phi(\mathbf{x};t), \phi(\mathbf{y};t)) \ . \tag{3}$$

with ψ being a function measuring the dissimilarity between labels of neighboring pixels and $\mathcal{N}(\mathbf{x})$ defines the local neighborhood of $\mathbf{x}$ in the 3D spatial domain. For more robustness, one can also consider temporal constraints on the labeling if the deformations from one volume to the next are not so important, which is, however, definitely not the case for the left and right ventricular motion. On the other hand, if we assume that this deformation is known, say, $d(\mathbf{x};t)$, then one can imagine using $d(\mathbf{x};t)$ towards determining the temporal derivative on the label space and introduce a smoothness constraint of the following form:

$$\begin{aligned} E_{seg,tm}(\phi | \, d) &= \sum_{t=0}^{\tau-1} \sum_{\mathbf{x} \in \Omega} \psi(\phi(\mathbf{x};t), \phi(\mathbf{x} + d(\mathbf{x};t); t+1)) d\mathbf{x} \\ &= \sum_{t=0}^{\tau-1} \sum_{\mathbf{x} \in \Omega} V_{tm}(\phi(\mathbf{x};t), \phi(\mathbf{x} + d(\mathbf{x};t); t+1)) \ . \end{aligned} \tag{4}$$

The interpretation of this term is straightforward, assuming known correspondences one would expect a coherent labeling between anatomical structures within the cardiac cycle. Based on this fact, we can therefore proceed as follows: we will first estimate the deformation $d(\mathbf{x};t)$, i.e. register the 3D volumes, and then we will extract the optimal segmentation (i.e. the optimal labeling $\phi(\mathbf{x};t)$)) by minimizing the total energy E_{4D} of the resulting binary 4D Markov Random Field, where the total energy is given by:

$$E_{seg}(\phi | \, d) = E_{seg,dt}(\phi) + \alpha E_{seg,sp}(\phi) + \beta E_{seg,tm}(\phi | \, d) \ . \tag{5}$$

Intuitively, the edges of the resulting 4D MRF will consist of regular links, connecting (in a grid-like manner) voxels belonging to the same 3D volume. On the other hand, they refer to irregular links in the temporal domain, connecting voxels between adjacent 3D volumes, being determined via the previously estimated deformation $d(\mathbf{x};t)$. We also note that because our MRF is binary, the exact global optimum can be easily extracted [12].

However, establishing correspondences between volumes is an ill-posed problem. Even if we assume the visual preservation assumption to be valid (not often the case for medical image modalities), one should determine three unknown variables from a single constraint. To deal with this issue, in the next section we show how we can regularize this motion estimation problem by reformulating it as another discrete MRF optimization problem.

2.2 Ventricular Motion Estimation

Let us thus assume that we wish to compute the deformation $d(\mathbf{x};t)$ between two adjacent 3D volumes at time t. To this end, we will introduce a sparse deformation grid $\mathcal{G}$ super-imposed on the source volume (no particular assumption is made on the grid except that it is sparser than the original volume). The central idea of our approach is to deform the grid (with a 3D displacement vector $d(\mathbf{p};t)$ for each control point $\mathbf{p}$) such that the underlying volumes are perfectly aligned. Without loss of generality, we can then assume that the displacement of a voxel $\mathbf{x}$ can be expressed using a linear or non-linear combination of the grid points, or:

$$d(\mathbf{x};t) = \sum_{p\in\mathcal{G}} \eta(|\mathbf{x}-\mathbf{p}|)\, d(\mathbf{p};t) \ . \tag{6}$$

where $\eta(\cdot)$ is the weighting function measuring the contribution of the control point $\mathbf{p}$ to the displacement field $d(\mathbf{x};t)$. The use of such a model is motivated by the fact that the observations refer to anatomical structures with a rather natural temporal deformation. Furthermore, such an approach could help us to account for the ill-posedness of the problem due to the fact that the estimation of a single 3D displacement is now an over-constrained problem with many observations being available. For $\eta(\cdot)$, we use a three-dimensional Free Form Deformation (FFD) model based on cubic B-splines [13] (other interpolation models can also be considered).

Therefore, based on (6), to estimate $d(\mathbf{x};t)$ it suffices to specify the displacements for the control points. To this end, we will consider a quantized version of the deformation space, say, $\{d^1,...,d^i\}$ - being 3D deformation vectors - as well as a corresponding set of discrete labels, say, $\mathcal{L} = \{1,...,i\}$. A label assignment, say, $\omega(\mathbf{p}) \in \mathcal{L}$ to a grid point $\mathbf{p}$ is associated with displacing $\mathbf{p}$ by the corresponding vector $d^{\omega(\mathbf{p})}$, i.e.:

$$d(\mathbf{p};t) = d^{\omega(\mathbf{p})} \ . \tag{7}$$

The visual preservation imposes the constraint that the observation of the same anatomical patch should be consistent across volumes, i.e., $\mathcal{V}(\mathbf{x};t) \approx \mathcal{V}(\mathbf{x} + d(\mathbf{x};t); t+1)$. In our discrete framework the deformation $d(\mathbf{x};t)$ is defined based on (6), (7), i.e. displacements are associated with labels, one can reformulate ventricular deformation estimation as a labeling problem. Consequently, the goal is to assign a set of appropriate labels $\{\omega(\mathbf{p})\}$ (to the grid points) so that the visual preservation constraint is satisfied as much as possible, or equivalently so that the following data cost is minimized:

$$E_{mot,dt}(\omega) = \sum_{\mathbf{x}\in\Omega} |\mathcal{V}(\mathbf{x};t) - \mathcal{V}(\mathbf{x}+d(\mathbf{x};t);t+1)| \overset{(6),(7)}{\approx} \sum_{\mathbf{p}\in\mathcal{G}} U^p_{dt}(\omega(\mathbf{p})) \ . \tag{8}$$

Here, the singleton potential functions $U^p_{dt}(\cdot)$ are not independent, thus the defined data term can only be approximated. Hence, we precompute the $|\mathcal{L}| \times |\mathcal{G}|$ (where $|\mathcal{G}|$ is the number of grid points) data term in a look-up table. The entry for label $\omega(\mathbf{p})$ and node $\mathbf{p}$ is determined by:

$$U^p_{dt}(\omega(\mathbf{p})) = \iint_{\Omega(\mathbf{p})} \eta^{-1}(|\mathbf{x}-\mathbf{p}|) \cdot \left|\mathcal{V}(\mathbf{x};t) - \mathcal{V}(\mathbf{x}+d^{\omega(\mathbf{p})};t+1)\right| d\mathbf{x} \ . \tag{9}$$

with the sum of absolute differences being considered as measure of similarity (η^{-1} is the inverse projection between $\mathbf{x}$ and $\mathbf{p}$). The use of an interpolation technique to determine the deformations of the volume will inherit natural smoothness to the estimates. However, one should also expect since we aim to recover measurements for physical objects deformations that the same assumption is satisfied for the deformation of the corresponding control points. Similar to the segmentation case, one can consider a term which enforces spatial similarities across labels, or:

$$E_{mot,sm}(\omega) = \sum_{\substack{\mathbf{p}\in\Omega \\ \mathbf{q}\in\mathcal{N}(\mathbf{p})}} U_{sm}(\omega(\mathbf{p}),\omega(\mathbf{q})) \ . \tag{10}$$

where $\mathcal{N}$ represents the neighborhood system associated with the deformation grid $\mathcal{G}$. For the distance $U_{sm}(\cdot,\cdot)$, we consider a simple piecewise smoothness term based on the Euclidean distance between the deformations corresponding to the assigned labels, i.e.:

$$U_{sm}(\omega(\mathbf{p}),\omega(\mathbf{q})) = \lambda_{pq}\left(|d^{\omega(\mathbf{p})} - d^{\omega(\mathbf{q})}|\right) \ . \tag{11}$$

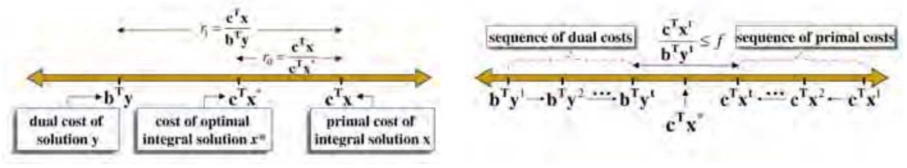
with λ_{pq} being a (spatially varying) weighting to control the influence of the smoothness/prior term. Such a smoothness term, together with the data term, allows to convert the problem of volume registration into a discrete MRF optimization problem with the following energy [14]:

$$E_{mot}(\omega) = E_{mot,dt}(\omega) + E_{mot,sm}(\omega) \ . \tag{12}$$

2.3 4D Segmentation and Ventricular Motion Estimation

One can now consider an objective function which recovers both the 4D segmentation map as well as the corresponding deformation fields:

$$E_{seg,mot}(\phi,\omega) = E_{seg}(\phi|\,\omega) + \gamma E_{mot}(\omega) \ . \tag{13}$$


$r_1 = \frac{\mathbf{c}^T\mathbf{x}}{\mathbf{b}^T\mathbf{y}}$
$r_0 = \frac{\mathbf{c}^T\mathbf{x}}{\mathbf{c}^T\mathbf{x}^*}$
$\mathbf{b}^T\mathbf{y}$
$\mathbf{c}^T\mathbf{x}^*$
$\mathbf{c}^T\mathbf{x}$
dual cost of solution y
cost of optimal integral solution x^*
primal cost of integral solution x
sequence of dual costs
$\frac{\mathbf{c}^T\mathbf{x}^t}{\mathbf{b}^T\mathbf{y}^t} \le f$
sequence of primal costs
$\mathbf{b}^T\mathbf{y}^1$ $\mathbf{b}^T\mathbf{y}^2$ ... $\mathbf{b}^T\mathbf{y}^t$
$\mathbf{c}^T\mathbf{x}^*$
$\mathbf{c}^T\mathbf{x}^t$ ... $\mathbf{c}^T\mathbf{x}^2$ $\mathbf{c}^T\mathbf{x}^1$

(a) The primal-dual principle (b) The primal-dual schema

Fig. 1. (a) By weak duality, the optimal cost $\mathbf{c}^T\mathbf{x}^*$ will lie between the costs $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ of any pair $(\mathbf{x}, \mathbf{y})$ of integral-primal and dual feasible solutions. Therefore, if $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ are close enough (e.g. their ratio r_1 is $\le f$), so are $\mathbf{c}^T\mathbf{x}^*$ and $\mathbf{c}^T\mathbf{x}$ (e.g. their ratio r_0 is $\le f$ as well), thus proving that $\mathbf{x}$ is an f-approximation to $\mathbf{x}^*$. **(b)** According to the primal-dual schema, dual and integral-primal feasible solutions make local improvements to each other, until the final costs $\mathbf{b}^T\mathbf{y}^t$, $\mathbf{c}^T\mathbf{x}^t$ are close enough (e.g. their ratio is $\le f$). We can then apply the primal-dual principle (as in Fig. (a)) and thus conclude that $\mathbf{x}^t$ is an f-approximation to $\mathbf{x}^*$.

which is a fully discrete optimization problem. For optimizing the resulting MRF, we seek to assign a pair of labels $(\phi(\mathbf{p}), \omega(\mathbf{p}))$ to each node $p \in \mathcal{G}$, so that the MRF energy in (13) is minimized. To this end, a recently proposed method, called Fast-PD (Fast Primal Dual), will be used. This is an optimization technique, which builds upon principles drawn from the duality theory of linear programming in order to efficiently derive almost optimal solutions for a very wide class of NP-hard MRFs. For more details about the Fast-PD algorithm, the reader is referred to [12]. Here, we will just provide a brief, high level description of the basic driving force behind that algorithm.

3 Linear Programming

The driving force of the algorithm consists of the *primal-dual schema*, which is a well-known technique in the Linear Programming literature. To understand how the primal-dual schema works in general, we will need to consider the following pair of primal and dual Linear Programs (LPs):

$$\textsc{Primal:}\ \begin{array}{l}\min \mathbf{c}^T\mathbf{x}\\ \text{s.t.}\quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \ge \mathbf{0}\end{array} \qquad \textsc{Dual:}\ \begin{array}{l}\max \mathbf{b}^T\mathbf{y}\\ \text{s.t.}\quad \mathbf{A}^T\mathbf{y} \le \mathbf{c}\end{array} \tag{14}$$

Here $\mathbf{A}$ represents a coefficient matrix, while $\mathbf{b}, \mathbf{c}$ are coefficient vectors. Also, $\mathbf{x}$, $\mathbf{y}$ represent the vectors of primal and dual variables respectively. We seek an optimal solution to the primal program, but with the extra constraint of $\mathbf{x}$ being integral. Due to this integrality requirement, this problem is in general NP-hard and so we need to settle with estimating approximate solutions. A primal-dual f-approximation algorithm achieves that by use of the following principle (illustrated also in Fig. 1(a)):

Primal-Dual Principle 1. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are integral-primal and dual feasible solutions having a primal-dual gap less than* f*, i.e.:*

$$\mathbf{c}^T\mathbf{x} \leq f \cdot \mathbf{b}^T\mathbf{y}, \tag{15}$$

then $\mathbf{x}$ *is an* f*-approximation to the optimal integral solution* $\mathbf{x}^*$*, i.e.* $\mathbf{c}^T\mathbf{x}^* \leq \mathbf{c}^T\mathbf{x} \leq f \cdot \mathbf{c}^T\mathbf{x}^*$*.*

Based on the above principle, that lies at the heart of any primal-dual technique, the following iterative schema can be used for deriving an f-approximate solution (this schema is also illustrated graphically in Fig. 1(b)):

Primal-Dual Schema 1. *Keep generating pairs of integral-primal, dual solutions* $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1}^t$*, until the elements* $\mathbf{x}^t$*,* $\mathbf{y}^t$ *of the last pair are both feasible and have a primal-dual gap which is less than* f*, i.e. condition* (15) *holds true.*

In order to apply the above schema to MRF optimization, it suffices that we cast the MRF optimization problem as an equivalent integer program. The Fast-PD algorithm is then derived by applying the primal-dual schema to this pair of primal-dual LPs, while using $f = 2\frac{d_{\max}}{d_{\min}}$ ($d_{\max} \equiv \max_{a \neq b} d(a,b)$, $d_{\min} \equiv \min_{a \neq b} d(a,b)$) as the approximation factor in (15). Fast-PD is a very general MRF optimization method, which can handle a very wide class of MRFs. Essentially, it only requires that the MRF pairwise potential function is nonnegative. Furthermore, as already mentioned, it can guarantee that the generated solution is always within a worst-case bound from the optimum. In fact, besides this worst-case bound, it can also provide per-instance approximation bounds, which prove to be much tighter, i.e. very close to 1, in practice. It thus allows the global optimum to be found up to a user/application bound. Finally, it provides great computational efficiency, since it is typically 3-9 times faster than any other MRF optimization technique with guaranteed optimality properties [12].

4 Validation

In order to validate the performance of the method we have considered a set of 13 MR spatiotemporal volumes of the heart, with manual segmentation from two clinical experts being available for the diastole and the systole. These data sets had a spatial resolution of around 100x100x12 and a voxel size of around 1.77x1.77x6 millimeters. We used as prior information two learned distributions of endocardium voxels and background voxels expressed as mixture of Gaussians.

Table 1. Comparison of automatic and experts' segmentations in diastole

Comparison	DSC Mean (Std)	Sensitivity	Specificity	ASD Mean (Std)
Our Method vs Expert1	0.86(±0.03)	99.06%	95.76%	1.54(±0.39)
Our Method vs Expert2	0.87(±0.02)	99.11%	96.88%	1.31(±0.37)
Expert1 vs Expert2	0.89(±0.02)	99.49%	94.46%	0.87(±0.12)
Expert2 vs Expert1	0.89(±0.02)	99.53%	94.16%	1.34(±0.47)

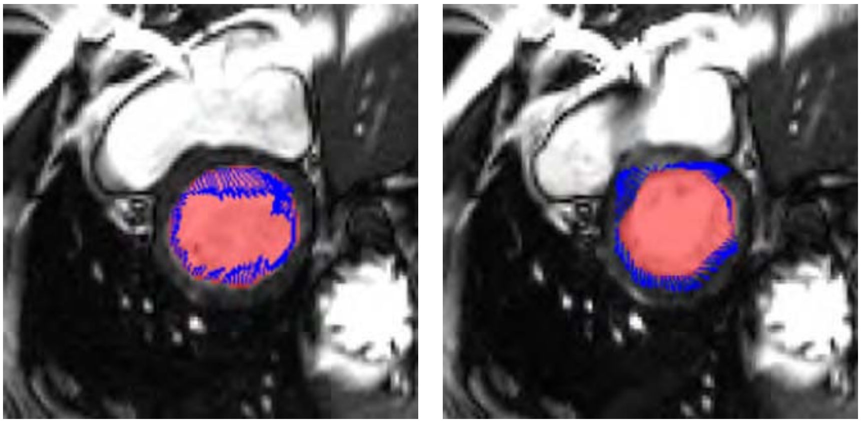

(a) centripetal motion field (b) centrifugal motion field

Fig. 2. Motion estimation. **(a)** beginning of systole **(b)** beginning of diastole.

These distributions were time-independent, and were used in diastole and systole as well. We also added a shape prior constraint (fixed shape S, initialized by the user) to account for the elliptic geometry of the left ventricle. In terms of segmentation performance we compare the experts' segmentations of the endocardium with the one obtained using the proposed method. We are interested on several common evaluation measurements [15], and in particular the Dice similarity coefficient (DSC), the sensitivity, the specificity, and the average surface distance (ASD) from the experts' segmentations. The ASD is computed in millimeters from an anisotropic 3D Euclidean distance transform of the surfaces. These measurements are computed in both diastole and systole and are presented in [Tab. (1)] and [Tab. (2)]. We also compare in these tables the performances of our method to those achieved manually by the experts.

We achieve an ASD which is below the voxel size in both diastole and systole. The DSC which measures the overlap between surfaces shows that our segmentation is closer to the one given by Expert2 than to the one given by Expert1. Overall, our performance is satisfactory compared to the one achieved by the experts. We get a worse sensitivity than the experts, but a better specificity. In terms of ventricular motion estimation, we present in [Fig. (2)] the deformation field of the endocardium and its motion estimation. We see in particular in this figure that the motion field is coherent with the left ventricle motion: the centripetal motion field at the beginning of systole is justified by the contraction of the myocardium, and the centrifugal motion field at the beginning of diastole is justified by its expansion. The 3D images in [Fig. (3)] show that we also correctly segment the papillary muscles.

With a reasonable number of displacement labels (the complexity is linear to the number of labels), the method takes about 10-20 seconds to converge (using a DELL Duo with (3GHz,2GB)) assuming that a ventricle isolation has been done and is able to produce good correspondences with a $16 \times 16 \times 16$ FFD grid. The

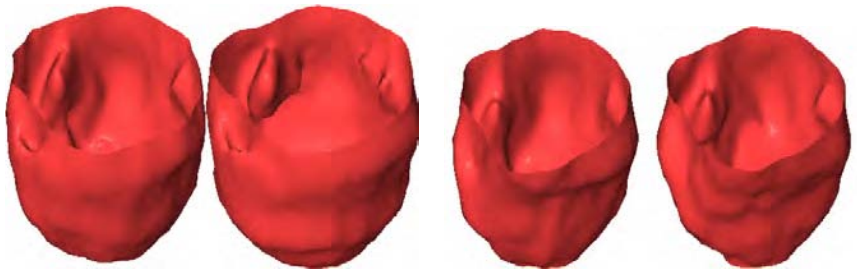


Fig. 3. Papillary muscles. In each image : automatic segmentation & expert's manual segmentation.

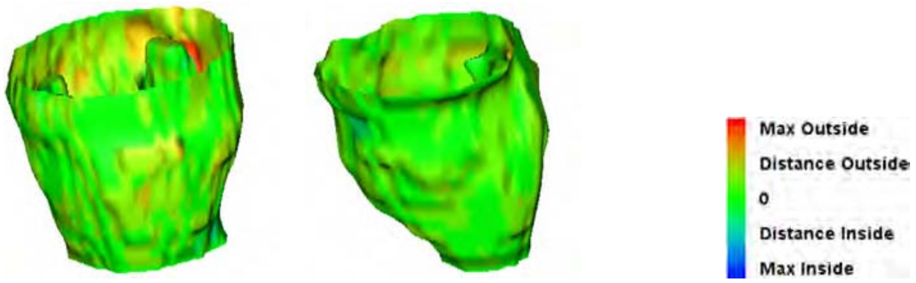


Fig. 4. Color-encoded visualization of the average surface distance for the example shown in [Fig. (2)]. **(a)** beginning of systole **(b)** beginning of diastole.

cardiac cycle being quantized by 20-25 time points, the whole 4D segmentation and motion estimation computation takes about 70-80 seconds for a 4D volume.

Table 2. Comparison of automatic and experts' segmentations in systole

Comparison	DSC Mean (Std)	Sensitivity	Specificity	ASD Mean (Std)
Our Method vs Expert1	0.82(±0.03)	99.39%	93.34%	1.51(±0.39)
Our Method vs Expert2	0.85(±0.03)	99.46%	94.34%	1.28(±0.37)
Expert1 vs Expert2	0.86(±0.03)	99.69%	91.07%	0.86(±0.15)
Expert2 vs Expert1	0.86(±0.03)	99.66%	91.50%	1.06(±0.22)

5 Discussion

In this paper we have proposed a novel discrete approach to spatiotemporal segmentation and ventricular motion estimation. The strength of our approach is the coupling between the two problems and the use of a powerful combinatorial algorithm to produce their solution. In order to demonstrate the concept, we have considered a set of several heart 4D MRI exams and we have obtained quite satisfactory results. More challenging perspectives are related with the introduction of prior knowledge both in space and time related with the evolving geometry of the structures of interest. The prior information used in our approach remains quite simple, and is time-independent. That is why our results are promising and can be probably improved by the use of more complex prior information which can better capture the anatomy and the temporal dynamics of the cardiac cycle. Knowledge-based segmentation using models that encode important statistical variation of training examples within discrete optimization is a quite promising direction to be considered.

References

1. Shi, P., Sinusas, A.J., Constable, R.T., Ritman, E., Duncan, J.S.: Point-tracked quantitative analysis of left ventricular surface motion from 3d image sequences. IEEE Trans. Med. Imaging 19(1), 36–50 (2000)
2. McInerney, T., Terzopoulos, D.: Deformable models in medical images analysis: a survey. Medical Image Analysis 1(2), 91–108 (1996)
3. Paragios, N.: A variational approach for the segmentation of the left ventricle in cardiac image analysis. Int. J. Comput. Vision 50(3), 345–362 (2002)
4. Jolly, M.-P.: Automatic segmentation of the left ventricle in cardiac mr and ct images. Int. J. Comput. Vision 70(2), 151–163 (2006)
5. Sermesant, M., Forest, C., Pennec, X., Delingette, H., Ayache, N.: Deformable biomechanical bodels: Application to 4D cardiac image analysis. Medical Image Analysis 7(4), 475–488 (2003)
6. Montillo, A., Metaxas, D.N., Axel, L.: Automated model-based segmentation of the left and right ventricles in tagged cardiac mri. In: MICCAI, vol. 1, pp. 507–515 (2003)
7. Guttman, M.A., Prince, J.L., McVeigh, E.R.: Tag and contour detection in tagged mr images of the left ventricle. IEEE Trans. Med. Imaging 13(1), 74–88 (1994)
8. McEachen II, J.C., Duncan, J.S.: Shape-based tracking of left ventricular wall motion. IEEE Trans. Med. Imaging 16(3), 270–283 (1997)
9. Bosch, J.G., Mitchell, S.C., Lelieveldt, B.P.F., Nijland, F., Kamp, O., Sonka, M., Reiber, J.H.C.: Automatic segmentation of echocardiographic sequences by active appearance motion models. IEEE Trans. Med. Imaging 21(11), 1374–1383 (2002)
10. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artif. Intell. 17(1-3), 185–203 (1981)
11. Sermesant, M., Delingette, H., Ayache, N.: An electromechanical model of the heart for image analysis and simulation. IEEE Trans. Med. Imaging 25(5), 612–625 (2006)
12. Komodakis, N., Tziritas, G., Paragios, N.: Fast, approximately optimal solutions for single and dynamic mrfs. In: CVPR 2007. IEEE Conference on Computer Vision & Pattern Recognition, IEEE Computer Society Press, Los Alamitos (2007)
13. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: SIGGRAPH 1986. Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, pp. 151–160 (1986)
14. Glocker, B., Komodakis, N., Paragios, N., Tziritas, G., Navab, N.: Inter and intramodal deformable registration: Continuous deformations meet efficient optimal linear programming. In: IPMI 2007. Information Processing in Medical Imaging (2007)
15. Gerig, G., Jomier, M., Chakos, M.: Valmet: A new validation tool for assessing and improving 3d object segmentation. In: Niessen, W.J., Viergever, M.A. (eds.) MICCAI 2001. LNCS, vol. 2208, pp. 516–523. Springer, Heidelberg (2001)

Teniæ Coli Detection from Colon Surface: Extraction of Anatomical Markers for Virtual Colonoscopy

Julien Lamy* and Ronald M. Summers

Building 10, Room 1C368X, 10 Center Drive MSC 1182, Bethesda, MD 20892-1182, USA
rms@nih.gov,
julien.lamy@urs.u-strasbg.fr

Abstract. The teniæ coli, three longitudinal bands of muscle on the colon surface, are important anatomical structures for several virtual colonoscopy applications: colon registration, virtual dissection, navigation and polyp matching. The current detection method involves manually tracing one of the teniæ on the colon surface, a long and error-prone process. In this paper, we present a semi-automated detection method, which only requires the user to reconnect a few long segments of teniæ. This method is based on local parameterizations of the colon surface, and on the application of classical image processing algorithms on the graph created by the triangular mesh of the surface. The results, computed on four test cases, are promising.

1 Introduction

The teniæ coli, shown on Fig. 1, three longitudinal bands of muscle on the colon surface, are an important tool for two CAD applications in the field of virtual colonoscopy, allowing a better polyp detection, and thus lowering the prevalence of colon cancer.

They are used to perform prone-to-supine colon lumen registration [1], serving as anatomical markers on the colon surface to define a deformation field of that surface.

They are also used for virtual dissection and navigation in the colon lumen [2], defining a circumferential frame of reference to synchronize prone and supine colon flythroughs.

The current detection method [3] requires the user to manually place markers every 2 to 10 cm along the tenia omentalis (the easiest to spot). This manually detected tenia is then the cutting line to generate a virtual dissection by a parameterization of the surface [4]. On this virtual dissection, the tenia mesocolica and the tenia libera are approximated by straight lines at one third and two thirds of the circumference. This method does not reflect the anatomical reality,

* Current address: LSIIT, UMR 7005 CNRS-ULP, Pôle API, boulevard Sébastien Brant, BP 10413, F-67412 Illkirch Cedex, France.

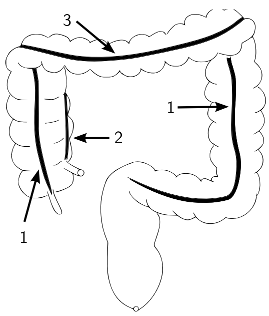


Fig. 1. Colon and teniæ coli. 1: Tenia libera. 2: Tenia mesocolica. 3: Tenia omentalis.

where the teniæ are not strictly equidistant, and this discrepancy is amplified by the distortions related to the parameterization.

In this paper, we attempt to overcome these limitations by providing a semi-automated method to detect the teniæ coli from the colon surface, exploiting the anatomical charateristics of those structures.

2 Methods

Our model of the teniæ coli takes into account the main characteristic of the teniæ: they are located where the haustral folds meet. Using this information, a detection of the haustral folds, and of their extremities, is the foundation of our teniæ detection algorithm.

We chose to work on the triangular mesh of the segmented colon [5], as the folds and teniæ are deformations on that surface. Such deformations are easier to characterize on the 2-manifold of the surface than on the 3D slices of the CT-Scan.

Our algorithm is based on two main techniques: a local parameterization of the colon surface, and the application of classical image processing techniques (e.g. mathematical morphology) to the graph defined by the triangular mesh of the surface [6].

2.1 Local Parameterization of the Colon

As the colon is a tube-like structure, on each point of the surface, we have two main, intuitive directions: a longitudinal direction (going towards the rectum or the cecum), and a circumferential direction (stay at the same place on the longitudinal axis, but go around the surface).

Our local parameterization of the colon is performed on a ring-like subset of the surface, defined using the centerline of the colon [7,8], shown on Fig. 2. Given a surface point $\mathrm{p}_{\mathrm{start}}$ around which to compute the parameterization, we first compute p_{c}, the closest centerline point to $\mathrm{p}_{\mathrm{start}}$. We then compute the Frenet frame, a local frame formed by the tangent, the normal and the binormal, at that centerline point. A region growing is then performed on the surface, seeded at $\mathrm{p_{start}}$. A candidate point is added to the region if its distance along the tangent axis from $\mathrm{p_{start}}$ is less than the user-defined ring size. This ring size is the longitudinal extent of the region. The full algorithm to compute such a ring is given in Algorithm 1.

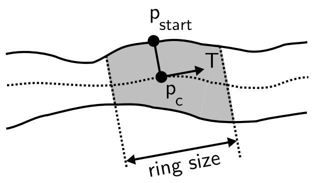


Fig. 2. Ring on the colon surface, showing the starting surface point, its closest centerline point, the tangent on the centerline, the ring size and the grown region

```
**Data**: centerline, surface, $\mathrm{p_{start}}$, ring size
$\mathrm{p_c} \leftarrow$ closest centerline point to $\mathrm{p_{start}}$;
compute Frenet frame $(\mathbf{T}, \mathbf{N}, \mathbf{B})$ at $\mathrm{p_c}$;
add $\mathrm{p_{start}}$ to queue;
while queue is not empty do
    $\mathrm{p_{current}} \leftarrow$ front of queue;
    for every direct neighbor n of $p_{current}$ on surface do
        if $|(n_{frenet})_{\mathbf{T}}| < \frac{ring\ size}{2}$ then
            add n to ring;
            add n to queue;
        end
    end
end
```

Algorithm 1. Ring algorithm

In areas of high curvature or torsion, the region growing can overflow in such a way that several centerline segments are contained within the ring, as shown on Fig. 3. In this case, the influence zones of each centerline segment in the ring are computed. The zone containing $\mathrm{p_{start}}$ is kept, while all others are discarded. This ring correction algorithm is given in Algorithm 2.

```
**Data**: centerline, ring, $\mathrm{p_{start}}$
for every segment C of the centerline contained in the ring do
    compute $\mathrm{IZ_C}$, the influence zone of C in the ring, i.e. all points which are
    closer to C than to any other segment;
    if $p_{start}$ is not contained in $IZ_C$ then
        discard $\mathrm{IZ_C}$;
    end
end
```

Algorithm 2. Ring correction algorithm

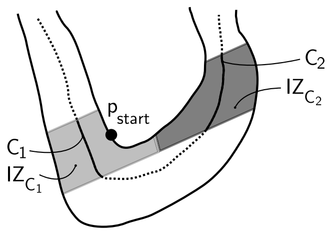


Fig. 3. Correction of an overflow, showing an overflowed ring containing two centerline segments and their respective influence zones. In this case, only the light gray influence zone, containing $\mathrm{p_{start}}$, will be kept.

Each point in a ring can then be expressed using two coordinates, (t, α), where t is the coordinate of the point on the tangent axis, and α is the angle of the point projected in the normal plane of the Frenet frame, spanned by the normal and the binormal. This gives us a parameterization of the 2-submanifold created by the ring. This parameterization corresponds to the two intuitive axes, the longitudinal and the circumferential, mentioned above.

2.2 Folds Detection

The haustral folds have two main anatomical characteristics which we exploit in our method:

- they have a hyperbolic curvature, i.e. the principal curvatures have oppposite signs;
- they are thin and elongated structures, each fold occupying about one third of the colon circumference, and having a small longitudinal extent.

A flowchart for our fold detection method is shown in Fig. 4.

A coarse set of surface vertices belonging to haustral folds is obtained by a simple threshold on the mean curvature of the surface points, as mentionned in [2]. This threshold is $k_H \in [-4, -0.5]$.

From this coarse set of points, we then compute the haustral folds by forming connected components and discarding those which do not meet the anatomical characteristics of folds. The connected components are extracted from the coarse set of surface vertices, which is first filtered by morphological operations. The classical morphological filtering (opening then closing) is computed on the graph spanned the triangular mesh of the surface. The structuring element is defined to be the 1-ring neighborhood around each surface vertex, i.e. its immediate graph neighbors.

If a connected component does not satisfy any of the following criteria, each of them representing an anatomical characteristic of the haustral folds, this component is discarded :

- Angle range: after parameterization, the angle range of all points in the component must be $\frac{2\pi}{3} \pm \varepsilon_r$.

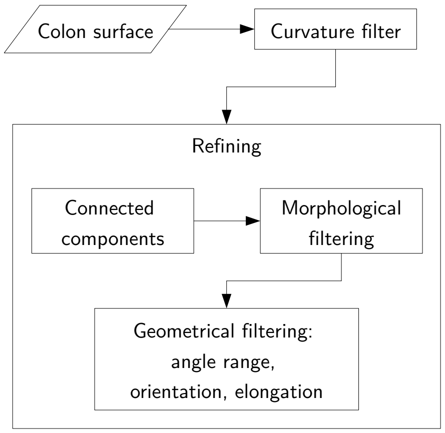


Fig. 4. Flowchart of fold detection

– Orientation: the component must have a circumferential orientation, i.e. it must be orthogonal to the centerline. The angle between the line passing through the points of minimal and maximal angle, and the tangent to the closest centerline point must be $\frac{\pi}{2} \pm \varepsilon_o$
– Elongation: the ratio of the lengths of the principal and secondary axes of the component must be greater than $e_{\min}$.

In order to compute the elongation of the component, we perform a principal components analysis on the parameterized connected component. As the parameterized component is a 2D structure, expressed in (t, α) coordinates, the PCA yields two axes. When the component is transformed in the eigenvectors basis, its elongation is measured using the ratio of the sizes of its bounding box. The result of this fold detection method is shown on a virtual dissection of the colon in Fig. 5, with one color per detected fold. Note that this virtual dissection is only used for an easier visualization: the method is applied on the non-dissected, original, colon surface.

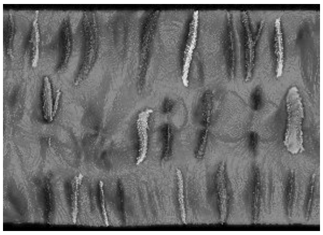


Fig. 5. Detected folds

2.3 Teniæ Detection

As mentioned above, the teniæ are located where the folds meet. In order to detect the teniæ, we first compute the extremities of the previously detected folds along the longitudinal axis, i.e. the points which have the minimal and the maximal angle values in the parameterized fold. We then cluster those points to form the teniæ.

The clusters are formed by an iterative process, mimicking the manual process of [2]: starting from the marker (i.e. fold extremity) closest to the cecum, we compute a ring around this marker. A set of candidates for the next marker is formed by the markers in the ring satisfying the following conditions :

- the candidate is not yet part of a cluster,
- the candidate is further on the centerline,
- in the parameterized ring, the candidate has an α coordinate value similar to the one of the current marker

If the candidates set is empty, a new cluster is started. Otherwise, the closest candidate to the current marker is added to the cluster, and that closest candidate becomes the current marker.

Once a cluster is formed, the vertices composing it are joined by the shortest path [9] on the surface graph, forming segments of teniæ. Ideally, we will obtain three clusters and three segments, one per tenia. The full algorithm is given in Algorithm 3. If more than three segments are formed, they can then be manually reconnected.

As the teniæ vanish at the rectum, a manual marker is placed by the user to limit the tenia propagation, thus avoiding the algorithm to behave incorrectly.

2.4 Numerical Values

Several user-defined numerical values are mentioned in the previous sections. We give here the values we used, and the method used to obtain them.

- Angle range of parameterized component, $\varepsilon_r = \frac{2\pi}{9}$.
- Max angle of principal axis to circumferential axis: $\varepsilon_o = 10$ degrees.
- Minimum elongation: $e_{\min} = 2$.
- Ring size for clustering: 5 cm
- Max angle difference for clustering: $\theta_{\max} = \frac{3\pi}{18}$.

The values of ε_r, ε_o and $\theta_{\max}$ were measured on virtual dissections of the colon. These virtual dissections were obtained by using the parameterization of the colon surface presented in [2]. To that flat parameterization, we added height information for easier visualization. On each surface vertex, the height was defined to be the distance from that vertex to the nearest centerline point.

```
Data: Markers: extremities of folds, sorted by distance along the centerline
for every marker $m_{start}$ do
    if $m_{start}$ is already colored then
        skip it
    else
        start new cluster at $\mathrm{m_{start}}$;
        $\mathrm{m_{current}} \leftarrow \mathrm{m_{start}}$;
        while cluster is not finished do
            add $\mathrm{m_{current}}$ to current cluster;
            compute local parametrization of surface around $\mathrm{m_{current}}$;
            for every marker m ahead of $m_{current}$ on centerline do
                if $|m_\alpha - (m_{current})_\alpha| < \theta_{\max}$ then
                    add m to candidates;
                end
            end
            if $candidates \neq \emptyset$ then
                $\mathrm{m_{start}} \leftarrow$ closest candidate;
            else
                connect all cluster points by shortest path;
                mark the cluster as finished;
            end
        end
    end
end
```

Algorithm 3. Clustering of tenia candidates

3 Results

Our test base consisted of four cases for which a fully segmented colon, a centerline and a virtual dissection (to assess the quality of the results) could be obtained. Our algorithm was run on each of those four test cases, for a runtime of approximately half an hour. Figure 6 shows the detected teniæ as green lines on an endoluminal view of the colon. Figure 7 shows the individual segments on a virtual dissection.

The number of segments per tenia and per case is presented in Table 1. On these cases, we have an average of 8.67 segments per tenia, which yields an easy reconnection.

Table 1. Number of segments per tenia per case

Case	Tenia 1	Tenia 2	Tenia 3
Case 1	8	11	7
Case 2	8	8	8
Case 3	8	7	9
Case 4	9	11	10

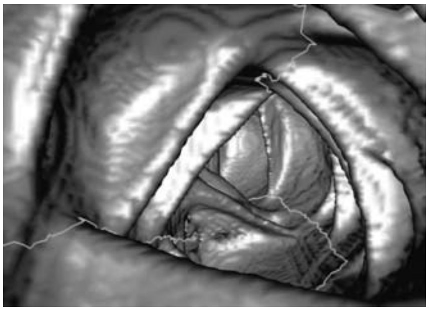


Fig. 6. Teniæ from inside the colon

Those promising results are however hampered by regions where folds cannot be detected: this will form more clusters, involving more manual reconnections. Moreover, less folds detected might cause a loss of precision, as the shortest path between two consecutive markers is able to stray from the teniæ.

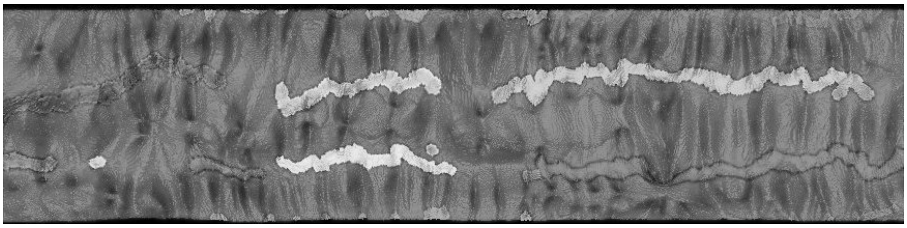


Fig. 7. Teniæ segments

4 Future Work

A better tenia detection is conditioned by a better fold detection. There are two main drawbacks to our current fold detection algorithm: merged folds and split folds. When several folds are merged into one component, the elongation criterion is not satisfied. On the other hand, when a fold is split, each sub-component does not have a large enough angle range and is discarded. We are currently investigating morphological processing methods to solve these drawbacks.

Two steps are required to obtain a fully automated algorithm: an automated detection of the rectum, and an automated reconnection of the teniæ segments. In our opinion, the segments reconnection will be solved with a better folds detection. The automated detection of the rectum, and by extension of the different segments of the colon will be the subject of further research, as this segmentation could help increasing the quality of the processings for several other methods by specializing the behavior according to the current colon segment.

5 Conclusion

We have presented a semi-automated method to detect the teniæ coli from the colon surface. Our method currently requires two user inputs, one giving the location of the rectum, and the other involving reconnecting a few tenia segments. We have tested this method on four cases, and have been able to detect the teniæ. Future work includes a better detection of haustral folds and a partition of the colon in its anatomical segments.

Acknowledgements

The authors wish to thank Dr. William Schindler for the virtual colonoscopy data and Marius Linguraru for helpful discussions. This research was supported by the Intramural Research Program of the NIH, Clinical Center.

References

1. Lamy, J., Summers, R.M.: Intra-patient colon surface registration based on teniæ coli. In: SPIE Medical Imaging 2007: Computer-Aided Diagnosis. Proceedings of the SPIE, vol. 6514, pp. 116–123 (2007)
2. Huang, A., Roy, D., Franaszek, M., Summers, R.M.: Teniae coli guided navigation and registration for virtual colonoscopy. In: IEEE Visualization, pp. 279–285. IEEE Computer Society Press, Los Alamitos (2005)
3. Huang, A., Roy, D.A., Summers, R.M., Franaszek, M., Petrick, N., Choi, J.R., Pickhardt, P.J.: Teniae colibased circumferential localization system for CT colonography: Feasibility study. Radiology 243, 551–560 (2007)
4. Floater, M.: Parametrization and smooth approximation of surface triangulations. Computer Aided Geometric Design 14(3), 231–250 (1997)
5. Franaszek, M., Summers, R.M., Pickhardt, P.J., Richard, C.J: Hybrid segmentation of colon filled with air and opacified fluid for CT colonography. IEEE Transactions on Medical Imaging 25, 358–368 (2006)
6. Vincent, L.: Mathematical morphology for graphs applied to image description and segmentation. In: Proceedings of the Electronic Imaging West Conference, pp. 313–318 (1989)
7. Lamy, J., Ronse, C., Soler, L.: Loop removal from colon central path through skeleton scale-space tracking. In: Bebis, G., Boyle, R., Koracin, D., Parvin, B. (eds.) ISVC 2005. LNCS, vol. 3804, pp. 68–75. Springer, Heidelberg (2005)
8. van Uitert, R., Bitter, I., Franaszek, M., Summers, R.M.: Automatic correction of level set based subvoxel accurate centerlines for virtual colonoscopy. In: Proceedings of the 2006 IEEE International Symposium on Biomedical Imaging, pp. 303–306. IEEE Computer Society Press, Los Alamitos (2006)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)

A Quantitative Object-Level Metric for Segmentation Performance and Its Application to Cell Nuclei

Laura E. Boucheron[1,2], Neal R. Harvey[2], and B.S. Manjunath[1]

[1] University of California Santa Barbara, Electrical and Computer Engineering, Santa Barbara, CA 93106-9560

[2] Los Alamos National Laboratory, Space and Remote Sensing Sciences, P.O. Box 1663, Los Alamos, NM 87545

Abstract. We present an object-level metric for segmentation performance which was developed to quantify both over- and under-segmentation errors, as well as to penalize segmentations with larger deviations in object shape. This metric is applied to the problem of segmentation of cell nuclei in routinely stained H&E histopathology imagery. We show the correspondence between the metric terms and qualitative observations of segmentation quality, particularly the presence of over- and under-segmentation. The computation of this metric does not require the use of any point-to-point or region-to-region correspondences but rather simple computations using the object mask from both the segmentation and ground truth.

1 Introduction

The subject of objective and quantitative evaluation of segmentation performance has received less attention than has the development of various segmentation algorithms themselves. This has been noted by many researchers in the fields of computer vision and image analysis [1,2,3,4,5,6,7].

In our development of a quantitative metric, we avoid metrics of segmentation performance that rely on point-to-point or region-to-region correspondences (e.g., [2,8,9]). We also avoid empirical goodness metrics, as defined in [6], whereby properties of a "good" segmentation are defined a priori according to human perception of a "good" segmentation; in our application domain of cancer imagery it is difficult to define a single model which applies to all image objects. While there is much research in the use of multiple ground truths, often manually defined by multiple human experts, we stick to the case of one ground truth assumed to be the gold standard and quantify the segmentation performance at the level of image objects. This falls under the empirical discrepancy metrics as defined in [6].

We present here our research on the segmentation of cell nuclei in routine H&E stained histopathology imagery. In our use of the term "segmentation," we are referring to an object-level segmentation, i.e., a delineation of individual nuclei, thus more standard metrics such as Receiver Operating Characteristics (ROC) curve analysis are not directly applicable. The pixel-level classification of nuclei pixels is described in our previous work [10].

In Section 2 we will first describe an object-level metric for segmentation accuracy (Section 2.1) as well as an analysis of the variation of our metric with segmentation quality (Section 2.2), a methodology for specification of object-level ground truth (Section 2.3), and a summary of our work on the segmentation metric (Section 2.4). We briefly discuss the application of our shape-based segmentation metric to non-elliptical objects in Section 3. We then present results for example nuclei segmentations in Section 4, including a standard watershed-based segmentation (Section 4.1), a combined shape-based and watershed-based segmentation (Section 4.2), and summarize our segmentation results (Section 4.3). Conclusions are presented in Section 5.

2 Segmentation Metric

The following metric was defined with the segmentation of cell nuclei, i.e., roughly circular or elliptical objects, in mind. For the segmentation of cell nuclei, we wish to penalize not only the size of regions missed and extraneous regions, but also the shape of those same regions. Additionally we include terms to penalize over- and under-segmentation. We introduce the quadrant sum as a method of quantifying deviation in shape from the ground truth by comparing the mass across two orthogonal axes through the object's center of mass. While this section will focus on elliptical objects, we will show the use of the quadrant sum for arbitrarily shaped objects in Section 3.

2.1 Definition

We define our segmentation metric as:

$$\begin{aligned} P = &\frac{1}{N_D}\sum_{i=1}^{N_D}\max\Bigg(0, \Bigg[1-\alpha_1\frac{SR-1}{\delta_{SR}} - \alpha_2\frac{1}{1.75}\left(\frac{PM}{GT}+\frac{2QS_{PM}}{GT}\right) \\ &-\alpha_3\frac{1}{1.75}\left(\frac{EP}{GT}+\frac{2QS_{EP}}{GT}\right)\Bigg]\Bigg)\cdot\left(1-\alpha_4\frac{N-N_D}{N}\right)-\alpha_5\frac{ER}{N\cdot\delta_{ER}} \end{aligned} \tag{1}$$

where

$$0 \le \alpha_i \le 1,\ i=1,\ldots,5\ .$$

Taking each additive term Equation 1, we will define all variables. N is the number of ground truth nuclei defined in the user markup and N_D is the number of nuclei detected by the segmentation algorithm; thus the summation averages scores for individual nuclei. We penalize for each nucleus detected[1]:

[1] For the sake of clarity and brevity we have not included in Equation 1 the necessary clipping functions to assure that each term is less than 1. We will discuss the need for these clipping functions and explicitly display them in the discussions of individual terms to follow.

1. *The number of segmented regions:*

$$\text{term}_1 = \alpha_1 \min\left(1,\ \frac{SR-1}{\delta_{SR}}\right) \tag{2}$$

We define SR as the number of segmented regions overlapping the ground truth nucleus, and δ_{SR} as the upper limit for number of segmented regions. For a perfect segmentation there would be only one segmented region per ground truth region and $\delta_{SR} = 1$ would be an intuitive value for evaluation of very good segmentations; we leave this as a parameter, however, to allow for comparison of poorer segmentations with more tendency to oversegment. We use the minimum function to clip this term to a maximum value of 1. Overall, the weight α_1 can be thought of as the penalty for an oversegmented nucleus, similar to the oversegmentation term of [4].

2. *The size and shape of the region of pixels missed:*

$$\text{term}_2 = \alpha_2 \min\left(1, \frac{1}{1.75} \cdot \left(\frac{PM}{GT} + \min\left(1,\ \frac{2 \cdot QS_{PM}}{GT}\right)\right)\right) \tag{3}$$

We define PM as the number of pixels missed: pixels belonging to the ground truth markup of the nucleus, but missed by the segmentation algorithm. GT is the number of pixels in the ground truth markup, thus, $\frac{PM}{GT}$ quantifies the size of the region of missed pixels. This is similar to the percentage of misclassified pixels used in [5].

We also look at the spatial distribution of the missed pixels, since we wish to penalize certain spatial distributions more than others. For example, a distribution of missed pixels in an annulus about the centroid of the nucleus will affect the shape and other higher-level feature statistics far less than a distribution of missed pixels encompassing half of the nucleus. Note that this is a different approach than a simple pixel distance error as in [5] and tends towards an appreciation of accurate higher-level measurements as in [7]. We take the "quadrant sum" of the pixels missed, QS_{PM} as follows:

$$QS_{PM} = \|r_1 + r_3 - r_2 - r_4\| + \|r_1 + r_2 - r_3 - r_4\| \tag{4}$$

where r_i are the number of pixels in the $i = 1, 2, 3, 4$ quadrants:

$$\begin{aligned} r_1 &= \sum \left\| e^{j\theta_{PM}} \right\|, \quad \text{for } 0 < \theta_{PM} < \frac{\pi}{2} \\ r_2 &= \sum \left\| e^{j\theta_{PM}} \right\|, \quad \text{for } \frac{\pi}{2} < \theta_{PM} < \pi \\ r_3 &= \sum \left\| e^{j\theta_{PM}} \right\|, \quad \text{for } -\frac{\pi}{2} < \theta_{PM} < -\pi \\ r_4 &= \sum \left\| e^{j\theta_{PM}} \right\|, \quad \text{for } 0 < \theta_{PM} < -\frac{\pi}{2} \end{aligned} \tag{5}$$

where θ_{PM} is the angle of the polar representation of the pixels missed. Thus, QS_{PM} is a measure of symmetry about the x- and y-axes of the region with the origin at the grouth truth centroid. Due to the discrete nature of the regions, it is possible that QS_{PM} may slightly exceed $\frac{GT}{2}$; to compensate for this, we take the minimum of 1 and $\frac{2 \cdot QS_{PM}}{GT}$.

Overall, α_2 can be thought of as the penalty regions of pixels missed, penalizing both size and shape. More details of the performance of this QS term is explained in Figure 1 for circular and elliptical regions, including the motivation for our normalization factor of 1.75. While this is a simple and easy to compute metric, there is no reason why another shape metric could not be substituted, with appropriate attention to the inclusion of the size metric.

3. *The size and shape of the region of excess pixels:*

$$\text{term}_3 = \alpha_3 \min\left(1, \frac{1}{1.75} \cdot \left(\min\left(1, \ \frac{EP}{GT}\right) + \min\left(1, \ \frac{2 \cdot QS_{EP}}{GT}\right)\right)\right) \quad (6)$$

Similar to term 2, we define EP as the number of excess pixels: pixels segmented as part of the nuclear region that do not correspond to the ground truth markup. We quantify the size of the region of extra pixels by $\frac{EP}{GT}$. We also take the "quadrant sum" of the excess pixels, QS_{EP} and normalize by $\frac{GT}{2}$. Again, we take the minimum of 1 and $\frac{2 \cdot QS_{EP}}{GT}$ and normalize the sum of the two factors by 1.75. α_3 is thus the penalty for size and shape of excess pixel regions, and is related to the degree of undersegmentation of the nucleus.

Averaging these three terms provides a measure of the segmentation performance on all detected nuclei. We also wish to weight this average by the general detection rate. Thus we scale the average of the previous three terms by:

4. *The fraction of nuclei detected:*

$$\text{term}_4 = 1 - \alpha_4 \frac{N - N_D}{N} \quad (7)$$

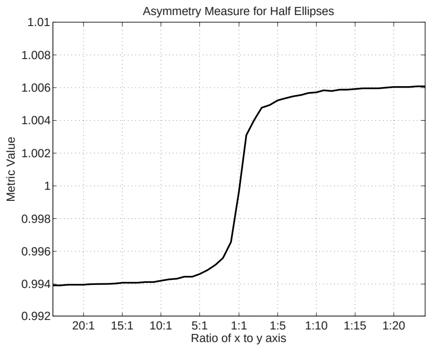
This term with $\alpha_4 = 1$ would simply be the detection rate. We leave this as a parameter since in the segmentation of nuclei, we are interested more in the accuracy of the nuclei that are segmented than in the actual detection rate. This harkens back to the theory of Ultimate Measurement Accuracy [7], wherein it is the accuracy of further image analyses that determine the accuracy of the underlying segmentation.

Finally we wish to penalize over the whole region of ground truth:

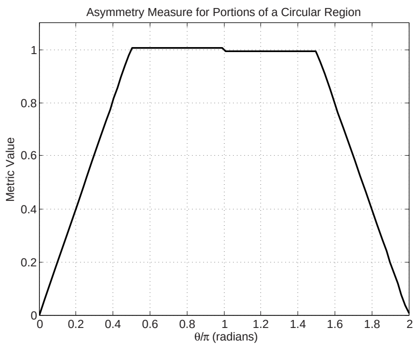
5. *The number of extra segmented regions:*

$$\text{term}_5 = \alpha_5 \min\left(1, \ \frac{ER}{N \cdot \delta_{ER}}\right) \quad (8)$$

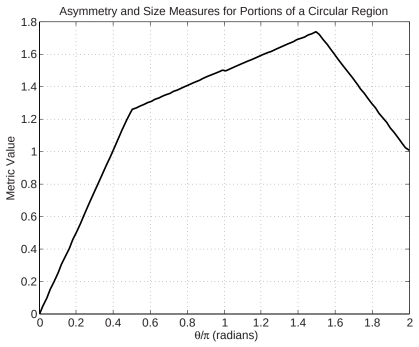
This term looks at the excess segmented regions that have no correspondence to a ground truth nucleus. We define ER as the number of excess segmented regions and δ_{ER} as the fraction of total ground truth nuclei that we will allow as excess regions. α_5 is, therefore, the penalty for excess segmented regions, similar to the concept of noise in [4].



(a) Effect on the QS metric of ellipticity and orientation of missed pixels. The region missed is below the x-axis: ellipses plotted to the left of circular are missing half of their area along the major axis and to the right of circular, half their area along the minor axis. Here we note the possibility for the metric to be slightly larger than 1.



(b) Effect on the QS metric of the portion of a circular region of pixels missed. The maximum value for this metric occurs at (and around) $\theta = \pi$, when half of the region is missed. The metric tapers off to zero for small and large angles; this illustrates the need for a separate size metric, since this metric is scoring only the asymmetry.



(c) Additive effect of the QS and size metrics. The combination of these two metrics yields the desired penalty. Note the maximum value of ~ 1.75.

Fig. 1. Values of the QS metric for pixels missed in discrete elliptical and circular regions. The QS metric in these plots has been normalized by $\frac{GT}{2}$, and the size metric by GT.

Overall, the choice of α_i reflects a weighting of the relative importance of the various penalties. Similarly, the choice of δ_{SR} and δ_{ER} reflects a choice in the latitude given to certain errors in segmentation. A reasonable choice for default parameters would be $\alpha = [0.5\ 0.5\ 0.5\ 1\ 0.5]$, $\delta_{SR} = 1$, and $\delta_{ER} = 1$, reflecting an equal penalty for under- and over-segmentation errors (α_1, α_2, and α_3), a direct weighting by the detection rate (α_4), equal importance given to the correct detection and segmentation of cell nuclei and the avoidance of erroneously detected and segmented nuclei (α_5), one segmented region allowed per nucleus (δ_{SR}), and weighting of the erroneously segmented regions proportional to the total number of cell nuclei (δ_{ER}). It is important to note, however, that while the choice of these parameters will effect the absolute values of the metric terms, a direct comparison of segmentation performance for different algorithms may be achieved with any reasonable parameter choice.

2.2 Metric Variation Versus Segmentation Quality

We apply our segmentation metric (Equation 1) to the watershed transform of the complemented Euclidean distance transform (WSCDT) of a thresholded red channel for an example image. The threshold is varied over the entire range of values it can assume, [0,255], and we retain all pixels less than the threshold. The use of the red channel is due to the high contrast for nuclei in this channel.

We compute the WSCDT as follows:

1. Compute the negative of the Euclidean distance transform on the complemented binary image, setting the distance of all background pixels in the binary image to a depth of $-\infty$.
2. Compute the watershed transform on the resulting distance transform.

By varying the threshold we compute a variety of binary images. We compute the segmentation metric (Equation 1) of the WSCDT segmentation of these binary images to gain a sense of the expected variation in our metric for a range of segmentation possibilities. These possibilities include the two extremes whereby either all or none of the pixels has been classified as nuclei. We display the performance of the individual metric terms as well as the overall performance in Figure 2. It is important to note that we are plotting the *performance* of the individual terms rather than the terms themselves; thus we are plotting the subtraction of each term from a value of 1.

We see in Figure 2 that the performance is zero for both extremes of the threshold classification. Observation of individual terms shows expected trends, namely that:

– Term 1 (extra GT regions) decreases in performance as the threshold increases. This is due to the thresholded nuclei regions becoming larger with more complicated boundaries which results in the distance transform having multiple minima per connected component.
– Term 2 (pixels missed) increases in performance as more pixels are attributed to nuclei. The dip in performance at high thresholds is due to an assumption that the largest watershed region is the background which becomes invalid as nearly the entire image is classified as foreground.
– Term 3 (extra pixels) decreases in performance as nuclei tend to merge in the binary thresholded image.
– Term 4 (nuclei detected) increases in performance as more pixels are attributed to nuclei.

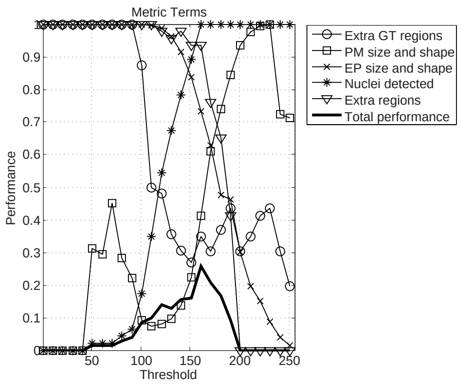


Fig. 2. Metric variation versus segmentation quality for an example image. The red channel was thresholded, retaining all pixels less than the threshold, and was then segmented with the WSCDT method. It should be noted that all terms plotted here are performance, i.e., one minus the penalty, where the penalties are the terms previously discussed in relation to the segmentation metric. The terms are denoted by a brief description in the legend, but they are also plotted in numerical order, i.e., blue circles are term 1, red squares are term 2, and so forth.

– Term 5 (extra regions) decreases in performance as more extraneous regions are thresholded as nuclei. The performance of this term returns to 1 for a threshold of 256, since there are no longer any extraneous regions; this is not apparent in Figure 2 since we have downsampled the plot for less clutter.

We note here that analysis of the individual metric terms is useful for quantifying segmentation, but we have integrated the terms into one metric to allow for single parameter to compare and/or optimize between different segmentations.

2.3 Ground Truth Image Markup Within a Truth Window

While it is easy to specify a pixel-level markup within a designated truth window, such a specification becomes more complicated with an object-level markup. In a pixel-level markup, an object that spans the truth window boundary can be marked up to the boundary without losing any important information for the overall classification. In an object-level markup, however, the actual extent and border of the object is of utmost importance. Moreover, if objects are marked within a rough concept of a truth window, the truth window may contain parts of objects that have not been delineated by the user. This will lead to erroneously low performance since the segmentation metric will assume that these regions were incorrectly segmented as image objects.

To help alleviate this problem, after the delineation of objects within the truth window is complete, the truth window is recomputed as the minimum bounding rectangle of the object markups. Using this new truth window, the user is asked to mark a minimum of one point for each unmarked object that is either completely or partially enclosed by the new truth window. This information is used in a connected-components analysis to determine if extra segmented regions are associated with an object that has not been delineated in the ground truth markup.

2.4 Summary

We have presented a general segmentation metric computed on an object level. This metric uses simple quantities that are easy to compute using the segmentation and ground truth masks, namely the regions of pixels missed by the segmentation and the regions of extra pixels not associated with a ground truth region. We have also shown the variation in this metric for a variety of segmentations using a simple watershed-based segmentation (WSCDT) and discussed the ground truth markup process for evaluation of the metric.

3 Application of the QS Shape Metric to Non-elliptical Objects

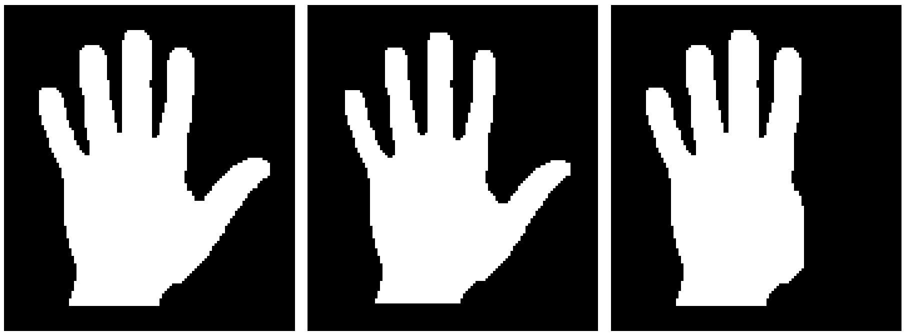
We would like to briefly discuss the applicability of the QS metric to non-elliptically shaped objects; we will be using the concepts of the PM QS metric, but the arguments are identical for the EP case. The use of the centroid of the ground truth object is what allows this metric to work for irregularly shaped objects. For a planar object with uniform density, the mass (number of pixels in our case) will be equal across any arbitrary line through the center of mass (equivalent to the centroid in the uniform density case). By defining orthogonal axes through the centroid, we can eliminate the chance of the arbitrary line corresponding to a reflectional symmetry of the region of pixels missed. We show an example of the application of the PM QS metric in Figure 3 for a hand silhouette. Further research into the use of our segmentation metric for arbitrarily-shaped objects is currently ongoing. In particular, the practical application of this metric may warrant a different normalization value than the theortical maximum of ~ 1.75 for the combined size and shape metrics for EP and PM.

4 Watershed-Based Segmentation

We investigate here two simple watershed-based segmentation methods for delineation of cell nuclei. We assign the default weights (discussed in Section 2.1) of $\alpha = [0.5\ 0.5\ 0.5\ 1\ 0.5]$, $\delta_{SR} = 1$, and $\delta_{ER} = 1$.

4.1 Watershed on the Complemented Distance Transform

We use the WSCDT method, as described in Section 2.2, on the pixel-level nuclei classifications from [10]. We present quantitative results in Table 1 and an example segmentation in Figure 4. The results in Table 1 represent the results averaged over the entire dataset of 58 images. The example semgnetaions in Figure 4 include the performance for the single example image.



(a) Original hand silhouette with $GT = 5270$ object pixels.

(b) Erosion by 1 pixel; total of $PM = 524$ pixels eroded (missed). $\frac{2 \cdot QS}{GT} = 0.188$, $\frac{PM}{GT} = 0.099$, $term_2 = 0.107$.

(c) Thumb removed; total of $PM = 524$ pixels missed. $\frac{2 \cdot QS}{GT} = 0.397$, $\frac{PM}{GT} = 0.099$, $term_2 = 0.227$.

Fig. 3. Application of the QS and size metrics to an example silhouette and "segmentations." Qualitatively, the segmentation in (b) retains more resemblance to the original silhouette in (a) than does the segmentation in (c), where the entire thumb is missed. A size metric alone would rank the two results in (b) and (c) as equally good segmentations, while the use of the QS metric penalizes the change in shape of (c). Note that in (b) the addition of the shape metric does not change the value of the original size-based metric by much (0.8%).

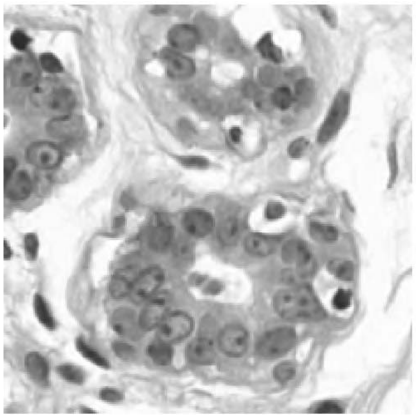
4.2 Marker-Based Watershed Segmentation

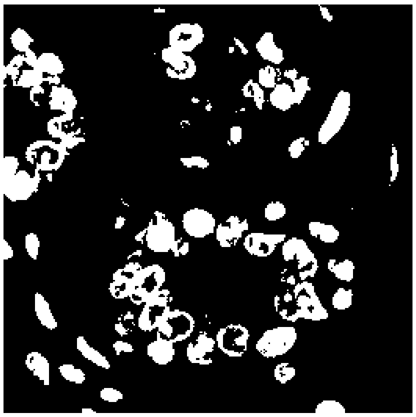
We use a prior assumption about the shape of cell nuclei, namely that they are roughly circular in shape and approximately the same diameter. Byun et al. [11] use an inverted Laplacian of Gaussian (LoG) filter for detection of nuclei in fluorescent confocal retinal imagery. For use in our brightfield imagery, we use a non-inverted LoG filter in the same "blobdetector" framework of [11].[2] We use the detection capabilities of this method as a seed for a subsequent watershed segmentation. This method (WSBlob) proceeds as follows:

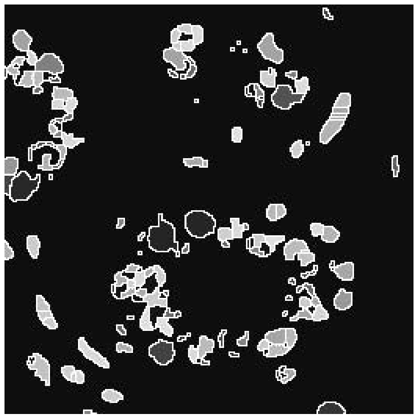
- Detect nuclei using the red channel of the imagery and use these locations as foreground markers for the watershed transform. A filter size of 25 pixels in diameter (average nucleus diameter) and an inter-blob distance of 12 (half the filter diameter) was empirically chosen.
- Use the eroded complement of the binary nuclei classification as background markers.

Quantitative results for the WSBlob method are presented in Table 1 and an example segmentation in Figure 4. As for the WSCDT, the results in Table 1 are averaged over the entire dataset and example segmentations in Figure 4 include the performance for the example image.

[2] Code available at `http://www.bioimage.ucsb.edu/software.html`


(a) Original RGB image.


(b) Original binary image (refer to [10] for more details).


(c) WSCDT, $P = 0.2974$.

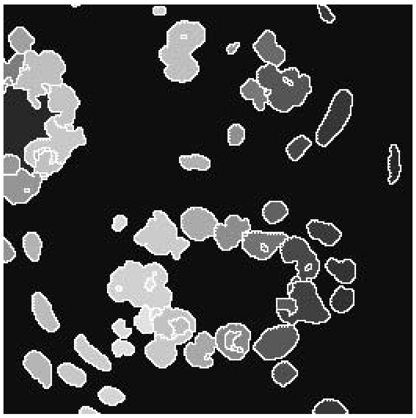

(d) WSBlob, $P = 0.3843$.

Fig. 4. Example watershed-based segmentations. Note the tendency of the WSCDT method to oversegment and the tendency of the WSBlob method to undersegment.

4.3 Summary

Referring to Table 1, we see terms 1 and 3 display the most difference between the two methods. In particular, WSCDT displays a worse performance for term 1 (extra GT regions) and better performance for term 3 (extra pixels), indicating a tendency for WSCDT to oversegment as compared to WSBlob which tends to undersegment. These observations are validated by observation of the example segmentations in Figure 4.

We have presented the application of our segmentation metric to the problem of segmentation of cell nuclei. We have shown the overall metric performance and the performance of individual terms for each segmentation method. We have also

Table 1. Nuclei segmentation term performance

Method	**P**	**term$_1$**	**term$_2$**	**term$_3$**	**term$_4$**	**term$_5$**
WSCDT	0.18	0.38	0.82	0.60	0.96	0.44
WSBlob	0.25	0.69	0.94	0.31	0.98	0.52

used the metric terms as a quantitative basis to compare the performance of the two methods, which corresponds well with the qualitative observations of general segmentation accuracy as illustrated by the example segmentations.

Further work should include observer studies to correlate the metric to the visual assessment of many individuals. Additionally, future and ongoing research includes the comparison of our metric to other applicable metrics, e.g., those presented in [2] and [9].

5 Conclusions

We have presented an object-level segmentation metric and its constituent terms and have shown that they correspond well with the qualitative observations of segmentation accuracy, including the general tendency of an algorithm to over- or under-segment an image. This metric also allows for a direct quantitative comparison between the outputs of different segmentation algorithms. While the metric defines a single performance, we have shown the usefulness of observing the performance of the individual metric terms.

We have also discussed a new method for specification of ground truth for this object-level segmentation problem. This involves not only the delineation of cell nuclei within an approximate truth window, but also the marking of non-delineated objects within the truth window. This allows us to focus our segmentation evaluation on only those objects that were delineated by the user.

In comparison to other work in segmentation evaluation, our metric does not require the computation of region or boundary correspondences which can be complicated. Instead we have introduced a metric based on simple subtrations of object masks and other object-level metrics (e.g., number of segmented regions). Additionally, we compute the segmentation performance on an object-by-object basis.

Acknowledgments

The authors gratefully acknowledge Karen Glocer and Ed Rosten for the fruitful technical discussions on and suggestions for this work. Many thanks also to Dr. David Rimm in the Department of Pathology at Yale University for the histopathology data.

References

1. Nattkemper, T.W.: Automatic segmentation of digital micrographs: A survey. In: Proc MEDINFO (2004)
2. Chalana, V., Kim, Y.: A methodology for evaluation of boundary detection algorithms on medical images. IEEE T. Med. Imaging 16, 642–652 (1997)
3. Udupa, J.K., LeBlanc, V.R., Zhuge, Y., Imielinska, C., Schmidt, H., Currie, L.M., Hirsch, B.E., Woodburn, J.: A framework for evaluating image segmentation algorithms. Comput. Med. Imag. Grap. 30, 75–87 (2006)
4. Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P.J., Bunke, H., Goldgof, D.B., Bowyer, K., Eggart, D.W., Fitzgibbon, A., Fisher, R.B.: An experimental comparison of range image segmentation algorithms. IEEE T. Pattern Anal. 18, 673–689 (1996)
5. Yasnoff, W.A., Mui, J.K., Bacus, J.W.: Error measures for scene segmentation. Pattern Recogn. 9, 217–231 (1977)
6. Zhang, Y.J.: A survey on evaluation methods for image segmentation. Pattern Recogn. 29, 1335–1346 (1996)
7. Zhang, Y.J., Gerbrands, J.J.: Segmentation evaluation using ultimate measurement accuracy. In: Proc. SPIE, vol. 1657, pp. 449–460 (1992)
8. Lezoray, O., Cardot, H.: Cooperation of color pixel classification schemes and color watershed: A study for microscopic images. IEEE T. Image Process 11, 783–789 (2002)
9. Cohen, L., Vinet, P., Sander, P.T., Gagalowicz, A.: Hierarchical region based stereo matching. In: Proc. CVPR, pp. 416–421 (1989)
10. Boucheron, L.E., Bi, Z., Harvey, N.R., Manjunath, B.S., Rimm, D.L.: Utility of multispectral imaging for nuclear classification of clinical histopathology imagery. BMC Cell Biology 8(Suppl 1):S8 (2007), `http://www.biomedcentral.com/1471-2121/8/S1/S8`
11. Byun, J., Verardo, M.R., Sumengen, B., Lewis, G.P., Manjunath, B.S., Fisher, S.K.: Automated tool for nuclei detection in digital microscopic images: Application to retinal images. Mol. Vis. 12, 949–960 (2006)

A Convex Semi-definite Positive Framework for DTI Estimation and Regularization

Radhouène Neji[1,2], Noura Azzabou[1], Nikos Paragios[1], and Gilles Fleury[2,⋆]

[1] GALEN Group, Laboratoire des Mathématiques Appliquées aux Systèmes, Ecole Centrale Paris, Châtenay-Malabry, France
[2] Département Signaux et Systèmes Electroniques, Ecole Supérieure d'Electricité, Gif-sur-Yvette, France

Abstract. In this paper we introduce a novel variational method for joint estimation and regularization of diffusion tensor fields from noisy raw data. To this end, we use the classic quadratic data fidelity term derived from the Stejskal-Tanner equation with a new smoothness term leading to a convex objective function. The regularization term is based on the assumption that the signal can be reconstructed using a weighted average of observations on a local neighborhood. The weights measure the similarity between tensors and are computed directly from the diffusion images. We preserve the positive semi-definiteness constraint using a projected gradient descent. Experimental validation and comparisons with a similar method using synthetic data with known noise model, as well as classification of tensors towards understanding the anatomy of human skeletal muscle demonstrate the potential of our method.

1 Introduction

Diffusion tensor imaging (DTI) is an emerging non-invasive modality allowing the quantitative investigation of water protons diffusion within biologic tissues. Since diffusion is sensitive to the presence of organized structures, DTI is used mostly in brain studies and has become a tool to infer white matter connectivity [1]. Such a modality offers measurements of the amount of diffusion of water molecules in several different directions. One then can infer the estimation of a tensor which is a 3×3 symmetric positive definite matrix representing the uncertainty on the position of water protons with a Gaussian model of displacement.

However, the DTI experimental protocol yields noisy observations due to the diffusion-sensitizing magnetic gradient. Furthermore, the clinical protocols refer to relatively low magnet strength, or a rather low signal-to-noise ratio. Therefore, signal reconstruction is crucial to obtain an appropriate estimate of the tensor field and for subsequent use of this estimate in applications like fiber tractography.

Several methods have been proposed to address diffusion tensor regularization. In [2], a two-step regularization was proposed consisting of the restoration

⋆ This work was partially supported by Association Française contre les Myopathies (AFM: `http://www.afm-france.org`).

of the principal diffusion directions using a total variation-model followed by the smoothing of the eigenvalues using an anisotropic tensor-driven formulation. In [3], the maximization of a log-posterior probability based on the Rician noise model is considered to smooth directly the diffusion-weighted images. A Bayesian model based on a Gaussian Markov Random Field was used in [4] to smooth the diffusion tensors. In [5], the authors consider the tensors as lying on a Riemannian manifold and use the corresponding distance to derive a local weighted averaging for DTI denoising. Tensors are assumed to be positive-definite matrices which was taken into account in [6] where an anisotropic filtering of the L^2 norm of the gradient of the diffusion tensor was considered and their proposed PDE scheme constrains the estimation to lie on this space. Such a concept was further developed in [7] where a variational method was proposed that aimed to minimize the L^p norm of the spatial gradient of the diffusion tensor under a constraint involving the non-linear form of Stejskal-Tanner equation. A non linear diffusion scheme is described in [8] where smoothing is made direction-dependent using a diffusion matrix in the PDE system. More recently, in [9] a joint reconstruction and regularization was proposed in the context of an energy minimization in a Log-Euclidean framework. The existing variational methods focused disproportionately on enforcing the positive-definiteness constraint, with the regularization term usually chosen as a function of the norm of the gradient. The main limitation of most of the above-mentioned methods is the nature of the cost function (non-convex) that entails a preliminary initialization step, while little attention was paid to defining appropriate smoothness components that account for the expected nature of tensors.

In this paper we propose a new variational approach to jointly estimate and regularize diffusion tensor images. We use a convex energy functional which combines the linearized form of Stejskal-Tanner equation as a data fidelity term and a new regularization term involving precalculated weights which measure the similarity between neighboring tensors. We show the results of our method both on synthetic datasets and real data of diffusion tensor muscle images.

2 DTI Estimation and Regularization

Let us assume that n DTI acquisitions $(S_k)_{k=1\dots n}$ with respect to different magnetic gradient directions $(\mathbf{g}_k)_{k=1\dots n}$ are available. Ideally, the expected signal at a voxel $\mathbf{x}$ for the direction k as explained in [10] should respect the following condition

$$S_k(\mathbf{x}) = S_0(\mathbf{x}) \exp\big(- b\mathbf{g}_k^t \mathbf{D}(\mathbf{x})\mathbf{g}_k \big)$$

with the tensor $\mathbf{D}$ being the unknown variable and b a value that depends on the acquisition settings. The estimation of the tensors in the volume domain Ω can be done through direct inference (6 acquisitions are at least available), which is equivalent to minimizing:

$$E_{data}(\mathbf{D}) = \int_{\Omega} \sum_{k=1}^{n} \Big(\log\big(S_k(\mathbf{x})/S_0(\mathbf{x})\big) + b\mathbf{g}_k^t \mathbf{D}(\mathbf{x})\mathbf{g}_k \Big)^2 d\mathbf{x}$$

This energy is based on the linearized diffusion tensor model which is reasonable for moderate values of SNR [11]. Such a direct estimation is quite sensitive to noise, on the other hand, it refers to a convex term, which is rather convenient when seeking its lowest potential. The most common approach to account for noise is through the use of an additional regularization term which constrains the estimation of $\mathbf{D}$ to be locally smooth. One of the most prominent uses of DTI is fiber extraction. Therefore it is natural to assume that locally these fibers do have similar orientations. In such a context, the tensor can be expressed as a linear combination of the tensors lying in its neighborhood since they are likely to represent the same population of fibers. Such a regularization constraint was introduced in the case of image restoration in [12]. This assumption still holds at the boundaries between different groups of fibers as long as the linear combination is thoroughly chosen to ensure that the contribution of tensors belonging to a different fiber population is negligible. It is also more accurate than the underlying assumption of total-variation based approaches where the tensor field is considered piecewise constant. This leads us to define the following regularization component:

$$E_{smooth}(\mathbf{D}) = \int_{\Omega} \left|\left| \mathbf{D}(\mathbf{x}) - \frac{1}{Z(\mathbf{x})} \int_{\mathbf{y} \in \mathcal{N}_{\mathbf{x}}} w(\mathbf{x},\mathbf{y}) \mathbf{D}(\mathbf{y}) d\mathbf{y} \right|\right|_F^2 d\mathbf{x}$$

where $w(\mathbf{x},\mathbf{y})$ reflects the similarity between tensors $\mathbf{D}(\mathbf{x})$ and $\mathbf{D}(\mathbf{y})$, $||A||_F$ being the Frobenius norm $||A||_F = \sqrt{tr(A^t A)}$ and $Z(\mathbf{x})$ is a normalization factor, i.e $Z(\mathbf{x}) = \int_{\mathbf{y} \in \mathcal{N}_{\mathbf{x}}} w(\mathbf{x},\mathbf{y}) d\mathbf{y}$. The most critical aspect of such an approximation model is the definition of weights, measuring the similarity between tensors within the local neighborhood. The use of Gaussian weights is a common weight's selection, i.e $\left[w(\mathbf{x},\mathbf{y}) = e^{\frac{-d^2(\mathbf{D}(\mathbf{x}),\mathbf{D}(\mathbf{y}))}{2\sigma^2}} \right]$, where $d(.;.)$ is a distance between tensors and σ a scale factor. In the context of direct estimation and regularization it is more appropriate to define similarities directly on the observation space rather than the estimation space. Such a choice will lead to a tractable estimation, while preserving the convexity of the cost function. Our distance definition as well as our minimization step are based on the representation of symmetric positive semi-definite matrices S^3_+ as a convex closed cone in the Hilbert space of symmetric matrices S^3, where the standard scalar product is defined by $\langle A, B \rangle_F = tr(A^t B)$ which induces the corresponding Frobenius norm.

2.1 Measuring Similarities from Diffusion Weighted Images

We aim at simultaneously estimating and smoothing the tensor field, therefore the weights $w(\mathbf{x},\mathbf{y})$ in E_{smooth} should be precalculated using the raw data. The most straightforward estimation of the distances can be done through the algebraic distance between the $\log(S_k/S_0)$ for two neighborhood voxels in any direction

$$d\big(\mathbf{D}(\mathbf{x}),\mathbf{D}(\mathbf{y})\big) = \frac{1}{b} \sqrt{\sum_{k=1}^{N} \Big(\log\big(S_k(\mathbf{x})/S_0(\mathbf{x})\big) - \log\big(S_k(\mathbf{y})/S_0(\mathbf{y})\big) \Big)^2}$$

One can easily show that such an expression does not reflect similarity between tensors according to the norm $||.||_F$. In fact, this leads to

$$d(\mathbf{D}(\mathbf{x}),\mathbf{D}(\mathbf{y}))=\sqrt{\sum_{k=1}^{N}\left(\mathbf{g}_k^t(\mathbf{D}(\mathbf{x})-\mathbf{D}(\mathbf{y}))\mathbf{g}_k\right)^2}=\sqrt{\sum_{k=1}^{N}<\mathbf{D}(\mathbf{x})-\mathbf{D}(\mathbf{y}),\mathbf{G}_k>_F^2}$$

where $\mathbf{G}_k = \mathbf{g}_k\mathbf{g}_k^t$ do not form necessarily an orthonormal basis. We use a Gram-Schmidt orthogonalization scheme to calculate an orthonormal basis $\widetilde{\mathbf{G}_k}$ such that $\widetilde{\mathbf{G}_k} = \sum_l \alpha_{kl}\mathbf{G}_l$ (each new vector of the new basis is a linear combination of the vectors of the initial basis). This procedure allows us to have an approximation of $||\mathbf{D}(\mathbf{x})-\mathbf{D}(\mathbf{y})||_F$ directly from the raw data S_k and S_0 as follows

$$\begin{aligned}||\mathbf{D}(\mathbf{x})-\mathbf{D}(\mathbf{y})||_F &= \sqrt{\sum_{k=1}^{N}<\mathbf{D}(\mathbf{x})-\mathbf{D}(\mathbf{y}),\widetilde{\mathbf{G}_k}>_F^2}\\ &= \frac{1}{b}\sqrt{\sum_{k=1}^{N}\Big(\sum_l \alpha_{kl}(\log\big(S_k(\mathbf{x})/S_0(\mathbf{x})\big)-\log\big(S_k(\mathbf{y})/S_0(\mathbf{y})\big))\Big)^2}\end{aligned}$$

2.2 Semi-definite Positive Gradient Descent

One now can seek the lowest potential of the cost function towards recovering the optimal solution on the tensor space. Unlike the Riemannian approaches where non convex functionals are minimized [6], the present framework consists of a convex energy with a unique minimum which can be reached using a projected gradient descent on the space of semi-definitive positive matrices. The projection from S^3 onto S^3_+ denoted by $\Pi_{S^3_+}$ is well defined and has an explicit expression. Indeed, projecting M amounts to replacing the negative eigenvalues in its spectral decomposition by 0 [6,13]. Note that we minimize over the set of semi-definite positive matrices because it is topologically closed, as opposed to the set of definite positive matrices. In the current setting, the problem is well posed and the projected gradient descent algorithm is convergent for a suitable choice of the time step dt. Using a weighting factor λ between the data attachment term and the regularization energy, the gradient descent can be expressed as the following equation

$$\begin{aligned}\mathbf{D}^{t+1}(\mathbf{x}) &= \Pi_{S^3_+}\Big(\mathbf{D}^t(\mathbf{x})-dt\frac{\partial E}{\partial\mathbf{D}(\mathbf{x})}(\mathbf{D}^t)\Big)\\ &= \Pi_{S^3_+}\Big(\mathbf{D}^t(\mathbf{x})-dt\lambda\frac{\partial E_{smooth}}{\partial\mathbf{D}(\mathbf{x})}(\mathbf{D}^t)-dt\frac{\partial E_{data}}{\partial\mathbf{D}(\mathbf{x})}(\mathbf{D}^t)\Big)\end{aligned}$$

where

$$\frac{\partial E_{smooth}}{\partial \mathbf{D}(\mathbf{x})}(\mathbf{D}) = 2\mathbf{D}(\mathbf{x}) - 2\int_{\mathbf{y}\in\mathcal{N}_{\mathbf{x}}} \frac{w(\mathbf{x},\mathbf{y})}{Z(\mathbf{x})}\mathbf{D}(\mathbf{y})d\mathbf{y}$$
$$- 2\int_{\mathbf{y}\in\mathcal{N}_{\mathbf{x}}} \frac{w(\mathbf{x},\mathbf{y})}{Z(\mathbf{y})}\Big(\mathbf{D}(\mathbf{y}) - \int_{\mathbf{z}\in\mathcal{N}_{\mathbf{y}}} \frac{w(\mathbf{z},\mathbf{y})}{Z(\mathbf{y})}\mathbf{D}(\mathbf{z})d\mathbf{z}\Big)d\mathbf{y}$$
$$\frac{\partial E_{data}}{\partial \mathbf{D}(\mathbf{x})}(\mathbf{D}) = 2b\sum_{k=1}^{N}\Big(\log\big(S_k(\mathbf{x})/S_0(\mathbf{x})\big) + b\mathbf{g}_k^t\mathbf{D}(\mathbf{x})\mathbf{g}_k\Big)\mathbf{G}_k$$

Let us define the norm $||.||_{TF}$ over the whole tensor field $\mathbf{D}$ as $||\mathbf{D}||_{TF} = \int_\Omega ||\mathbf{D}(\mathbf{x})||_F d\mathbf{x}$. Considering two tensor fields $\mathbf{D}_1$ and $\mathbf{D}_2$, we show in the following that the gradient of our energy functional is L-Lipschitz. The constant L will allow us to choose automatically a time step that insures the convergence of the algorithm.

$$\left\|\frac{\partial E_{data}}{\partial \mathbf{D}(\mathbf{x})}(\mathbf{D}_1) - \frac{\partial E_{data}}{\partial \mathbf{D}(\mathbf{x})}(\mathbf{D}_2)\right\|_F = 2b^2\sum_{k=1}^{N} < \mathbf{G}_k, \mathbf{D}_1(\mathbf{x}) - \mathbf{D}_2(\mathbf{x}) >_F$$
$$\leq 2b^2\sum_{k=1}^{N} ||\mathbf{G}_k||_F||\mathbf{D}_1(\mathbf{x}) - \mathbf{D}_2(\mathbf{x})||_F$$

Therefore $\|\nabla E_{data}(\mathbf{D}_1) - \nabla E_{data}(\mathbf{D}_2)\|_{TF} \leq 2b^2\sum_{k=1}^{N} ||\mathbf{G}_k||_F||\mathbf{D}_1 - \mathbf{D}_2||_{TF}$. Besides, we can easily show the following inequality

$$\|\nabla E_{smooth}(\mathbf{D}_1) - \nabla E_{smooth}(\mathbf{D}_2)\|_{TF} \leq 2(1 + 2|\mathcal{N}_{\mathbf{x}}| + |\mathcal{N}_{\mathbf{x}}|^2)||\mathbf{D}_1 - \mathbf{D}_2||_{TF}$$

where $|\mathcal{N}_{\mathbf{x}}|$ is the number of the considered neighbors. Thus the gradient of the objective function is L-Lipschitz with $L = 2b^2\sum_{k=1}^{N} ||\mathbf{G}_k||_F + 2\lambda(|\mathcal{N}_{\mathbf{x}}| + 1)^2$. Choosing $0 < dt < \frac{1}{b^2\sum_{k=1}^{N}||\mathbf{G}_k||_F + \lambda(|\mathcal{N}_{\mathbf{x}}|+1)^2}$ makes the projected gradient descent convergent [14].

We can give an interpretation of our regularization energy in terms of diffusion-weighted images smoothing. It can be easily verified that for each direction k

$$\int_\Omega < \mathbf{D}(\mathbf{x}) - \int_{\mathbf{y}\in\mathcal{N}_{\mathbf{x}}} \frac{w(\mathbf{x},\mathbf{y})}{Z(\mathbf{x})}\mathbf{D}(\mathbf{y})d\mathbf{y}, \mathbf{G}_k >_F^2 d\mathbf{x} =$$
$$\frac{1}{b^2}\int_\Omega \Big[\log\Big(\frac{S_k(\mathbf{x})}{S_0(\mathbf{x})}\Big) - \log\Big(\prod_{\mathbf{y}\in\mathcal{N}_{\mathbf{x}}}\big(\frac{S_k(\mathbf{y})}{S_0(\mathbf{y})}\big)^{\frac{w(\mathbf{x},\mathbf{y})}{Z(\mathbf{x})}}\Big)\Big]^2 d\mathbf{x}$$

Using Cauchy-Schwartz inequality we obtain :

$$\frac{1}{b^2}\int_\Omega \Big[\log\Big(\frac{S_k(\mathbf{x})}{S_0(\mathbf{x})}\Big) - \log\Big(\prod_{\mathbf{y}\in\mathcal{N}_{\mathbf{x}}}\big(\frac{S_k(\mathbf{y})}{S_0(\mathbf{y})}\big)^{\frac{w(\mathbf{x},\mathbf{y})}{Z(\mathbf{x})}}\Big)\Big]^2 d\mathbf{x} \leq E_{smooth}||\mathbf{G}_k||_F^2$$

We can see that minimizing E_{smooth} has a direct implication on the normalized diffusion weighted images $\frac{S_k}{S_0}$. Reconstructing the tensors using a linear combination of the tensors in its neighborhood leads to the reconstruction of the normalized signals using a weighted geometric mean of the neighboring signals where the weights are not calculated only with a single volume S_k but also with the volumes obtained from the other magnetic gradient directions.

3 Experimental Validation

In order to validate the performance of the method we (i) have generated artificial tensors volumes corrupted with synthetic noise, (ii) used manual segmentation on T1 muscle images and tried to improve the separability of classes in the DTI space after regularization.

3.1 Artificially Corrupted Tensors

Let us consider two volumes, one that consists of two classes with orthogonal axes on a $20\times20\times20$ lattice and a helix in which the internal voxels are anisotropic and the external ones are spheric [Fig.1-b]. For the first volume, the tensor fields for each region are $\mathbf{D_1} = 0.001 \times [1\ 0.5\ 0.5\ 0\ 0\ 0]$ and $\mathbf{D_2} = 0.001 \times [0.2\ 0.4\ 0.2\ 0\ 0\ 0]$ where $\mathbf{D}$ is presented in the form of $\mathbf{D} = [D_{xx}\ D_{yy}\ D_{zz}\ D_{xy}\ D_{xz}\ D_{yz}]$. The helix dataset can be found at [15]. We considered for both datasets a field strength $b = 700 s.mm^{-2}$, a constant value for $S_0 = 60$ for all volume voxels and twelve directions for diffusion gradient, which are used to generate the DTI corresponding to such tensor estimations. The chosen directions are the following :

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0.41 & 0.41 & 0.41 & 0.41 & 0.41 & 0.41 & 0.41 & 0.41 \\ 0.41 & -0.41 & -0.41 & 0.41 & 0.41 & 1 & 1 & 0.41 & -0.41 & -1 & -1 & -0.41 \\ -0.41 & -0.41 & 0.41 & 0.41 & 1 & 0.41 & -0.41 & -1 & -1 & -0.41 & 0.41 & 1 \end{pmatrix}$$

The images were corrupted with a white zero-mean Gaussian noise forming a data set where ground-truth on the tensor are available. An estimation of the tensor field relative to the noisy images provides the noisy tensors data.

Then, to perform comparisons we considered the regularization algorithm on noisy tensors presented in [6]. The following parameters were used for our method: $\lambda = 50$, $\mathcal{N}_{\mathbf{x}} = 3 \times 3 \times 3$, $dt = 10^{-7}$ with 50 iterations. To evaluate the performance of these methods, we considered the average sum of squared differences (**SSD**) between the regularized tensors and ground truth ones. In [Table 1], we can see that our estimation and regularization approach achieves better results and produces a tensor close to the ground truth. Our method outperforms the one of [6] when the level of noise is relatively important. In fact, our method considers a more robust resemblance degree between voxels. Such a criterion insures a better selection of neighboring tensors involved in the estimation of a given tensor. On the other hand, the anisotropic diffusion based regularization relies on gradient information which is not robust in case of high noise. In order to assess qualitatively our algorithm, we reported in [Fig. 1] the resulting tensors using our regularization method and the constrained anisotropic one. We can observe that our method achieves a better direction preservation, even in the presence of a strong noise.

Table 1. Average Sum of Square Differences (SSD)$\times 10^4$. Comparisons between our method and the one in [6].

	Helix dataset			Homogeneous regions		
σ_n	0.5	1.2	3	1.5	4	9
Noisy Tensor	1.08	6.24	39.54	9.82	71.25	393.38
Method in [6]	0.33	1.60	10.57	3.32	22.47	120.70
Our Method	0.41	1.38	3.78	0.44	4.23	18.30

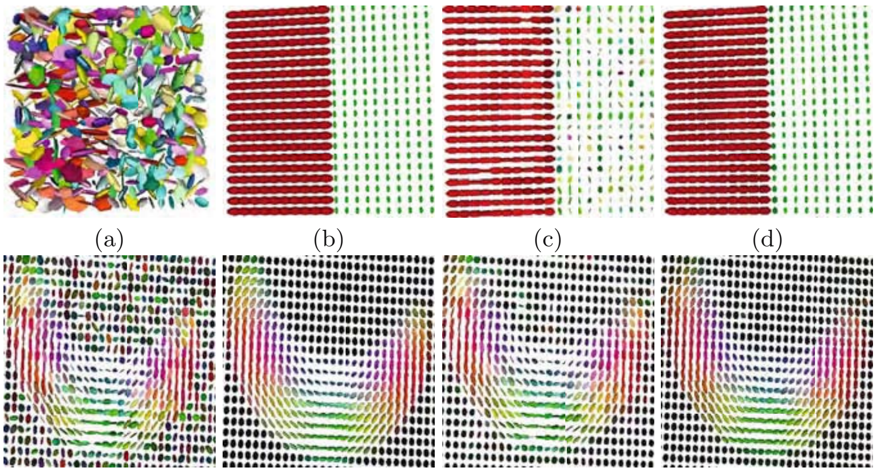


Fig. 1. Tensors on a volume slice: (a) Noisy tensors (b) Ground-truth (c) Result obtained with [6] (d) Result obtained with our method

3.2 DTI Towards Understanding the Human Skeletal Muscle

In order to perform validation using real data, the following experiment was considered. DTI acquisitions of human skeletal muscle (calf) using 12 directions were carried out on a 1.5 T MRI scanner with the following parameters : repetition time (TR)= $3600ms$, echo time(TE) = $70ms$, slice thickness = $7mm$ and b value of $700\, s.mm^{-2}$. In order to improve the signal-to-noise ratio, the acquisition was repeated thirteen times (one can use the average of the measurements) while a high resolution T1-weighted volume was also obtained and manually segmented [Fig. 2]. The muscles that were considered in our study were the soleus (SOL), lateral gastrocnemius (LG), medial gastrocnemius (MG), posterior tibialis (PT), anterior tibialis (AT), extensor digitorum longus (EDL), and the peroneus longus (PL). Several previous studies investigated the use of diffusion tensor imaging to study the architecture of skeletal muscle and to separate these muscle groups according to different properties (fiber orientation, mean diffusivity, fractional anisotropy . . .)[16,17].

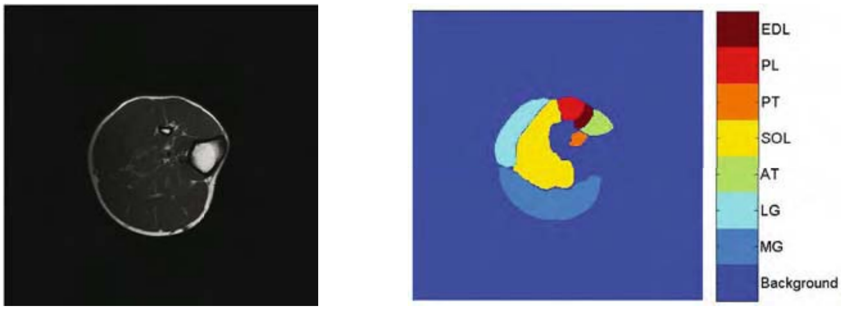


Fig. 2. A slice of the T1-weighted volume, different muscle groups segmented manually

Table 2. Correct classification rates for the different methods and for each muscle group. The first and third row show the average correct classification rates for the set of 13 volumes.

	Overall	MG	LG	SOL
DE	78.1%	86.16%	51.1 %	84.43%
ADE	84.46%	90.47%	65.72%	88.43%
DER	86.45%	91.82%	69.76%	89.97%

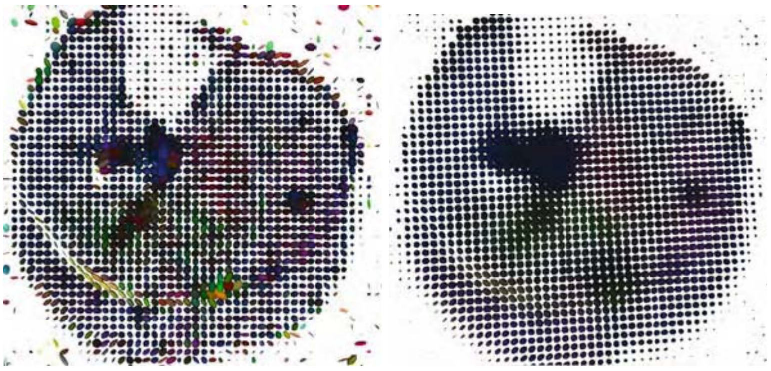


Fig. 3. Estimated tensors without regularization, tensors obtained with our method

In order to proceed with an evaluation of the proposed method, the following scenario was considered: Using the manual segmentation, and the observed measurements of a given acquisition (12 directions), we have constructed seven weak linear classifiers (in our case a multi-class linear SVM[18]) separating each class of muscle versus all others. Then, the success rate (percentage of voxels being attributed to the right class) from the classifier with respect to the ground truth was determined. We remark that linear separation is hardly achieved for PT, PL, EDL and AT while it yields quite satisfactory results for the MG, LG and to a lesser extent SOL which form the major part of the muscle.We have performed this test thirteen times for: (i) direct estimation (**DE**), (ii) direct estimation and regularization (**DER**), as well as using direct estimation of the average measurements of the thirteen acquisitions (**ADE**) . One would expect that since muscles consist of myo-fibers of the same nature, the classification should be improved if the estimation of the tensors is properly done, i.e. with appropriate regularization. However, it is important to note that the aim of this paper is not automatic classification of voxels in different muscle regions using DTI (in such a case more advanced classifiers can be used).

In [Table 2], we present quantitative validation of the present framework for the linearly separable muscles. One can see that our method leads to an improvement in the correct classification rates with respect to a plain direct estimation. We also obtain better results when compared to the averaging+estimation method. We also show the result of our regularization on a slice of the volume in [Fig. 3].

4 Discussion

In this paper a novel approach to direct estimation and regularization of diffusion tensor images was proposed. The main strength of our approach is the novel regularization term that assumes linear approximation of neighborhood tensors as well as the convex nature of the proposed cost function which can be easily optimized. Our method was compared and outperformed the anisotropic constrained regularization using generated data with known noise model, and significantly improved human skeletal muscle segmentation/classification through DTI using real data.

The selection of the bandwidth σ is a critical parameter of the process. Data-driven variable bandwidth models is a natural extension of the method. One would expect that the optimal bandwidth depends on the form of the observed anatomical structure which varies spatially. Another possible extension of this work is to replace the Frobenius norm in the energy functional by the Riemannian metric [5] or the Log-Euclidean metric [9]. However this will be done at the expense of the convexity of the function and the computational time.

The use of DTI towards understanding the human skeletal muscle as well as providing means of diagnosis for muscular diseases is a more long-term objective of our research. The ability to understand the remodeling of myofibers due to muscular diseases using non-invasive means is a great perspective.

Acknowledgements. We are thankful to D. Tschumperlé for providing us with the implementation of the regularization algorithm [6] for comparison and also to J.-F. Deux, G. Bassez and A. Rahmouni from Henri Mondor Hospital for providing the DTI data.

References

1. Bihan, D.L., Mangin, J.F., Poupon, C., Clark, C.A., Pappata, S., Molko, N., Chabriat, H.: Diffusion tensor imaging: concepts and applications. Journal of Magnetic Resonance Imaging 13, 534–546 (2001)

2. Coulon, O., Alexander, D.C., Arridge, S.: Diffusion tensor magnetic resonance image regularization. Medical Image Analysis 8, 47–67 (2004)
3. Basu, S., Fletcher, P.T., Whitaker, R.T.: Rician noise removal in diffusion tensor mri. In: Larsen, R., Nielsen, M., Sporring, J. (eds.) MICCAI 2006. LNCS, vol. 4190, pp. 117–125. Springer, Heidelberg (2006)
4. Martín-Fernández, M., Westin, C.F., Alberola-López, C.: 3D Bayesian regularization of diffusion tensor MRI using multivariate Gaussian Markov random fields. In: Barillot, C., Haynor, D.R., Hellier, P. (eds.) MICCAI 2004. LNCS, vol. 3216, pp. 351–359. Springer, Heidelberg (2004)
5. Castano-Moraga, C.A., Lenglet, C., Deriche, R., Ruiz-Alzola, J.: A Riemannian approach to anisotropic filtering of tensor fields. Signal Processing [Special Issue on Tensor Signal Processing] 87, 263–276 (2007)
6. Deriche, R., Tschumperle, D., Lenglet, C., Rousson, M.: Variational approaches to the estimation, regularization and segmentation of diffusion tensor images. In: Faugeras, P.C. (ed.) Mathematical Models in Computer Vision: The Handbook, 2005th edn., Springer, Heidelberg (2005)
7. Wang, Z., Vemuri, B.C., Chen, Y., Mareci, T.H.: A constrained variational principle for direct estimation and smoothing of the diffusion tensor field from complex DWI. IEEE Transactions on Medical Imaging 23, 930–939 (2004)
8. Weickert, J., Feddern, C., Welk, M., Burgeth, B., Brox, T.: PDEs for tensor image processing. In: Weickert, J., Hagen, H. (eds.) Visualization and Processing of Tensor Fields, pp. 399–414. Springer, Heidelberg (2006)
9. Fillard, P., Arsigny, V., Pennec, X., Ayache, N.: Clinical DT-MRI estimation, smoothing and fiber tracking with log-Euclidean metrics. In: ISBI 2006. Proceedings of the IEEE International Symposium on Biomedical Imaging, Crystal Gateway Marriott, Arlington, Virginia, USA, pp. 786–789. IEEE Computer Society Press, Los Alamitos (2006)
10. Stejskal, E., Tanner, J.: Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient. Journal of Chemical Physics 42, 288–292 (1965)
11. Salvador, R., Pea, A., Menon, D.K., Carpenter, T.A., Pickard, J.D., Bullmore, E.T.: Formal characterization and extension of the linearized diffusion tensor model. Human Brain Mapping 24, 144–155 (2005)
12. Azzabou, N., Paragios, N., Guichard, F., Cao, F.: Variable bandwidth image denoising using image-based noise models. In: CVPR 2007 (2007)
13. Hiriart-Urruty, J.B., Lemaréchal, C.: Fundamentals of Convex Analysis. Springer, Heidelberg (2001)
14. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont, MA (1999)
15. `www.sci.utah.edu/~gk/DTI-data/`
16. Galban, C.J., Maderwald, S., Uffmann, K., de Greiff, A., Ladd, M.E.: Diffusive sensitivity to muscle architecture: a magnetic resonance diffusion tensor imaging study of the human calf. European Journal of Applied Physiology 93, 253–262 (2004)
17. Galban, C.J., Maderwald, S., Uffmann, K., Ladd, M.E.: A diffusion tensor imaging analysis of gender differences in water diffusivity within human skeletal muscle. NMR in Biomedicine (2005)
18. Joachims, T.: Making large-scale support vector machine learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge (1998)

Robust Self-calibration from Single Image Using RANSAC

Qi Wu[1], Te-Chin Shao[2], and Tsuhan Chen[1]

[1] Carnegie Mellon University
[2] Industrial Technology Research Institute

Abstract. In this paper, a novel approach for the self-calibration of single image is proposed. Unlike most existing methods, we can obtain the intrinsic and extrinsic parameters based on the information of restricted image points from single image. First, we show how the vanishing point, vanishing line and foot-to-head plane homology can be used to obtain the calibration parameters and then we show our approach how to efficiently adopt RANSAC to estimate them. In addition, noise reduction is proposed to handle the measurement uncertainties of input points. Results in synthetic and real scenes are presented to evaluate the performance of the proposed method.

1 Introduction

As is well know, camera calibration, which involves finding the relation of image points and their corresponding points in 3D, is an important issue in computer vision. In generally, the extensive knowledge of the scene geometry must be available in solving this problem. Recent years, camera self-calibration has been proposed that can only rely on the image itself instead of a particular calibration object. In these techniques, not only camera intrinsic constraints or camera motion constraints but also scene constraints are used to solve the problem of calibration. Some practical methods for computing the calibration for multiple images have been given by Maybank[1] for using the Kruppa equation; Hartley[2] calibrated a camera from pure rotation based on recovered homographies and Triggs[3] use planar scenes constraints. Comparing to these methods, we can obtain both intrinsic and extrinsic parameters just from the information of point in single image, which is more useful in the reality.

Moveover, under the assumption that image is obtained by perspective projection, vanishing line and vanishing point of parallel planes have proven to be useful features. Liebowitz and Zisserman[4] used vanishing points of orthogonal directions and rectified planes as the scene constraints. Lv et al.[5] used the horizon line and the vanishing point by tracking a moving human in the scene to get an approximate solution with some assumptions. Krahnstoever[6] aimed to handle the noise models in the data for self-calibration by using homology and Bayesian solution.

Unfortunately, the estimation of the vanishing line and especially far off vanishing points is made difficult in practice by errors and noise in the image point selection. Henceforth, in the paper, we proposed a robust estimation by adopting RANSAC, which can eliminate the outliers and cull the input points to improve the accurateness of estimation. In addition, using foot-to-head homology to converge the symmetric transfer errors, noise reduction is employed to correct the errors of input points in order to make our calibration algorithm more stable and reliable. Finally, the experiment in synthetic and real scenes are presented to prove our method is effective and robust.

2 Camera Model

Under the assumptions of zero skew, unit aspect ration and known principal point, which is $(U_0\ V_0)$ in the cartesian coordinates, the matrix of intrinsic parameters of our projective camera is denoted by:

$$K = \begin{bmatrix} f & 0 & U_0 \\ 0 & f & V_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Since the World Coordinate System(WCS),defined by three orthogonal vectors $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$, is vertically below the the Camera Coordinate System (CCS) along the $\boldsymbol{Y}$-axis, the 3×4 projection matrix P is then denoted by $P = K[R|t]$ with the rotation matrix:

$$R = R^z_\alpha R^x_\gamma = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{2}$$

and translation matrix $t = [0, -h_c, 0]$.

Given an image, we assume that the scene is well modelled by a dominant ground plane that defines the horizon and on which the vertical segments of objects have constant height. In that case, the objects in the scene have parallel head and foot planes. Moreover, any two points on these two planes correspond if the line passing them is vertical to the ground plane. Therefore, given the corresponding points manually selected from the image as the input data (fig 1),our goal is to compute the intrinsic parameter(the focal length f) and extrinsic parameters(the rotation angle α,γ and height of the camera h_c) for calibrating the camera.

3 Self-calibration from Single Image

In the input image, the lines passing through corresponding heads and feet intersect at the 3D point $V^\infty = [0, 1, 0, 0]^T$ at infinity. The image location of this point along $\boldsymbol{Y}$-axis is call vanishing point(v^∞ in the fig.2). Given the projective matrix $P = [p_1, p_2, p_3, p_4]$,where p_i is the i-th column of P. we can get that $v^\infty = PV^\infty = p_2$. Therefore, the vertical vanishing point is denoted by

$$v^\infty = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -f\cos(\alpha)\sin(\gamma) + U_0\sin(\alpha) \\ f\cos(\alpha)\cos(\gamma) + V_0\sin(\alpha) \\ \sin(\alpha) \end{bmatrix} \tag{3}$$

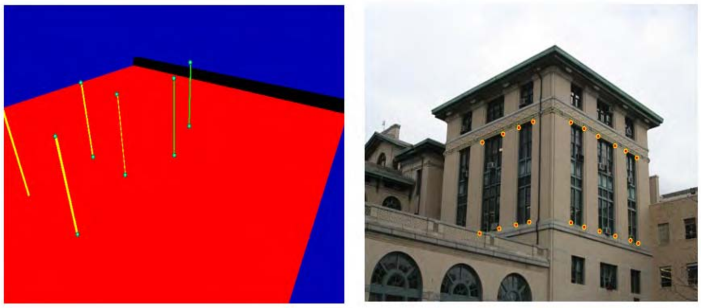


Fig. 1. The input image marked head points and foot points. Left figure is the synthetic data and right figure is the real data.

which can be written in cartesian coordinates as

$$v^{\infty} = \begin{bmatrix} v_1/v_3 \\ v_2/v_3 \end{bmatrix} = \begin{bmatrix} f\cot(\alpha)\sin(-\gamma) + U_0 \\ f\cot(\alpha)\cos(-\gamma) + V_0 \end{bmatrix} \tag{4}$$

where α and γ are rotation angles. Because $\alpha \neq \pi/2$. We can see that the vanishing point v_y^{∞} is at a distance of $fcot(\alpha)$ from the principal point(U_0, V_0) and at an angle of $-\gamma$ from the $\boldsymbol{Y}$-axis. For generally situation in our case, if an upward looking camera has $\boldsymbol{X}$-axis rotation angle in the range $\alpha \in (0, \pi/2)$,the $\cot(\alpha)$ will be in range of $(0, \infty)$.

In addition, the heads and feet are from two parallel planes. The image of the line at infinity of these two planes is defined as the vanishing line(l^{∞} in fig.2) parallel to the ground plane. The image point of $V_1^{\infty} = [1, 0, 0, 0]^T$ and $V_3^{\infty} = [0, 0, 1, 0]^T$ are both on the vanishing line and given p_1 and p_3 respectively. Hence,the vanishing line is written as $l^{\infty} = p_1 \times p_3$,which can be denoted by:

$$l^{\infty} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \begin{bmatrix} f\cos(\alpha)\sin(\gamma) \\ -f\cos(\alpha)\cos(\gamma) \\ -f^2\sin(\alpha) + f\cos(\alpha)(V_0\cos(\gamma) - U_0\sin(\gamma)) \end{bmatrix} \tag{5}$$

Now from the above equations, we can easily find the rotation angles α,γ and the focal length f from the equations:

$$\gamma = \arctan(\frac{l_1}{l_2}) \tag{6}$$

$$\alpha = \arctan(\sqrt{\frac{V_0\cos^2(\gamma) - U_0\sin\gamma\cos\gamma - \frac{l_3}{l_1}\sin(\gamma)\cos(\gamma)}{\frac{v_2}{v_3} - V_0}}) \tag{7}$$

$$f = \frac{\frac{v_2}{v_3} - V_0}{\cos(\gamma)}\tan(\alpha) \tag{8}$$

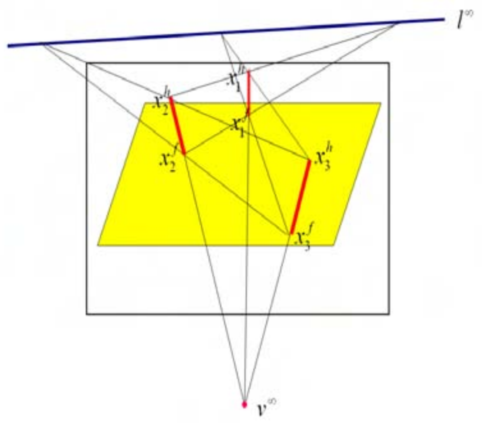


Fig. 2. The Geometry in our assumption

To compute h_c, we use the estimated foot-to-head homology H. It is a particular homography reflecting the mapping from foot plane to head plane. The 3×3 matrix H defining the foot-to-head transformation can also be represented as:

$$H = I - \frac{h}{h_c} \frac{v^\infty (l^\infty)^T}{(v^\infty)^T l^\infty} \tag{9}$$

where h is the known constant height of poles and h_c is the position of camera which can be derived from the above equation.

The discussion above shows that the information in the vanishing point, vanishing line and homology are equivalent to the information contained in the classical projection model. Knowledge of one can be used to recover the other.

4 Estimation of Vanishing Point, Vanishing Line and Homology

Use of the method in section 3 requires finding the vanishing point, vanishing line and foot-to-head homology. In our approach, these estimations are achieved by running RANSAC algorithm. RANSAC is a random sampling procedure which is known to eliminate the outliers from a large number of samples. The inliers culled from RANSAC can be used to estimate vanishing point,vanishing line and homology more robust and precise.

The RANSAC algorithm adopted for estimating vanishing line is outlined as Algorithm.1

In estimation of vanishing point, we first find all N lines passing corresponding head and foot and compute the intersection of any two lines as the data set S_p. After RANSAC procedure similar to the estimation of vanishing line. We can get the mean of largest set of inliers S_p^{max} as our estimation of vanishing point.

Algorithm 1. Determining the vanishing line

```
Input: The set of $N$ corresponding heads and feet
Output: The vector of estimation of vanishing line.
1: Find all $\binom{N}{2}$ lines passing two heads and feet respectively. Compute the
   intersections of pair of head and feet lines as our data set $S_l$(see fig.2)
2: Randomly select 2 intersection vectors to fit a line $l^e$
3: Find the inliers set $S_l'$ under the distance threshold criterion: $d < T_l$, $d$ is the
   distances between line and other intersections.
4: Repeat 2 and 3 till all lines have been tested. Find the largest set of inliers $S_l^{max}$.
5: Fit $S_l^{max}$ using Least Median Square to get the best estimation of vanishing line.
```

Algorithm 2. Determining the foot-to-head homology

```
Input: The set of $N$ corresponding heads and feet
Output: The best estimation of foot-to-head homology matrix.
1: Define the set of $N$ corresponding heads and feet as the data set $S_H$
2: Randomly select 4 pairs of corresponding heads and feet, use DLT algorithm to
   compute the homography and convert to homology $H$.
3: Given origin feet and heads, estimate the corresponding new heads and feet using
   homology $H$ and reverse homology $H^{-1}$
4: Find the inliers set $S_H'$ under the symmetric transfer error threshold criterion:
   $d(x_i^f, x_i^h) = ||x_i^h - Hx_i^f|| + ||x_i^f - H^{-1}x_i^h|| < T_H$
5: Repeat 2,3 and 4 till all homologies estimated from 4 pairs of points have been
   tested. Find the largest set of inliers $S_H^{max}$.
6: Estimate the best homology from the inliers $S_H^{max}$
```

The RANSAC algorithm adopted for estimating foot-to-head homology is outlined as Algorithm.2

5 Noise Analysis and Reduction

Despite the robustness of RANSAC, the conditions of manual heads and feet selection are noisy. With the repetitious computation using these noisy points, the errors on vanishing points and vanishing line are greatly amplified and give the poor results in calibration. Therefore, we propose using a stable noise reduction to correct the coordinates of heads and feet such that they can be located more accurately.

Shown as the fig.3, the pair of points(x_1^f, x_1^h) is the precise selection and there are 6 error situations existed in the human heads and feet selection $(x_i^f, x_i^h, i \in [2, 7])$. In addition, foot-to-head homology, which represents the mapping between foot plane and head plane, can be used to correct the error as following: (a) use foot-to-head homology to estimate all corresponding heads and feet from the origins: $x_e^h = Hx^f, x_e^f = H^{-1}x^h$; (b) compute the new points with mean of

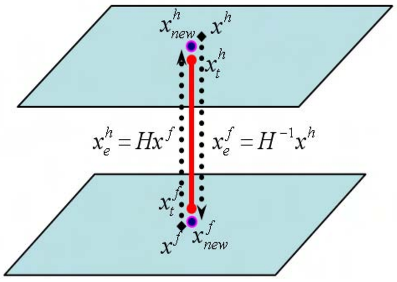


Fig. 3. The Geometry in our assumption

the estimated and the origins: $x^h_{new} = mean(x^h_e, x^h), x^f_{new} = mean(x^f_e, x^f)$; ($c$) compute the symmetric transfer error $d{=}||x^h - x^h_{new}|| + ||x^f - x^f_{new}||$ between new points and origin ones; (d) quit the iteration if the transfer error d converge and less than a Threshold T, $d < T$. Otherwise, update the origin with the new points and repeat previous steps.

The x^h_{new} and x^f_{new} of last iteration are our final corrected points which are more accurate than the origins.

6 Experiments

The proposed method has been tested on both synthetic data and real data. First,we create a synthetic scene as fig.1(left). The resolution of these input images is 640×480. The camera is located 2 meters above the ground plane with a tilt angle of $\pi/6$. The focal length of camera is $f = 480$ in the situation of unit aspect ratio, zero skew and principal point at [320,240]. we randomly insert some vertical poles, of 1.7 meters in height. Gaussian noise with zero mean and different standard deviations are added to heads and feet separately. we repeated the experiments 100 times for each value of the standard deviation. Table 1 shows the comparison of estimation and ground truth for different values of the noise standard deviation. Table 2 demonstrates the comparison of origin estimation and estimation with noise reduction. From these tables we can note, based on the increase of noise, the mean of results have more bias with larger standard deviation. But it can be observed that the errors of mean and standard deviation after noise reduction are much less than the origin ones which means the noise reduction procedure is effective.

In the real data as shown in figure 4, we show a example of four images taken from a real world with different focal lengthes(28mm lens and 50mm lens). The image resolution is 3456 × 2304. Table 3 shows the comparison of focal length under 28mm lens and 50mm for different noises in two real scenes respectively. We use Zhengyou Zhang's method[7] for the comparison to verifie the accuracy of the proposed method. However,from the table, we can note that if the scenes in the images are not perspective enough(such as 50mm lens compared to 28mm lens or scene 2 compared to scene 1), the vanishing point and vanishing line

Table 1. Comparison of estimation and ground trues for different values of the noise standard deviation in synthetic Scene

	f	α	γ	h_c
Truth	480	0^o	30^o	2m
Std.Gaus.Error	***Estimation***			
0	480.04	0.00^o	30.01^o	1.95m
1	478.67	0.45^o	29.79^o	1.91m
2	483.13	-0.65^o	29.12^o	1.89m
3	475.39	-0.75^o	28.13^o	1.82m

Table 2. Comparison of origin estimation and estimation with noise reduction

	Origin		Noise Reduction	
Std.Gaus.Error	mean	std	mean	std
0	478.95	±0.994	480.04	±0.112
1	483.88	±1.787	478.67	±1.061
2	487.95	±5.631	483.13	±2.433
3	496.32	±9.118	475.39	±5.698

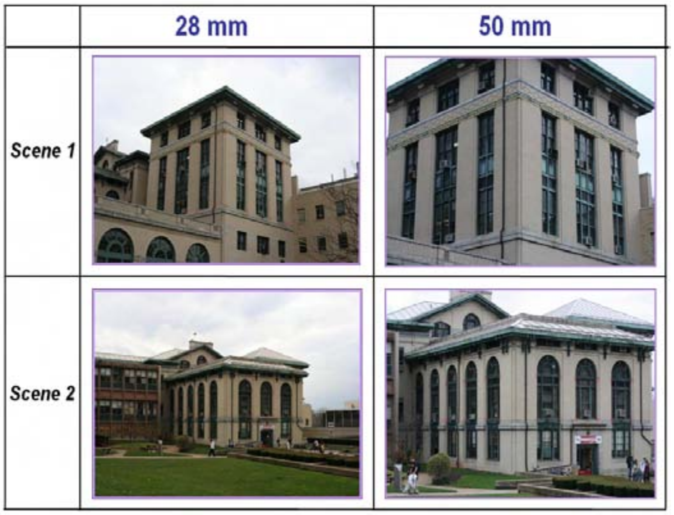


Fig. 4. Example of four images taken from a real world with different focal lengthes(28mm lens and 50mm lens)

will be formed so far that a little noise at locations of heads and feet can cause large errors in the estimation of focal length. Thus, in our algorithm, the images should contain perspective scene as input for obtaining a better result.

Table 3. Comparison of focal length under 28mm lens and 50mm for different noises in real scene 1

	28 mm		*50 mm*	
Zhengyou's	4378.79		7445.16	
	mean	std	mean	std
Scene 1	4456.32	±15.941	7576.55	±22.531
Scene 2	4498.43	±21.913	7647.57	±25.113

7 Conclusion

In this paper, given corresponding heads and feet, a novel and readily method for calibrating a camera from single image is developed. To acquire more precise camera parameters, a robust method to estimate the vanishing point, vanishing line and foot-to-head homology is necessary. Besides, noise reduction is proposed to reduce the noise of human points selection. It can be seen from the experiments that our method is robust and effective when the scene of input image is perspective.

References

1. Maybank, S., Faugeras, O.: A theory of self-calibration of a moving camera. International Journal of Computer Vision 8(2), 123–151 (1992)
2. Hartley, R.I.: Self-calibration from multiple views with a rotating camera. In: Proc. of European Conf. on Computer Vision, vol. 1, pp. 471–478 (1994)
3. Triggs, B.: Autocalibration from planar scenes. In: Proc. of European Conf. on Computer Vision, vol. 1, pp. 89–105 (1998)
4. Liebowitz, D., Zisserman, A.: Combining scene and autocalibration constraints. In: Proc. of International Conf. on Computer Vision, vol. 2, p. 293 (1999)
5. Lv, F., Tao, Z., Nevatia, R.: Self-calibration of a camera from video of a walking human. In: Proc. of International Conf. on Pattern Recognition, vol. 1, pp. 562–567 (2002)
6. Krahnstoever, N., Mendonca, P.: Bayesian autocalibration for surveillance. In: Proc. of International Conf. on Computer Vision, vol. 2, pp. 1858–1865 (2005)
7. Zhang, Z., Faugeras, O., Deriche, R.: An effective technique for calibrating a binocular stereo through projective reconstruction using both a calibration object and the environment. Journal of Computer Vision Research 1(1), 58–68 (1997)

Contour Matching in Omnidirectional Images

Yongho Hwang, Jaeman Lee, and Hyunki Hong

Dept. of Image Eng., Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University
{hwangyh,leejaeman}@wm.cau.ac.kr, honghk@cau.ac.kr

Abstract. This paper presents a novel method for contour matching in the architectural scenes captured by the omnidirectional camera. Since most line segments of man-made objects are projected to lines and contours, contour matching problem is important for 3D analysis in an omnidirectional indoor scene. First, we compute an initial estimate of the camera parameters from corner points and correlation-based matching. Then, the obtained edges by Canny detector are linked and divided into separate 3D line segments. By using a minimum angular error of endpoints of each contour, we establish the corresponding contours, and the initial parameters are refined iteratively from the correspondences. The simulation results demonstrate that the algorithm precisely estimates the extrinsic parameters of the camera by contour matching.

Keywords: omnidirectional camera, contour matching, corresponding contour, camera calibration, epipolar constraint.

1 Introduction

Camera pose estimation and 3D reconstruction from image sequence have long been one of the central topics in computer vision. Since the multi-view image analysis is based on establishing correspondence of images, matching features—points, lines, contours—is an important process. When the motion between two images is large, however, the matching problem becomes very difficult.

The omnidirectional camera system is given increasing interest by researchers working in computer vision [1-14], because it can capture large part of a surrounding scene with an angle of over 180°. Therefore, using wide angle of view often makes it possible to establish many spacious point correspondences, which lead to more complete 3D reconstruction from few images. In addition, it is widely used to capture the scene and illumination from all directions from far less number of images.

This paper aims at estimating the extrinsic parameters of the omnidirectional camera by contour matching in the architectural scenes. Contours are more general primitives than points or line segments, and they contain more information about the image [15]. However, most of previous calibration researches of omnidirectional images are based on not the contour correspondence but the feature points such as corners.

Previous methods for contour matching have been mainly proposed in perspective images. They use the epipolar geometry, the continuity of contours over successive

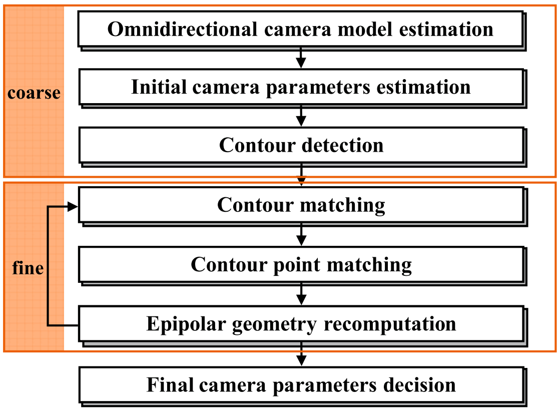

coarse
Omnidirectional camera model estimation
Initial camera parameters estimation
Contour detection
fine
Contour matching
Contour point matching
Epipolar geometry recomputation
Final camera parameters decision

Fig. 1. Block diagram for proposed algorithm

frames, or the mathematical relation between the observed edge segments and the projections of the reconstructed lines [15-17]. In addition, contour matching researches for camera calibration in the omnidirectional images by the paracatadioptric camera were presented. They use the geometric properties induced by the optical system or the conic curve fitting [13, 14]. However, there were few methods to solve this problem in omnidirectional images by fish-eye lens.

Straight line features are prominent in most man-made environments such as the artificial building and indoor scenes, and they provide a great deal of information about the scene structure. Furthermore, because edge features have more image support than point features, they can be localized more accurately. Since the line segments of man-made objects are projected to contours in omnidirectional images, this paper aims at contour matching for 3D image sequence analysis.

The initial estimate of an essential matrix is obtained from corner points and correlation-based matching. Contours are extracted by using edge detection and linking algorithm that segments the detected edges to 3D lines. Then, we establish the corresponding contours by using a minimum angular error of endpoints of each contour, and the initial parameters are refined from the matched contour set. The simulation results showed that the proposed method can estimate accurate omnidirectional camera parameters and achieve more precise contour matching.

The remainder of this paper is structured as follows: Sec. 2 explains contour matching in both the initial estimation and the refine process, and the simulation results are presented in Sec. 3. Finally, the conclusion is described in Sec. 4.

2 Proposed Method

To estimate the relative camera rotation and position, we propose a novel coarse-to-fine algorithm by using the corresponding contours between omnidirectional stereo images. For the first phase, an initial camera motion parameter between views is computed using epipolar geometry. Secondly, after contour matching, we obtain more accurate final parameters minimizing the proposed cost function. Fig. 1 shows a block diagram for the proposed algorithm.

2.1 First Step for Initial Estimate

2.1.1 Omnidirectional Camera Model and Projected Contour

General perspective projection model cannot represent the projection relation of the omnidirectional camera, and the radial symmetric projection model depends on the imaging system. By using calibration pattern images, we derive a model function with 9 parameters and utilize it to estimate the projection model of the omnidirectional camera [12]. The projection model consisting of the incident vector **p** of the pixel at the distance r in the omnidirectional image with a radius $r_{\max}$, and a height value z is shown in Fig. 2.

Fig. 3 (a) shows a line segment in the 3D space mapped to an arc on a hemi-sphere, S. Arc on S is orthogonally projected into a contour, c, on the image plane. The interpretation plane, Π, consisting of the line segment and the camera center is defined, and its normal vector $\mathbf{m} = (m_x, m_y, m_z)$ is computed through cross product of two directional vectors of endpoints: $\mathbf{p}_1 \times \mathbf{p}_2$. The 3D vector of the line segment on S,

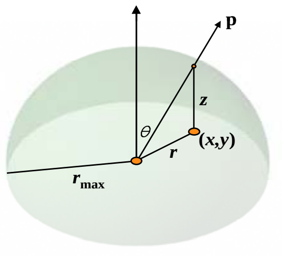
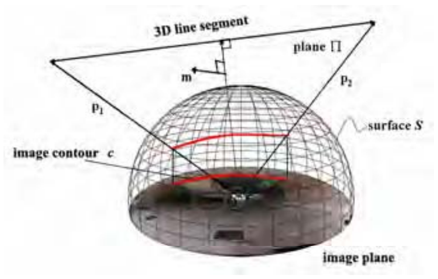
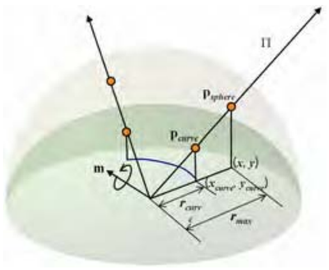


Fig. 2. Projection model of omnidirectional camera



(a)



(b)

Fig. 3. (a) 3D line segment and contour on the image plane (b) relation of intersection curve between sphere and projection model which is projected to a point on the contour in the omnidirectional image, is calculated by rotating $\mathbf{p}_1$ to $\mathbf{p}_2$ using the normal vector, **m** as a basis.

Since the projection model of the omnidirectional camera with the fisheye lens is generally not modeled using a hemispherical shape, it is necessary to compute the vector, $\mathbf{p}_{curve}$, at which point the plane, Π, and the estimated camera model meet. At first, we obtain the vector, $\mathbf{p}_{sphere}$, which intersects with a sphere with a radius the distance from the camera center when θ is the maximum view angle, r_{max}. While rotating with the normal vector, **m**, on the plane, Π, as the basis, the contour trace is calculated by projecting the vector, $\mathbf{p}_{sphere}$, to the estimated camera model. The r_{curve} is computed from the estimated camera model using the incident angle θ. The 2D coordinates of the contour projected on the image plane are then determined as follows [18]:

$$y_{curve} = \pm\sqrt{r_{curve}^2 - x_{curve}^2}\,, \quad x_{curve} = x \times (r_{curve} / r_{sphere}) \ . \tag{1}$$

2.1.2 Initial Camera Parameters Estimation

Between two views there is an epipolar constraint and it is known that at least 8-point matches are needed to estimate the eipopolar geometry in terms of the essential matrix [19]. In this paper, we use Harris corner detector to extract feature points in the view frames to be matched [20]. Then, matching candidates between two images are established using a correlation-based technique. The correlation-based point matching is utilized as the initial estimation of the essential matrix to recover epipolar geometry.

Since the essential matrix estimation is sensitive to the false matched errors, 8-point RANSAC, which is one of the representative robust algorithms, is performed to compute the essential matrix from the selected inliers. The camera's relative rotation matrix **R** and unit translation vector **t** can be determined by using Singular Value Decomposition (SVD). We can obtain 4 possible combinations of rotation and translation from one essential matrix. A correct combination can be determined by recovering the depths to each tracked point according to the relative poses implied by

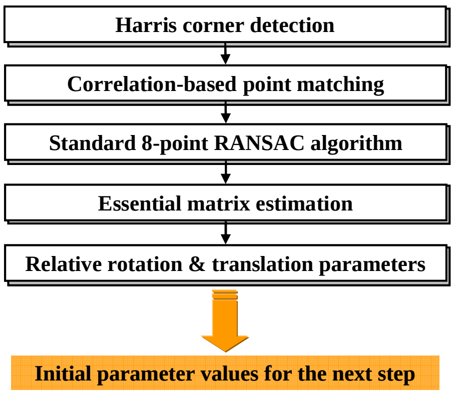

Harris corner detection
Correlation-based point matching
Standard 8-point RANSAC algorithm
Essential matrix estimation
Relative rotation & translation parameters
Initial parameter values for the next step

Fig. 4. Initial estimation of camera parameters

each combination. The correct combination is used as an initial parameter for the fine approach. Fig. 4 shows a block diagram for the estimation of initial camera parameters.

2.1.3 Contour Detection

Canny edge detector [21] is applied to each image, and the detected edges are linked into 1-pixel width contours using an edge linking algorithm [22]. In this process, more than two of line segments in 3D space may be linked as one segment. However, these should be divided into the separate segments in order to establish a precise correspondence. Fig. 5 shows that two projected contour are segmented separately by using the normal vector of the plane composed of 3D line segments and the camera center.

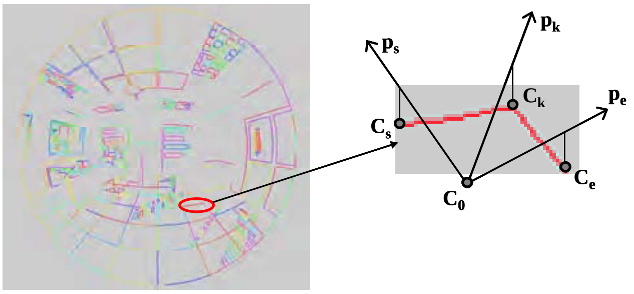


Fig. 5. Contour segmentation by normal vector of 3D plane

In Fig. 5, the linked contour is composed of the projection of two line segments in the 3D space. Here, $\mathbf{C}_s$ and $\mathbf{C}_e$ are two endpoints of the contour, $\mathbf{p}_s$ and $\mathbf{p}_e$ are the incident 3D vectors toward the camera center $\mathbf{C}_0$, which are projected into $\mathbf{C}_s$ and $\mathbf{C}_e$, respectively. In order to segment the contour into two parts, we trace it from its start point $\mathbf{C}_s$ to its end $\mathbf{C}_e$. 3D plane consisting of the line segment and the camera center is defined, and its normal vector is computed through the cross product of two vectors: $\mathbf{p}_s \times \mathbf{p}_k$. When the variance of its normal vector is greater than a predefined threshold, we can find a point $\mathbf{C}_k$ where two segments meet each other, and divide finally the linked line into two parts.

2.2 Second Step for Fine Estimate

2.2.1 Contour Matching

The next process for estimating more accurate parameters utilizes the rotation and unit translation vectors obtained in the previous step. In the first frame, the position $\mathbf{C}_0$ of the base view is (0, 0, 0) and the rotation matrix, $\mathbf{R}_0$, is the identity matrix, $\mathbf{I}$. Since the second frame constructs the epipolar plane, it is assigned as the reference view. Fig. 6 shows the geometric relation between the base and the reference views.

We define an angular error function between the epipolar plane and back-projected vectors of endpoints on the contours, by equation (2). In Fig. 6, $\mathbf{p}_{i0}$ and $\mathbf{p}_{i1}$ are the back-projected 3D vectors of the i^{th} corresponding point from the base and the reference

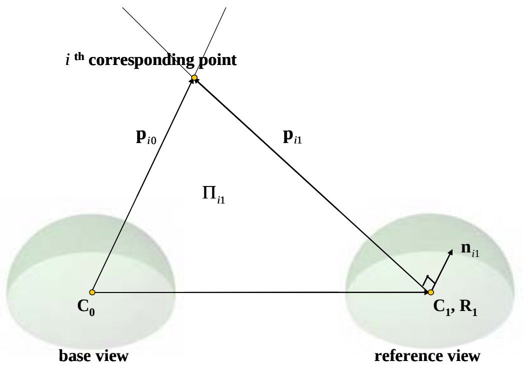


Fig. 6. Geometric relation between base view and reference

view. Π_{i1} is the epipolar plane, defined by $\mathbf{p}_{i0}$, the normal vector $\mathbf{n}_{i1}$ and the unit directional vector $\mathbf{t}_1$ of the reference viewpoint $\mathbf{C}_1$. Conversely, the epipolar plane of Π_{i0} and the normal vector $\mathbf{n}_{i0}$ are calculated geometrically using the relation to $\mathbf{p}_{i1}$. If the rotation matrix $\mathbf{R}_1$ of the reference view and the unit directional vector $\mathbf{t}_1$ of the base view are perfectly precise, equation (2) yields an error of 0; if not, the error is 1.

$$E_{angular}(i) = \frac{(\hat{\mathbf{n}}_{i0} \cdot \hat{\mathbf{p}}_{i0} + \hat{\mathbf{n}}_{i1} \cdot \hat{\mathbf{p}}_{i1})}{2}, \tag{2}$$

where ^ denotes the unit vector. The corresponding contour can be selected as the contour that minimizes $E_{angular}$.

2.2.2 Establishing Correspondences and Recomputing Epipolar Geometry

Given a set of points on the corresponding contour, we determine the correspondences between the pair of point sets by using the cost function, E_{error} as follows:

$$\begin{aligned} &E_{error} = \alpha E_{epi} + \beta E_{corr}, \\ &E_{epi} = \|x_0' F x_1\| / \sigma_k, \;\; E_{corr} = \frac{1 - corr(x_0, x_1)}{2}, \end{aligned} \tag{3}$$

where α and β are the weights used to alter the relative significance of each score. In the simulation results on various frames, we determine two weights of the cost function as: α = 0.7 and β = 0.3. The corresponding point on the contour can be selected as the point that minimizes E_{error}.

In equation (3), the first term, E_{epi} examines how much a pair of match points (x_0, x_1) satisfies the epipolar constraints. σ_k is a factor to normalize its error distribution between 0 and 1. The second term, E_{corr} evaluates the similarity of the intensity distributions in the search windows.

Since a slight rotation and translation of the omnidirectional camera invokes large movement of corresponding points over consecutive frames, the standard rectangular windows commonly used for matching in perspective images are no more adequate to the omnidirectional views. The shape and the size of the search windows vary depending on the distance from the camera center. Direct comparison of different windows is inappropriate for precise matching because of their different size and non-integer coordinates of their boundaries [23]. Differently sized windows would be normalized to some shape and size that enables a direct measure of similarity. We normalize the differently sized windows to ($i \times j$) rectangular window.

$$corr = \frac{\sum_{ij}\left(I_{ij} - \bar{I}\right)\left(J_{ij} - \bar{J}\right)}{\sqrt{\sum_{ij}\left(I_{ij} - \bar{I}\right)^2 \sum_{ij}\left(J_{ij} - \bar{J}\right)^2}}, \tag{4}$$

where I and J are intensities in the image windows, and $\bar{I}$ and $\bar{J}$ are average intensities in the windows.

The epipolar geometry is recomputed by at least 8 match points. The re-estimated essential matrix and camera parameters are iteratively updated for the contours and their correspondences.

3 Experimental Results

In order to examine the accuracy of the camera parameters estimation of our presented algorithm, we use two synthetic images (768×768) which were obtained by rotating and translating the camera with POV-Ray rendering software [24]. In the simulation, we can compare exactly the obtain results with the actual parameter values. Fig. 7 shows input omnidirectional images by the equidistance model with spherical projection.

Since the peripherical region in the omnidirectional image is much more distorted than the center, it is difficult to establish a precise correspondence in that part. Therefore, red points representing the inliers are located mainly in the center. Fig. 8 shows the 8-point RANSCAC results to estimate the initial camera parameters.

We extract 458 and 451 contours extracted in the base and reference views by Canny edge detector, as shown in Fig. 9. In the edge linking process, the contours with less than 30 pixels were discarded, and we set the threshold for the contour segmentation to 2°. As described in contour detection process, it determines whether two line segments were merged into one line or not. The numbers of candidates for corresponding contours were 68 and 70 in each view, and we establish finally 20 pairs of the corresponding contours between two views in Fig. 10.

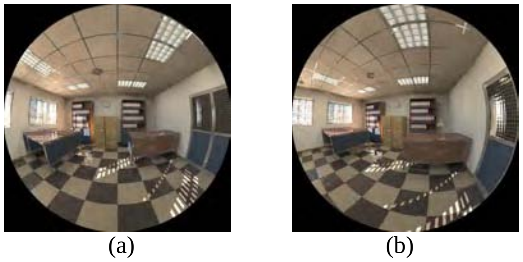


Fig. 7. Input omnidirectional images by equidistance projection model, (a) base view, (b) reference view

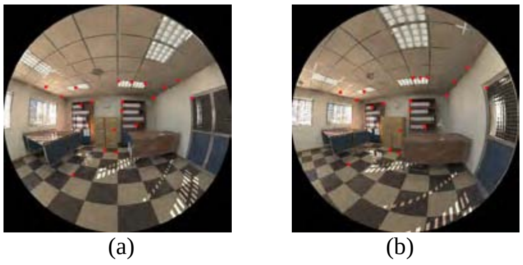


Fig. 8. 19 inlier points by 8-point RANSAC for initial parameters estimation, (a) base view, (b) reference view

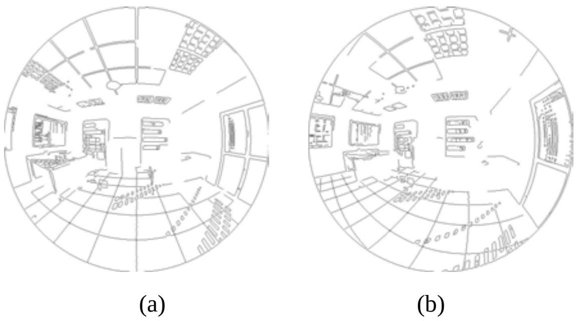


Fig. 9. Detected edges by Canny operator, (a) base view, (b) reference view

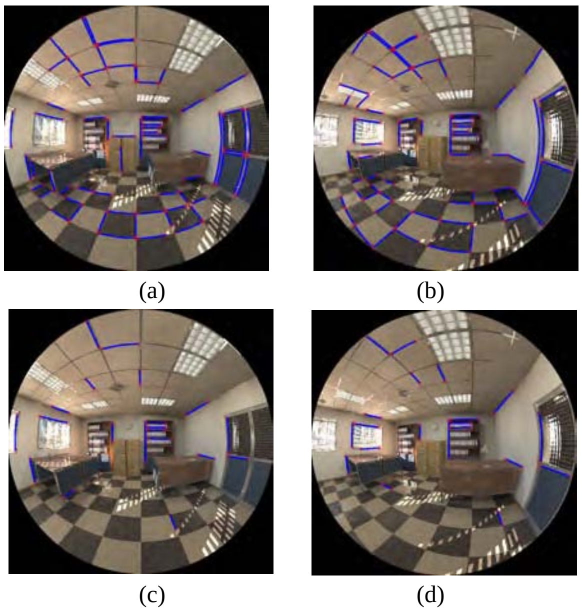


Fig. 10. Edge linking (first row) and contour matching (second row) results of base (left) and reference view (right)

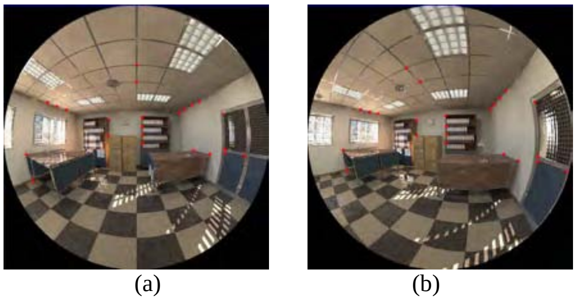


Fig. 11. Correspondences on the contour, (a) base view, (b) reference view

Table 1. Simulation results on input views

	R / error	**t** / error
input parameter	-15.00° / -	(0.97, 0, 0.24) / -
coarse estimation	-13.83° / 1.17°	(0.96, -0.11, 0.26) / 3.72°
fine estimation	-15.07° / 0.07°	(0.97, 0.002, 0.24) / 0.00°

Fig. 11 presents the number of match points on the corresponding contours is 23, and the simulation results are summarized in Table 1. The camera of the reference view are rotated -15° around y-axis and translated to the direction $\mathbf{t}$=(0.97, 0, 0.24). Table 1 shows the proposed algorithm achieves a precise camera calibration.

4 Conclusion

This paper presents a two-step approach to contour matching in architectural scenes captured by the omnidirectional camera. The first step computes the initial estimate of the camera parameters from corner points and correlation-based matching. In the second step, by using a minimum angular error of endpoints of each contour, we establish accurate correspondences among the detected contours. In this process, the initial parameters are refined iteratively from the matched contour set. The simulation results demonstrate that the algorithm precisely estimates the extrinsic parameters of the omnidirectional camera by contour matching.

Acknowledgments. This research was supported by Seoul Future Contents Convergence (SFCC) Cluster established by Seoul R&BD Program, and by the Korea Culture and Content Agency on the CT project.

References

1. Brauer-Burchardt, C., Voss, K.: A new algorithm to correct fish-eye and strong wide-angle-lens-distortion from single images. In: Proc. ICIP, pp. 225–228 (2001)
2. Devernay, F., Faugeras, O.: Straight lines have to be straight. Machine Vision and Applications 13(1), 14–24 (2001)

3. Basu, A., Licardie, S.: Alternative models for fish-eye lenses. Pattern Recognition Letters 16, 433–441 (1995)
4. Xiong, Y., Turkowski, K.: Creating image based VR using a self-calibrating fisheye lens. In: Proc. of Computer Vision and Pattern Recognition, pp. 237–243 (1997)
5. Sato, I., Sato, Y., Ikeuchi, K.: Acquiring a radiance distribution to superimpose virtual objects onto a real scene. IEEE Trans. on Visualization and Com. Graphics 5(1), 1–12 (1999)
6. Shah, S., Aggarwal, J.: Intrinsic parameter calibration procedure for a (high distortion) fish-eye lens camera with distortion model and accuracy estimation. Pattern Recognition 29(11), 1775–1788 (1996)
7. Bakstein, H., Pajdla, T.: Panoramic mosaicing with a 180° field of view lens. In: Proc. IEEE Workshop on Omnidirectional Vision, pp. 60–67. IEEE Computer Society Press, Los Alamitos (2002)
8. Micusik, B.: Two-View Geometry of Omnidirectional Cameras. PhD. Thesis Czech Technical University (2004)
9. Thirthala, S., Pollefeys, M.: Multi-view geometry of 1D radial cameras and its application to omnidirectional camera calibration. In: Proc. ICCV, pp. 1539–1546 (2005)
10. Claus, D., Fitzgibon, A.W.: A rational function lens distortion model for general cameras. In: Proc. CVPR, pp. 213–219 (2005)
11. Barreto, J.P., Daniilidis, K.: Fundamental matrix for cameras with radial distortion. In: Proc. ICCV, pp. 625–632 (2005)
12. Kannala, J., Brandt, S.S.: A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE Trans. on Pattern Analysis and Machine Intelligence 28(8), 1335–1340 (2006)
13. Geyer, C., Daniilidis, K.: Paracatadioptric camera calibration. IEEE Trans. on Pattern Analysis and Machine Intelligence 24(4), 687–695 (2002)
14. Barreto, J. P., Araujo, H.: Paracatadioptric camera calibration using lines. Proc. ICCV (2003) 1359-1365.
15. Han, J., Park, J.: Contour matching using epipolar geometry. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(4), 358–370 (2000)
16. Strickland, R.N., Mao, Z.: Contour motion estimation using relaxation matching with a smoothness constraint on the velocity field. Computer Vision, Graphics, and Image Processing: Image Understanding 60(2), 157–167 (1994)
17. Taylor, C.J., Kriegman, D.J.: Structure and motion from line segments in multiple images. IEEE Trans. on Pattern Analysis and Machine Intelligence 17(11), 1021–1032 (1995)
18. Hwang, Y., Lee, J., Hong, H.: Two-step calibration algorithm to estimate the extrinsic parameters of the omnidirectional camera. Optical Eng. 47 (to be published, 2008)
19. Hartley, R.: In Defense of the 8-Point Algorithm. In: Proc. 5th Int'l Conf. Computer Vision, pp. 1064–1070 (1995)
20. Harris, C.G., Stephens, M.J.: A combined corner and edge detector. In: Proc. 4th Alvey Vision Conf., pp. 147–151 (1988)
21. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)
22. Wall, K., Daniclson, P.: A fast sequential method for polygonal approximation of digitized curves. Graphics Image Process 28, 220–227 (1984)
23. Svoboda, T., Pajdla, T.: Matching in catadioptric images with appropriate windows, and outliers removal. In: Proc. 9th Int. Conf. Computer Analysis of Images and Patterns, pp. 733–740 (2001)
24. http://www.ignorancia.org

A Progressive Edge-Based Stereo Correspondence Method

Xiaoyuan Su and Taghi M. Khoshgoftaar

Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431
xsu@fau.edu, taghi@cse.fau.edu

Abstract. Local stereo correspondence is usually not satisfactory because neither big window nor small window based methods can accurately match densely-textured and textureless regions at the same time. In this paper, we present a progressive edge-based stereo matching algorithm, in which big window and small window based matches are progressively integrated based on the edges of disparity map of a big window based matching. In addition, an arbitrarily-shaped window based matching is used for the regions where big windows and small windows can not find matches, and a novel optimization method, progressive outlier remover, is used to effectively remove outliers and noise. Empirical results show that our method is comparable to some state-of-the-art stereo correspondence algorithms.

1 Introduction

Stereo correspondence is an active research topic. The main task of stereo correspondence is to find the disparity map between a pair of images taken from two different orientations on the same scene. Accurate stereo matching remains a difficult vision problem, especially for textureless regions, disparity discontinuity, and occlusions [1]. Stereo correspondence methods roughly fall into two categories. Local stereo matching methods (window-based) capture disparities only using intensity values within a finite neighboring window. Global stereo correspondence methods such as graph cut [2] and belief propagation [1] are used to optimize the disparity map through various minimization techniques of energy that considers matching cost, disparity discontinuities, and occlusion.

For local stereo matching, small-window based matching can more accurately capture disparity in densely-textured regions, but it produces noisy disparities in textureless regions; while big-window matching produces smooth disparities in textureless regions, but is difficult to get accurate disparities for densely-textured regions. Some algorithms have been proposed to capture disparity values for densely-textured regions, such as variable windows [3], and rod-shaped shiftable windows [4].

In an attempt to accurately match stereo for both densely-textured and textureless regions, we propose a progressive edge-based stereo matching method. The main idea is to progressively integrate big-window matching and small-window matching using the edges of a disparity map from the big-window stereo matching, so that we can match densely-textured and textureless regions at the same time. An arbitrarily-shaped windows matching, which has arbitrary shapes and orientations, is applied to the regions where a regular local stereo matching (either small window or big window matching) fails to make stereo matches.

Instead of using energy minimization based optimization, we propose an optimization method called *progressive outlier remover (POR)* to optimize the disparity map. When a disparity value is surrounded by different disparities, it will be replaced by its neighbors' average disparity when certain conditions are met. We progressively vary the distance values between the current pixel and its neighbors and use threshold values to avoid over-pruning. *POR* is similar to a diffusion-based technique [5] with respect to its smoothing out outliers.

To evaluate the performance of a stereo algorithm, a commonly-used approach is to compute the error rate with respect to some ground truth of the disparity maps [6].

$$B = \frac{1}{N}\sum_{(x,y)}(|d_C(x,y) - d_T(x,y)| > \delta_d)\cdot \tag{1}$$

where N is the total number of pixels, $d_C(x, y)$ is the computed disparity map and $d_T(x, y)$ is the ground truth map, δ_d is a disparity error threshold.

We work on the *Middlebury* stereo data and evaluate the performance of our algorithm in terms of the accuracy for all regions, non-occluded regions and disparity discontinuity regions of the resulting disparity maps against the ground-truth according to the *Middlebury* test bed [6].

The framework of our algorithm is in Section 2. The experimental design and result are in Section 3. Our conclusions are in Section 4.

2 Framework

As a local stereo matching method, our basic idea is to integrate big window and small window matching with the help of edges, which are extracted from the disparity map of a big window matching. An arbitrarily-shaped window matching is used for the regions where a regular local matching fails. We use a progressive outlier remover to effectively remove outliers and optimize the disparities.

2.1 Local Stereo Matching

A local stereo matching method seeks to estimate disparity at a pixel in one image (reference image) by comparing the intensity values of a small region (usually a square window) with a series of same-sized-and-shaped regions in the other image (matching image) along the same scanline. The correspondence between a pixel (x, y) in reference image *R* and a pixel (x', y') in the matching image *M* is given by

$$x' = x + dis(x,y), \qquad y' = y\,. \tag{2}$$

where $dis(x, y)$ is the disparity value at the point (x, y).

We use root mean squared error (*RMSE*) as the matching metric

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[R_i(x,y) - M_i(x',y)]^2} \quad . \tag{3}$$

where N is the total number of pixels in a window, $R_i(x, y)$ and $M_i(x', y)$ are intensity values of pixels in the window of the reference image and matching image. The advantage of using *RMSE* is that we can use a universal threshold value for different window sizes to determine a match or non-match, without trying and choosing different truncation values as other metrics such as normalized cross correlation (*NCC*) and sum of squared differences (*SSD*) do.

For each pixel in each scanline in the reference image, we seek the most similar pixel in the same scanline of the matching image, in terms of the minimum *RMSE*. If this value is smaller than a threshold value, we conclude that there is a match between the pixels and then calculate their difference along the horizontal axis as the disparity value, $dis(x,y)=x'-x$ (Equation 2). Otherwise, we report there is no match here. A disparity map has the disparity values for every pixel in the reference image.

2.2 Arbitrarily-Shaped Windows

We propose an arbitrarily-shaped window based stereo matching to accurately capture disparity values for densely-textured regions. Our arbitrarily-shaped-window strategy is to try out all kinds of shapes and orientations and pick the winning shape that has the minimum similarity value in terms of *RMSE*.

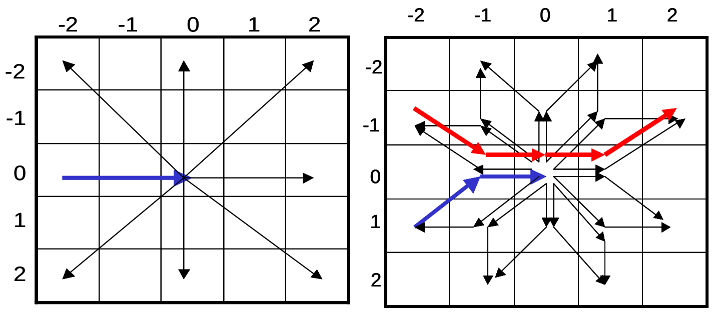


Fig. 1. Arbitrarily-shaped windows (a) scenario A, (b) scenario B

The arbitrary shapes or orientations come from two scenarios, scenario A and scenario B (Fig. 1(a)(b)). Each shape or orientation is actually a unique combination of five neighboring pixels inside a 5*5 window with the pixel (0, 0) in the middle, which is the active matching pixel.

In scenario A (Figure 1(a)), when the first three pixels are (-2, 0), (-1, 0) and (0, 0), and our searching route for other pixels to form a unique 5-pixel combination ends at one of the other peripheral points, we will have seven different shapes or orientations. Next, starting from another peripheral pixel and ending at a different peripheral one, we will have six (excluding the shape/orientation found in the previous search). Continuing this search until every peripheral starting pixel is tried results in a total of

$\Sigma^7{}_1(i)=28$ shapes/orientations for scenario A. For example, the horizontal window across the point (0, 0) can be represented as (-2, 0), (-1, 0), (0, 0), (1, 0), (2, 0), where the (*x*, *y*) values of the points are the horizontal and vertical differences from the active matching pixel (0, 0). In scenario B (Figure 1(b)), we use the remaining peripheral pixels of the 5*5 square from scenario A. When our first three pixels are (-2, -1), (-1, 0) and (0, 0), we will have 15 different shapes or orientations. Taking other searching routes to form unique 5-pixel combinations, and keeping (0, 0) as the central pixel and start point and end point peripheral pixels of the square, we will get $\Sigma^{15}{}_1(i)=120$ unique shapes or orientations. For the highlighted example of Figure 1(b), the window is represented as (-2, -1), (-1, 0), (0, 0), (1, 0), (2, -1).

Summed from these two scenarios, we will have a total of 148 different shapes/orientations to pick a 5-pixel arbitrarily-shaped window.

We use five as the pixel number of an arbitrarily-shaped window, because three will be too small to compute reliable matching costs and seven and more will be cost prohibitive. Comparing with regular square windows, the computation time for an arbitrarily-shaped window based matching is the single 5-pixel matching time multiplied by 148 (5*148), which is equivalent to matching with a square window of size 27 (27*27). By applying the arbitrarily-shaped windows only for the regions where a regular window based matching can not find matches (less than 10%), the complexity is greatly reduced.

Figure 2 illustrates the effect of using arbitrarily-shaped windows on the stereo data *Tsukuba*, and its combinational usage with a regular square window stereo.

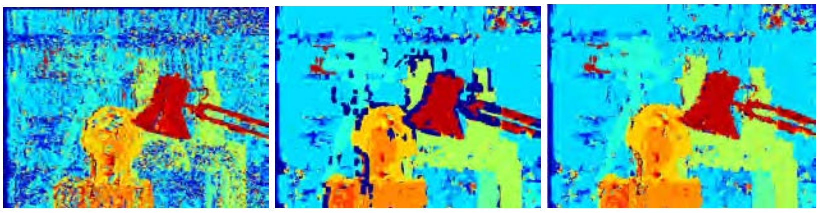


Fig. 2. (a) Arbitrarily-shaped window (*Wa*) matching for the stereo data Tsukuba, (b) window 7*7 (*W7*) matching, (c) window 7*7 + arbitrarily-shaped window (*W7+Wa*) matching

2.3 Progressive Edge-Based Stereo Matching

The main steps of our progressive edge-based stereo matching are illustrated in Figure 3. As an example of our stereo matching on the stereo data *Teddy*, Figure 3(a) is the disparity map from a small-window matching (*win_small*) of size 3*3 (*W3*); (b) is the disparity map from a big window matching (*win_big*) of size 25*25 (*W25*); (c) is the disparity map from the arbitrarily-shaped windows matching (*win_arbi, or Wa*); (d) is *W3+Wa* (*W3*, plus *Wa* where *W3* can not make matches); (e) is *W25+Wa*; (f) is the edges of the *W25+Wa* optimized by the *POR*; (g) is the disparity strips combined from (d) and (e) around the edges in (f); (h) is the smoothed disparity map from (g), (i) is the final disparity map optimized from the *POR* optimization method.

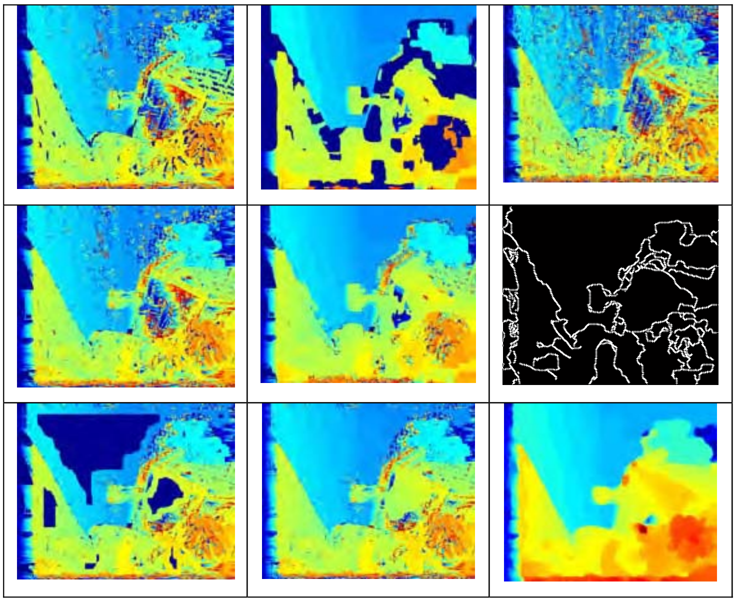


Fig. 3. An illustration of our edge-based stereo correspondence on the stereo data Teddy (top row) (a) W3, (b) W25, (c) Wa; (middle row) (d) W3+Wa, (e) W25+Wa, (f) edges of W25+Wa; (bottom row) (g) strips of W3+Wa and W25+Wa around the edges, (h) smoothed disparities between strips, (i) final disparity map after POR optimization

We use the optimal edge detector *Canny* to detect edges [7]. We use the command *edge(image, 'canny', k)* in MATLAB to get the edge file for the input disparity map image. A smaller *k* value will generate more edges in the binary output edge file, in which 1 represents an edge and 0 otherwise.

When combining the big window and small window matching, with arbitrarily-shaped windows used for regions where a big window or small window can not make matches, we use the disparity values from *small window matching* for the strips around the edges; and use the disparities from *big window matching* for the strips away from the edges (next to the *small window matching* strips). The width of *small window matching* strips at each side of the edges and that of the neighboring *big window matching* strips are $W_{strip}=[size(win\_big)-size(win\_small)]/2+1$.

We enforce the disparity continuity between the strips of *big window matching* using a disparity averaging scheme. Suppose a pixel *(x, y)* inside the region is to be smoothed, the disparity value *dis(x, y)* depends on the closest disparity values of four directions on its neighboring strips. Given horizontally left and right disparities $dis(x_1,y)$ and $dis(x_2, y)$, and vertically above and below disparities $dis(x, y_1)$ and $dis(x, y_2)$), we calculate the disparity value *dis(x,y)* by $dis(x,y)=\frac{1}{2}[dis_x(x,y)+dis_y(x,y)]$, where $dis_x(x,y)=\frac{dis(x_2,y)-dis(x_1,y)}{x_2-x_1}(x-x_1)+dis(x_1,y)$, $dis_y(x,y)=\frac{dis(x,y_2)-dis(x,y_1)}{y_2-y_1}(y-y_1)+dis(x,y_1)$.

2.4 The Progressive Outlier Remover Optimization

Our *Progressive Outlier Remover (POR)* optimization is based on the disparity continuity assumption: in a small region, when a disparity value is greatly different from its surroundings, it is deemed as an outlier and should be replaced or optimized.

Illustrated in Figure 4, for each value in the disparity map, we compare it with four equally-distanced neighbors in four directions, separately, one kind of neighbors are directly above, below, left, and right neighbors (Figure 4(a)), and another kind are four corners of a square where the current pixel is centered (Figure 4(b)).

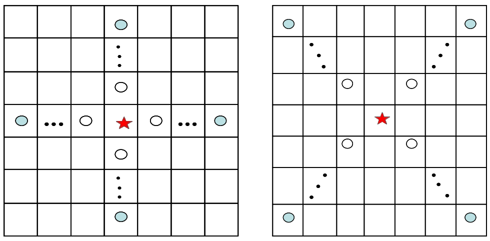


Fig. 4. An illustration of POR optimization: a disparity outlier is replaced with an average of its four neighbors' disparities in either scenario A (a) or scenario B (b), and the neighborhood distance is adjustable

When the disparity of the central pixel is not equal to any of its neighbors' disparities, and its difference from the average of the neighbors' disparities is bigger than a threshold, it will be replaced by the average of the neighbors' disparities. The threshold *T* is proportional to the product of the neighborhood distance *d* to the central pixel and the standard deviation σ of the four neighbors' disparity values

$$T = k \times d \times \sigma, \quad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2 - (\frac{1}{N}\sum_{i=1}^{N} x_i)^2} \; . \tag{4}$$

where N = 4, and k takes a value of 1 or 0.5. A k value of 1 represents relatively stricter thresholds than that of 0.5.

The *POR* optimization algorithm takes two parameters, maximum neighborhood distance *d* (a value usually from 2 to 20), and a decremental rate *R* (a value of 2/3, 1/2 or 2/5) for getting decrement iteration numbers in different rounds of iterations. For each round *i* (i=1, 2, 3, ……), we calculate the iteration number n_i, with initialization of $n_1=d$, and the iterations will not stop until n_i =1.

$$n_i = \lfloor n_i \times R^{i-1} - 0.5 \rfloor + 1 \; . \tag{5}$$

For example, when we have *(d, R)*=(20, 2/3), we will have 8 rounds of optimizations, but with *R=1/2* and *R=2/5*, we will have 6 and 4 respectively. For each round of iterations, we have 1 and 0.5 as the *k* values alternatively, i.e., we have (20, 1), (20, 0.5), (8, 1), (8, 0.5), (3, 1), (3, 0.5), and (1, 1) as the (n_i, k) combinations for *(d, R)*=(20, 2/5), and we do not have (1, 0.5) for n_i =1. For each round, with the iteration number n_i, the *POR* algorithm will have iteration *i* from 1 to n_i, each of which has the neighborhood distance of *i*, and have the threshold for outlier removal of $k*i*\sigma_i$ defined in Equation 4.

A complete *POR* algorithm is in Figure 5. An example of applying this algorithm on the stereo image *Venus* is shown in Figure 6.

```
Algorithm: progressive outlier remover (POR) (d, R, dis(x,y))
for each $n_i$ of (i=1, $n_1$=d; $n_i \geq 1$; $n_i = \lfloor n_i * R^{i-1} - 0.5 \rfloor + 1$, i++)
  for k_round=1:2
    if (k_round==1) k=1; else k=0.5;  end if
    for (n=1; n<= $n_i$; n++)
      d=n (neighborhood distance)
      $T_A$=k*d*$\sigma_A$, $T_B$=k*d*$\sigma_B$ (thresholds for A, B scenarios)
      for each pixel dis(x, y) in the disparity map,
        if(dis(x,y)$\neq \forall \in$ {dis(x-d,y),dis(x+d,y),dis(x,y-d),
        dis(x,y+d)} && abs(dis(x,y)-$\mu_A$)>$T_A$)
          dis(x,y)=(dis(x-d,y)+dis(x+d,y)+dis(x,y-d)
            +dis(x,y+d))/4
        end if
        if(dis(x,y)$\neq \forall \in$ {dis(x-d,y-d),dis(x+d,y+d),dis(x+d,y-d),
        dis(x-d,y+d)} && abs(dis(x,y)-$\mu_B$)>$T_B$)
          dis(x,y)=(dis(x-d, y-d)+dis(x+d, y+d)+
            dis(x+d, y-d)+dis(x-d, y+d))/4
        end if
      end for
    end for
  end for
end for
```

Fig. 5. The Progressive Outlier Remover (POR) optimization algorithm

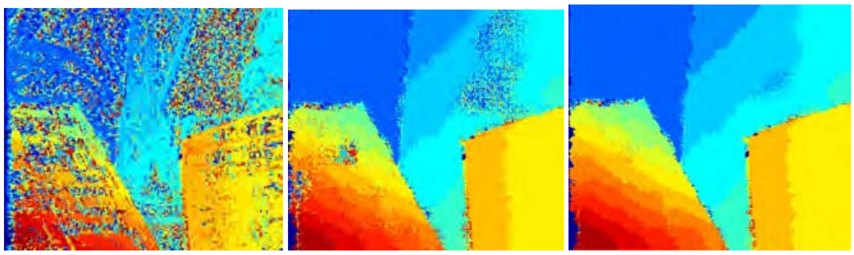


Fig. 6. Applying the POR optimization on the stereo data Venus (a) disparity map of W3+Wa (win_size 3*3 +arbitrarily-shaped windows), (b) (c) after the first two rounds of POR optimizations with d=14, R=1/2

3 Experimental Design and Results

We work on four *Middlebury* stereo images, *Tsukuba*, *Venus*, *Teddy*, and *Cones* for quantitative evaluation, which are the benchmark data for stereo correspondence algorithms [8]. We evaluate our algorithm in terms of the percentage of bad pixels, i.e., pixels whose absolute disparity error is greater than a threshold (such as 1 and 0.5). We calculate percentages for (1) pixels in non-occluded regions, (2) all pixels and (3) pixels near disparity discontinuities, and ignore a border of 10 pixels for *Venus*, and 18 for *Tsukuba* when computing statistics, according to the evaluation standard on the *Middlebury* stereo [6].

For window based stereo matching, we use *RMSE* as the cost metric, and use 15 as a universal cut off value for determining a correspondence, based on our preliminarily experiments. As described in Section 2, our big window and small window matching are actually *win_big+win_arbi* and *win_small+win_arbi*, where the arbitrarily-shaped windows matching is applied to the regions that a regular big window or small window matching fail to make matches. When optimizing the disparity map using the *POR* optimization algorithm, we use different parameters of (*d*, *R*) for different stereo data, according to the distribution of densely-textured and texture-less regions.

The data *Teddy* has a big textureless area, which a regular local stereo algorithm has difficulty to deal with. By using a big window match of size 25, optimized by *POR* of *d=12*, the big hole in the textureless regions of the disparity map is well smoothed (Figure 3). For the representative densely-textured data *Tsukuba*, we use relatively small window sizes (with big window size of 5) and small parameters for the *POR* optimization (d=3), to avoid the loss of the accurate disparities for the delicate textures.

The overall evaluation of our algorithm is in Table 1, Table 2 and Figure 7. In Table 1, we find that there is apparent improvement of stereo correspondence using edge-based strategy over that without using it, the later of which simply uses a *win_small+win_arbi* matching and gets it optimized by the *POR* algorithm.

By the time of submission, the average rankings of our algorithm on the *Middlebury* stereo evaluation system [8] are No. 16 for error threshold of 0.5, and No. 22 for error threshold of 1, out of 29 submissions to the system, most of which are results from existing state-of-the-art algorithms. Compared with other disparity optimization

Table 1. Improvement of using progressive edge-based stereo matching over without using edge-based strategy (in terms of percentage of bad pixels for non-occluded, all and disparity discontinuity regions, with threshold of 1)

	Tsukuba			Venus		
	nonocc	All	disc	nonocc	all	disc
w/o edge-based	2.98	4.92	15.1	2.47	3.48	27.5
edge-based	**2.73**	**4.65**	**13.9**	**2.25**	**3.24**	**27.4**
	Teddy			Cones		
	nonocc	All	disc	nonocc	all	disc
w/o edge-based	14.5	22.4	33.0	9.78	17.5	21.3
edge-based	**14.3**	**22.1**	**30.2**	**7.63**	**16.1**	**19.7**

Table 2. Overall evaluation of our algorithm on the Middlebury data (in terms of percentage of bad pixels for non-occluded, all, and disparity discontinuity regions; the subscripts of the results are our rankings amongst other state-of-the-art algorithms on the Middlebury stereo system, with thresholds 1 and 0.5)

	Tsukuba			Venus		
	nonocc	All	disc	nonocc	all	disc
Thre=1	2.73_{19}	4.65_{20}	13.9_{24}	2.25_{21}	3.24_{21}	27.4_{27}
Thre=0.5	8.26_{5}	10.4_{6}	23.0_{21}	8.57_{13}	9.67_{13}	33.9_{25}
	Teddy			Cones		
	Nonocc	all	disc	nonocc	all	disc
Thre=1	14.3_{22}	22.1_{22}	30.2_{26}	7.63_{20}	16.1_{20}	19.7_{22}
Thre=0.5	24.3_{20}	32.2_{22}	43.0_{24}	15.0_{17}	23.0_{17}	28.0_{20}

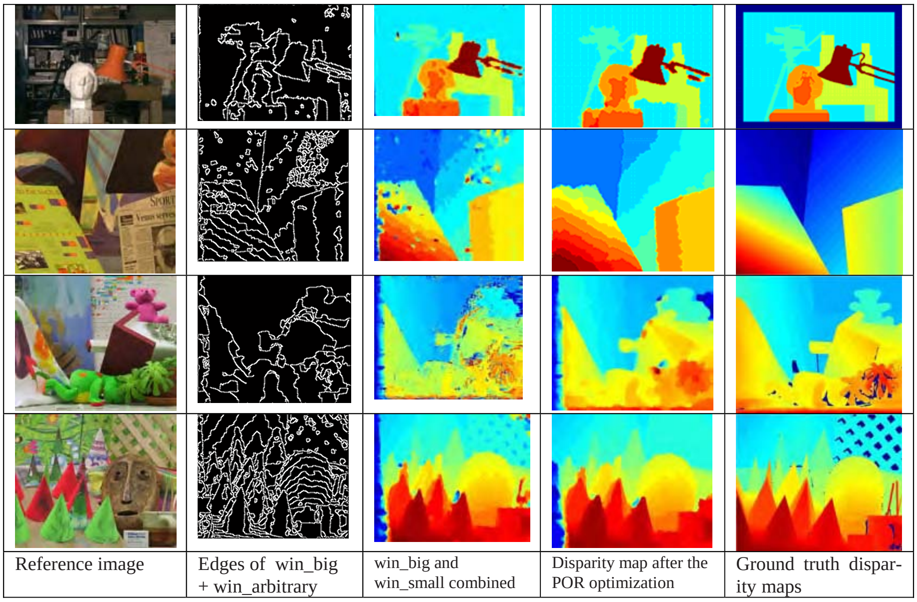

Reference image
Edges of win_big + win_arbitrary
win_big and win_small combined
Disparity map after the POR optimization
Ground truth disparity maps

Fig. 7. The results of our progressive edge-based stereo matching algorithm (from top to down: Tsukuba, Venus, Teddy, and Cones. Same color on different maps does not necessarily represent the same disparity)

methods, our algorithm is better than scanline optimization [6] and comparable to graph cuts using alpha-beta swaps [9] and dynamic programming [10] on the new version of *Middlebury* evaluation. On the previous version of *Middlebury* evaluation data, our algorithm is better than other window-based stereo correspondence algorithms such as the pixel-to-pixel algorithm [11], the discontinuity preserving algorithm [12], and the variable window algorithm [3].

With the threshold of 0.5, our progressive edge-based stereo matching has the average rankings of No. 14 for the non-occluded regions and all regions apiece, but has the average ranking of No. 22 for the disparity discontinuity regions. We plan to improve this algorithm in our future work, especially for its performance of matching the disparity discontinuity regions.

The parameter settings of our progressive edge-based stereo matching can be unified for different stereo data by analyzing the distributions of highly textured and textureless regions. We will investigate this in the future. It will also be interesting to combine our local stereo method with global optimization algorithms such as graph cut and belief propagation.

4 Conclusions

Stereo correspondence is a difficult vision problem, especially for textureless regions, disparity discontinuity and occlusions. In order to address the problem that regular window based stereo matching methods fail to make accurate matches for densely textured and textureless regions at the same time, we propose a progressive edge-based stereo matching method to unify big window matching and small window matching with the help of edges. The edges are extracted from the disparity map of the big window matching using an optimal edge detector. An arbitrarily-shaped windows based matching is used for the regions where a regular big window or small window matching can not find matches. Instead of using an energy-minimization based optimization, we propose a novel optimization algorithm called *progressive outlier remover (POR)*, which progressively replaces an outlier disparity value with the average of its surrounding ones when certain constraints are met, and effectively removes outliers and enforces disparity continuity. Experiments on the standard *Middlebury* stereo data show, that our progressive edge-based stereo matching method performs comparable with some state-of-the-art stereo matching algorithms in terms of matching accuracy for non-occluded regions and all regions.

References

1. Sun, J., Zheng, N-N., Shum, H-Y.: Stereo Matching Using Belief Propagation. PAMI 25(7) (2003)
2. Kolmogorov, V., Zabih, R.: Computing Visual Correspondence with Occlusions using Graph Cuts. ICCV (2001)
3. Veksler, O.: Fast Variable Window for stereo correspondence Using Integral Images. CVPR (2003)
4. Kim, J.C., Lee, K.M., Choi, B.T., Lee, S.U.: A Dense Stereo Matching Using Two-Pass Dynamic Programming with Generalized Ground Control Points. CVPR (2005)
5. Scharstein, D., Szeliski, R.: Stero Matching with Non-linear Diffusion. IJCV 28(2), 155–174 (1998)
6. Scharstein, D., Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. IJCV 47(1), 7–42 (2002)
7. Canny, J.: A Computational Approach To Edge Detection. PAMI 8, 679–714 (1986)
8. Middlebury stereo, http://www.middlebury.edu/stereo
9. Boykov, Y., Veksler, O., Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. PAMI 23(11) (2001)
10. Bobick, A.F., Intille, S.S.: Large Occlusion Stereo. IJCV 33(3), 181–200 (1999)
11. Birchfield, S., Tomasi, C.: Depth Discontinuities by Pixel-to-Pixel Stereo. ICCV (1998)
12. Agrawal, M., Davis, L.: Window-Based Discontinuity Preserving Stereo. CVPR (2004)

Creating Stereoscopic (3D) Video from a 2D Monocular Video Stream

Xiaokun Li[1], Roger Xu[1], Jin Zhou[2], and Baoxin Li[2]

[1] Intelligent Automation, Inc, Rockville, MD 20855
[2] Arizona State University, Tempe, AZ 85287

Abstract. It is a challenge to generate stereoscopic (3D) video through a single moving camera under widely varying conditions. We propose an efficient approach to create true stereoscopic video from a monocular video stream captured under various moving conditions. The approach contains three major steps. First, we apply Harris' corner detector to detect distinctive feature points from a pair of image frames selected from the incoming video captured by a moving camera. Second, according to the consecutive property of the video, a local-window search based algorithm is developed for fast and accurate feature correspondence between the two image frames. Third, a hierarchical image rectification technique is designed to guarantee the success in creating a true and visually-comfortable stereo image for each incoming image frame. Besides, a software-based video stabilization algorithm is also developed for improved stereo video generation performance. Extensive tests using real video collected under various situations were performed for performance evaluation of the proposed approach.

1 Introduction

In gaming and TV programs, 3D video effects are one of the most attractive features. 3D video techniques have also found wide civilian applications, such as medical operations, microscopy, scientific data display, and CAD/CAM. Military applications of 3D techniques include battlefield reconnaissance and surveillance. Conventional computer-based stereo vision techniques, although studied for many years, still have many limitations. For example, a conventional stereo vision system requires two identical cameras, a narrow baseline, fixed parameter settings and positions. It is only suitable for short-range scenes. However, in real world, camera motion is often nonstationary and viewpoints of the camera are different from time to time. Furthermore, parameter settings of the camera are sometimes unknown and variable during video capturing.

To generate stereo (3D) video captured by a moving camera under widely varying conditions presents a challenge. The main reason is that the camera is obliquely mounted on a platform when the platform moves non-linearly or when the camera parameters vary while the platform is moving. In recent years, many research efforts have been made on stereo generation with uncalibrated cameras. Fusiello *et al.* [1] developed a compact and efficient stereo generation algorithm via image rectification.

However, their approach assumes that the stereo rig is calibrated, which means the intrinsic parameters of the camera pair such as focal length, aspect ratio, and their relative position are already precisely known. Unfortunately, as mentioned earlier, the camera parameters are not readily available and the relative position between the two cameras is difficult to obtain or calibrate in practice. Loop and Zhang at Microsoft Research [2] developed one method to construct stereo images with uncalibrated cameras. Their method mainly relies on stereo matching and the residual distortion may result in poor visualization performance. The method proposed by Hartley & Zisserman [3], [4] for stereo generation from uncalibrated cameras is the most advanced one in the literature to our best knowledge. One important distinction of their method is that it is insensitive to unknown and variable camera parameter settings during image/video capture. However, the quality of the generated stereo images cannot be guaranteed. In some cases, the resulting stereo image may even be corrupted.

Although many efforts have been made on stereo generation with uncalibrated cameras, stereo video generation from a single moving camera is a fairly new research topic. In this paper, we focus on the following stereo vision problem: given two video frames acquired by a moving video camera at two different locations with different viewing angles, we create a stereo image frame based on the two video frames by means of feature extraction and correspondence, and image rectification. The resulting stereo frame gives a viewer a realistic 3D perception, and can be displayed on any type of display devices. Furthermore, for each video frame in the video stream, we can successfully construct a stereo image based on the current frame and a second frame with a predefined constant time delay very rapidly. As a result, stereo video can be generated. The biggest challenge in generation stereo video from the 2D video stream is how to generate a true and eye-comfortable stereo image for each video frame. As can be found in the later part of this paper, our proposed robust and efficient approach can successfully create stereo video in various situations.

2 Algorithm Description

The proposed approach consists of four algorithms executed in a sequential order. The first algorithm is software-based video stabilization. The second is feature extraction which provides an efficient way for image feature extraction by Harris' corner detection. The third is a local-window-based search method to find feature correspondence of input frame pair. The last algorithm is stereo video generation based on image rectification.

2.1 Software-Based Stabilization

When a camera is in motion during video acquisition, the acquired video may contain unexpected jitters that will certainly affect the image rectification process of stereo generation. Such scenario is illustrated in Fig. 1. To show how frame jitters may affect image rectification, we use the example in Fig. 2 (a) for illustration. Suppose

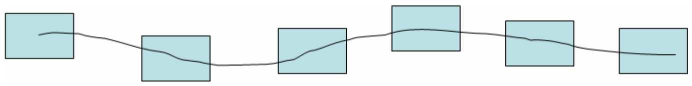


Fig. 1. Frame jittering

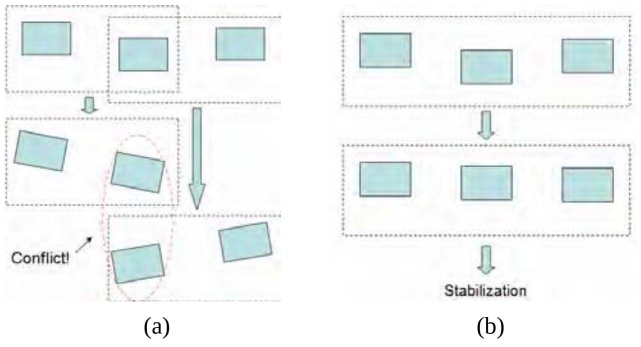


Fig. 2. Conflict and video stabilization

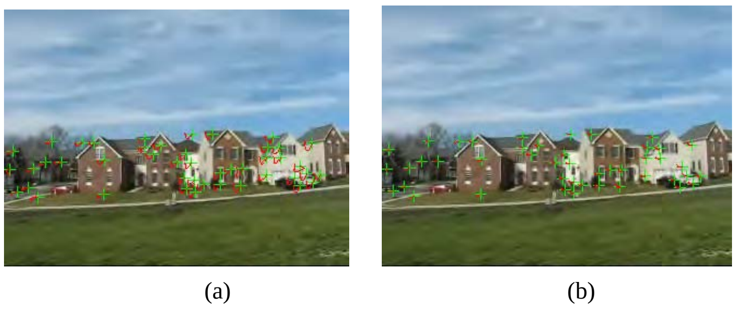


Fig. 3. (a) Left image shows original moving path of feature points; (b) Right image shows the moving path after stabilization

that there are three incoming video frames which are not in a straight line. For the first two image frames, their stereo result is a clockwise rotation of the original frames. On the other hand, for the second and third frames, the stereo result is a counterclockwise rotation of the original frames. The conflict will damage the consistence of the stereo video and bring the viewpoint varying problem. To alleviate the effect of frame-jittering, we developed a method to stabilize the video along the vertical direction. We smooth the scene translation in the video by vertically shifting frames, which makes the stereo generation easier to be performed. Although there exist some hardware based stabilization systems, we still found that the software-based stabilization is very useful due to the following two reasons: **i)** most existing stabilization systems remove only high-frequency jitters, and the vide still contain low-frequency vertical shifting across frames; and **ii)** those hardware-based stabilization systems are unlikely to be mounted on a light-weight platform such as a UAV/UGV.

The objective of our method is to make the frames roughly aligned horizontally, as illustrated in Fig. 2 (b). To stabilize the video, we first subtract the two selected frames. Then, we analyze the residua of the subtraction. The degree of horizontal shift of the two image frames can be estimated by counting the number of horizontal edges and the average distance between the corresponding horizontal edges in the residual image. Based on the estimated horizontal shift, we align the two frames by shifting the second frame up or down with the estimated shifting value which is equal to the average distance value of the corresponding horizontal edges. Fig. 3 (a) shows one example of the tracked feature trajectories from a real-life video sequence. The green crosses are feature points and the red curves are the trajectories of the features across ten frames. Fig. 3 (b) shows the result of the stabilization. As we can see from the result, the red curves in horizontal direction are reduced a lot after stabilization.

2.2 Feature Extraction

To accurately detect image features for stereo, we use Harris' corner detector [8] whose efficiency on corner and edge detection has been proven due to its strong invariance to object rotation, scale, illumination and noise. The Harris corner detector is based on the local auto-correlation function of a signal, where the local auto-correlation function measures the local changes of the signal with patches shifted by a small amount in different directions. The brief introduction of Harris corner detector is given below:

Given a shift ($\Delta x, \Delta y$) and a point (x, y), the auto-correlation function is defined as,

$$c(x,y)=\sum_W [I(x_i,y_i)-I(x_i+\Delta x, y_i+\Delta y)]^2$$

where *I(·, ·)* is denoted as the image function and (x_i, y_i) are the points in the window W (Gaussian) centered on *(x, y).*

The shifted image is approximated by a Taylor expansion truncated to the first order terms,

$$I(x_i+\Delta x, y_i+\Delta y)\approx I(x_i,y_i)+[I_x(x_i,y_i) \quad I_y(x_i,y_i)]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

where *Ix*(·,) and *Iy*(·,) denote the partial derivatives in x and y, respectively.

Substituting the approximation *I*(·, ·) to *c(x,y)* yields,

$$c(x,y)=\sum_W [I(x_i,y_i)-I(x_i+\Delta x, y_i+\Delta y)]^2$$

$$=[\Delta x \quad \Delta y]\begin{bmatrix}\sum_W (I_x(x_x,y_i))^2 & \sum_W I_x(x_i,y_i)I_y(x_i,y_i)\\ \sum_W I_x(x_i,y_i)I_y(x_i,y_i) & \sum_W (I_y(x_i,y_i))^2\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

$$= [\Delta x \quad \Delta y] C(x,y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

where matrix C(x, y) captures the intensity structure of the local neighborhood. Let λ_1, λ_2 be the eigenvalues of matrix C(x, y). The eigenvalues form a rotationally invariant description.

There are three possible results after the eigenvalues are obtained:

1. If both λ_1 and λ_2 are small, which means that the local auto-correlation function is flat (i.e., little change in *c(x, y)* in any direction), so the windowed image region is of approximately constant intensity.
2. If one eigenvalue is large and the other is small, which means the local auto-correlation function is ridge shaped and only local shifts in one direction (along the ridge) cause a small change in *c(x, y)* and a significant change in the orthogonal direction, the point *(x,y)* is on an edge.
3. If both eigenvalues are large, which means the local auto-correlation function is sharply peaked and shifts in any direction will result in a significant increase, the point *(x,y)* is a corner point.

2.3 Feature Correspondence

According to the consecutive property of video, we developed a fast feature correspondence method based on local-window search. The key idea of the local-window based search is summarized as follows:

First, we select some feature points from the first image of a given image pair. These points are selected according to their significance (edge energy) and geometrical distribution, and can be expressed as the following:

$$P_j = \{ p_{ij} \mid i = 1..n_j \}$$

where P_j is the set of the selected feature points of Image j and p_{ij} is ith feature point of Image j.

Second, suppose we need to find feature correspondences between the first image (*t*) and the second image (*s*) of a selected video frame pair, that is, we need to output a set of feature point pairs which assumed to be point correspondences between the two images.

$$PC(t,s) = Matching(P_t, P_s) = \{ (p_{it}, p_{js}) \}$$

To do this, we need to find the best correspondence in the second image for each feature point in the first image. Now, the problem becomes how to compute the matching score for two feature points from an image pair.

Third, since the motions of the scene/objects are continuous in video, for any feature point in the first image its corresponding feature point in the second image should be located very close to its position, which means we do not have to search its corresponding feature point in the whole image space. Based on this observation, we developed a local-window search method for fast feature correspondence. For each selected feature point in the first image, we define a local-window which is centered at this point. Then, we set up a sliding window in the second image, which has the same size of the local-window in the first image, and shift the window in a region of the second image. During the sliding, we compare the difference between the local-window and the sliding window at each position by computing its matching score which is defined as the average difference of pixel intensity in the two windows.

$$Diff\,(n,i) = \frac{1}{area(window)} \sum_{j=1}^{n} \left| I(n) - I(i,\, j) \right|$$

where *I*(*n*) represents the sum of the intensity values of the local window centered at the selected feature point *n* in the first image ; *I*(*i*,*j*) represents the sum of the intensity values of the sliding window centered at the feature point *i* in the second image; *j* is the pixel located in the current sliding window; *i* is the feature point on the sliding trace; *area*(*patch*) is the size of the local window. For the feature point *n*, the feature point in the second image with the minimum difference will be chosen as its best match. In this way, we can find the feature correspondence for each feature point in the first image.

2.4 Hierarchical Rectification for Fast and Robust Stereo Video Generation

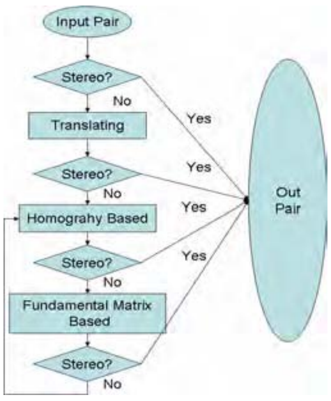


Fig. 4. Hierarchical image rectification

In the introduction section, we knew that the most efficient way on stereo is using fundamental matrix for stereo generation. It is known that the fundamental matrix is computed from point correspondences. As we discussed before, in some cases an incorrect fundamental matrix might be obtained because the fundamental matrix computation has some rigorous assumptions, such as no dominant plane is permitted. In a video steam, there are many cases, where those assumptions do not hold. Therefore, although a true stereo image can be guaranteed with a correct fundamental matrix we cannot completely rely on the fragile and time consuming fundamental matrix for stereo construction. In addition, for many cases we do not need to do any image rectification for an image pair as the pair is a stereo pair already. According to this observation, we developed a hierarchical image rectification strategy as illustrated in Fig. 4 to not only improve the quality and the success rate of stereo but also e save the processing time. Also, to overcome the viewpoint varying problem during stereo construction, we only perform image rectification to the second image of a selected frame pair. In this way, the viewpoint of the generated stereo video will always be the same as the camera viewpoint.

(1) Stereo check

One property of a stereo pair is that all disparities are horizontal. If a frame pair satisfies the constraint that all disparities are horizontal, no rectification is needed. In our stereo generation algorithm, we will first compute the average absolute vertical disparities. If the value is less than a predefined threshold and the average value of the horizontal disparities is closed to the standard baseline value, we assume that the frame pair is a stereo image pair already.

(2) Image Rectification Method 1: Image Translation

If the stereo check fails for the frame pair, we will shift the second image in the horizontal direction to minimize horizontal disparities. Afterwards, we will perform stereo check again. If the result can pass the stereo check, we finish the stereo generation for this image pair. If it fails again, we apply the second method described below.

(3) Image Rectification Method 2: Homography

Here, we make the disparities of the frame pair to be horizontal by applying a homography transformation to the second image of the frame pair. After the homography transformation, the feature points of the second image should have the same horizontal line as the feature points of the first image. We can compute a homography from four correspondences via the following steps:

Algorithm - Homography

Step 1: Select four points from a set of feature points of the first image.
Step 2: Find their correspondence in the second image,
Step 3: Call RANSAC-based model matching method [9] to compute the Homography transformation matrix, H.

In practice, the point correspondences we found may still include some mismatches. Therefore, we use RANSAC to find the best Homography transformation matrix in global optimum. If the stereo construction by Homography fails, we will apply the third method for stereo.

(4) Image Rectification Method 3: Fundamental Matrix

The fundamental matrix based image rectification method is built on our recent work [5-7]. That is, for two selected uncalibrated video frames of the same scene, we rectify them into a standard stereo pair (which is the subject of [5-7]). In this research, we customize our fundamental-matrix-based stereo imaging algorithm for stereo video generation. Instead of rectifying two images into a standard stereo pair, we only calculate H matrix for the second image. In this way, we can solve the “viewpoint varying” problem. Here, we give a very concise description and the basic steps of the stereo construction.

Algorithm - Stereo image generation based on fundamental matrix

Step 1: Estimate fundamental matrix F [2] which represents the spatial difference of the two selected images
Step 2: Compute the rectification matrix H_2 for the second image
Step 3: Rectify each pixel of the second image by multiplying H_2 to generate a wide-baseline or narrow baseline stereo pair

Step 4: Compute the average Z value of the center part of the images and translate them to configure a proper baseline (standard baseline) to construct a true stereo pair

Step 5: If the result cannot pass through stereo check, jump to the Homography based method and output the stereo result of the Homoography based method without stereo check.

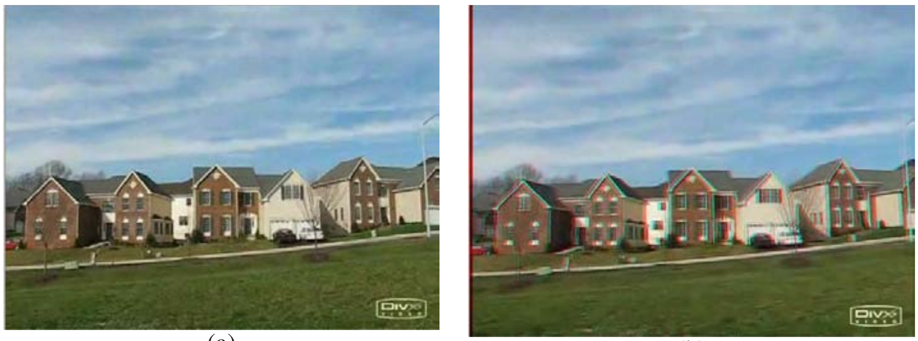
3 Experimental Results

Extensive tests including many side-looking and down-looking scenarios were performed to improve and optimize our proposed stereo algorithm. We summarized our results in this section. The statistic results on side-looking and down-looking video clips are listed in Table 1 and Table 2 separately. Some experimental results are shown in Fig. 5 to Fig. 7. All tests were performed on an ordinary PC (Intel P-4 2.4GHz and 1G memory) with MS Windows XP. For the incoming video with 25fps and 640 by 480 resolution, the average processing time on stereo video generation is 6.2 times real-time (about 4 fps). The processing includes video decoding and encoding, stereo generation and display. According to our calculation, the average processing speed on pure stereo creation is 4.2 times real-time (about 6 fps). After algorithm optimization, we expect to improve the processing speed significantly so that it can run in near real time.

3.1 Experimental Results on Side-Looking Video Clips

Table 1. Statistic results of side-looking video clips

File name	Length (hr:min:sec)	Frame rate	Resolution	Entire processing time including decoding and encoding time	Processing time for stereo
SideLooking-1.avi	00:02:17	30	640x480	00:14:50	00:09:50
Movie1_1.avi	00:01:50	25	720x480	00:12:18	00:07:58
Movie2_1.avi	00:03:01	25	720x480	00:20:31	00:13:32



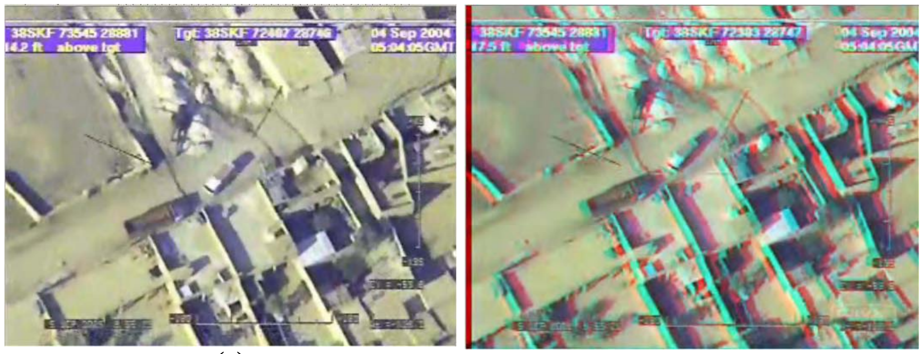
(a) (b)

Fig. 5. Example #1 Side-looking ((a) 2D frame, (b) 3D frame in Red-cyan format)

3.2 Experimental Results on Down-Looking Video Clips

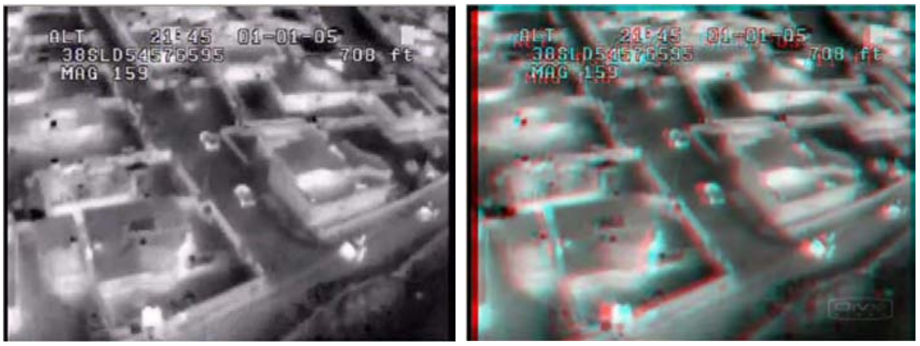
Table 2 Statistic results of down-looking video clips

File name	Length (hr:min:sec)	Frame rate	Resolution	Entire processing time including decoding and encoding time	Processing time for Stereo
Recon_1_all.avi	00:00:49	25.0	720x480	00:04:54	00:03:11
Side_1_all.avi	00:05:54	29.6	304x224	00:39:25	00:28:33
VTS_01_1.avi	00:04:32	29.4	720x480	00:52:54	00:40:32
VTS_01_2.avi	00:34:46	29.1	720x480	04:53:23	03:54:11
VTS_01_3.avi	00:05:50	29.8	720x480	00:03:42	00:02:43
1stmsnops.avi	00:00:25	23.5	320x240	00:02:31	00:01:36
Safe house.avi	00:03:34	23.7	320x240	00:23:12	00:15:18
Bda bldgs.avi	00:00:15	23.7	320x240	00:01:36	00:01:03
Kidnapping.avi	00:03:28	23.8	320x240	00:22:51	00:15:18
KiowaDown.avi	00:00:41	23.9	320x240	00:04:06	00:02:35
KufaCompressed.avi	00:00:24	23.7	640x480	00:02:36	00:01:43
Raven Footage.avi	00:00:48	22.2	640x480	00:05:07	00:03:25



(a) (b)

Fig. 6. Example #2 Down-looking ((a) 2D frame, (b) 3D frame in Red-cyan format)



(a) (b)

Fig. 7. Example #3 Down-Looking ((a) 2D frame, (b) 3D frame in Red-cyan format)

4 Conclusions

We have presented a systematic approach to stereo video formation. The approach consists of software-based stabilization, robust feature extraction using Harris' corner detection, accurate feature correspondence by a local-window based search, and reliable and robust video generation based on hierarchical image rectification. Key advantages of our approach include stereo video formation from a single moving uncalibrated camera and real-time processing capability. Extensive evaluations using real-life data collected from a single moving camera with unknown parameters and unknown relative position and orientation of the camera clearly demonstrated the efficiency of the approach. The future work of our research includes algorithm optimization and improvement on stereo video generation, and real-time implementation.

Acknowledgements

This work was supported by a grant from the U. S. Army Research Lab (ARL). The authors would like to thank Dr. Barry Vaughan for his helpful suggestions and comments.

References

1. Fusiello, A., Trucco, E., Verri, A.: A compact algorithm for rectification of stereo pairs. Journal of Machine Vision and Applications 12, 16–22 (2000)
2. Loop, C., Zhang, Z.: Computing rectifying homographies for stero vision. In: CVPR, pp. 125–131 (1999)
3. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision, 2nd edn. Cambridge University, Cambridge (2003)
4. Hartley, R.: Theory and practice of projective rectification. IJCV 35(2), 1–16 (1999)
5. Zhou, J., Li, B.: Image Rectification for Stereoscopic Visualization without 3D Glasses. In: Proc. Int. Conf. on Image & Video Retrieval (July 2006)
6. Li, X., Kwan, C., Li, B.: A generic approach on object matching and tracking. In: Campilho, A., Kamel, M. (eds.) ICIAR 2006. LNCS, vol. 4141, Springer, Heidelberg (2006)
7. Li, X., Kwan, C., Li, B.: Stereo imaging with uncalibrated camera. In: ISVC 2006 (2006)
8. Harris, C., Stephens, M.J.: A combined corner and edge detector. In: Alvey vision conference (1988)
9. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM 24, 381–395 (1981)

3D Shape Recovery by the Use of Single Image Plus Simple Pattern Illumination

Zhan Song and Ronald Chung[*]

Department of Mechanical & Automation Engineering
The Chinese University of Hong Kong, Hong Kong, China
{zsong,rchung}@mae.cuhk.edu.hk

Abstract. This paper presents a method of surface orientation and in turn shape recovery from a single image captured under projection of a simple checker-board pattern. The essences of the method include that only one image is required, that accurate correspondence establishment between the image and the projected pattern is not necessary, that the determination of 3D is much less sensitive to imaging noise and illumination condition than intensity-based methods like shape from shading. The method relies upon the fact that surface orientations at the grid points are only decided by image tangents in the image data. Experiments on planar, spherical, and ribbon-like surfaces show that, with accurate calibration of the projector-and-camera system through a mechanism we proposed earlier, 3D shape can be recovered with ease and precision both much better than before.

1 Introduction

Recovering 3D surface description of a scene or object has been one of the most important problems in computer vision. Many approaches have been proposed, such as stereo vision [1], structured light based approach [2], and laser strip scanning [3]. Correspondence problem is the key challenge in these approaches. To avoid tackling the correspondence problem, single image-based methods such as shape from texture [4], shape from shading [5], and shape from specularity [6] have also been suggested, which allow not absolute depth but relative depth information to be determined. Relative depth description has the scale ambiguity in comparison with the absolute depth description, but is a shape description sufficient for many applications including model registration, object recognition, pose estimation, and segmentation [7].

In this paper, we propose a method of recovering relative depth from a single image that is captured under projection of a simple pattern – the checker board pattern. The method does not require to establish correspondence between the image plane and the projector panel. Compared with methods like shape from shading and shape from specularity, the method does not depend upon the absolute values of image intensities and therefore is far less sensitive to imaging noise, illumination condition, and surface reflectance variance of the imaged scene. The method is based upon this principle. A known projected pattern, upon reflection from scene *S*, will have its image appearance

[*] Corresponding author.

modulated by the surface profile of *S*. With knowledge of the projected pattern which is under full control of the method, plus image observations of the reflected pattern, 3D information of the imaged scene *S* can be determined. In this paper, we shall show that surface orientation at any projected grid-point **P** on the target scene can be determined solely by the tangents to the imaged grid-lines that intersect to form the image position $\mathbf{p}_c$ of **P** on the image plane. Experimental results on a variety of surfaces – planar, spherical, ribbon-like ones – show that with accurate calibration of the projector-and-camera system using a system we proposed earlier [18], shape can be reconstructed with ease and precision far surpassing those of the previous work.

This paper is organized in the following way. In Section 2, previous work on surface orientation recovery is briefly reviewed. The principle of determining surface orientation from two image tangents is presented in Section 3. In Section 4, experimental results on a variety of surfaces are shown. Conclusions and future work are offered in Section 5.

2 Previous Work

Approaches for visual recovery of relative depth or surface orientation can be categorized into two classes. The first class makes use of the natural features on the target surface and assume certain ideal property (constant surface albedo, uniform textures etc.) of them. It is also named shape-from-X techniques, where X can be shading, texture, etc. The second class utilizes artificial textures or patterns that are projected onto the target surface. Since it does not depend upon the existence and observability of natural features, it makes less assumption on the target surface, and is generally more robust to imaging noise and illumination condition and variance in the reflectance property. Below we review in more details two reconstruction methods that are particularly related to this work.

Shape from shading is one representative method of the shape-from-X technique. The gradual variation of shading (i.e., the gray level intensity) in the image is utilized to recover the surface shape. The Lambertian reflectance model is usually assumed, in which the gray level at any position in the image is solely determined by the surface orientation and the incident angle of light there [9]. To a gray level image the method aims to recover the light source direction and the surface orientation in 3D at each image position. The shape from shading problem can be formulated as that of finding the solution of a nonlinear first-order partial differential equation called the brightness equation. It has been shown to be an ill-posed problem that generally has infinitely many solutions [10]. Additional heuristics on surface smoothness or intensity gradient are usually adopted to let surface orientations be determined in more precise form.

Structured light projection methods have also been proposed to recover local surface orientation. In the method proposed by Shrikhande and Stockman [11], a grid pattern is projected onto an object. Surface orientations at the grid-points (the intersections of the grid-lines) are induced from the change of the lengths of the grid edges in the image data. The result however indicates that there could be large errors at places where the grid-cell edges have large distortion, and such distortion is often induced by sharp edges in 3D. Woodham [12] proposed a photometric stereo method which determines surface orientation at each image position by changing the direction of light sources. Sugihara [13] proposed a method which infers surface orientation using the distortion of a known pattern projected on the surface from a structured light source. The projected textures on the object surface are detected and the distortion from the regular pattern is measured in order to estimate the surface orientation. Winkelbach and Wahl [14] used a uni-direction strip pattern to compute the surface normal at each edge point. A fringe pattern is projected onto the object twice from two different angles, each time with an image captured, so that two surface tangents are available for each image position. Surface normal at every image point can be interpolated from the data in the two images. A simplified method which determines surface normals from the slopes and intervals of the stripes in the image was also proposed [15] [16]. It is based on the assumption that the surface patch between two strip edges is planar or very smooth. Thus the tilt angle can be estimated from the deformed widths of the strip by comparing them with the strip width measured for a reference plane. In their system, pattern projection is assumed a parallel projection, and image capture a parallel imaging as well. In other words, the intrinsic parameters of both the camera and projector are not considered, and errors due to the much simplified projection and imaging models are inevitable.

Our method is related to the method proposed in [14], but we use checker-board pattern not fringe pattern. We also assume not parallel projection nor parallel imaging, but perspective projection and imaging. With more accurate models for imaging (camera) and for illumination (pattern projection), more accurate 3D determination can be attained. We show that by the use of a calibration method we proposed [18] for precise calibration of the projector-camera system, surface orientation and relative depth or shape can be recovered precisely.

3 A Method Requiring Only One Image Under Illumination of Simple Pattern

3.1 The Underlying Principle

Suppose we program a pattern on the display panel of the projector, which consists of a 2D array of alternate dark and bright rectangular blocks, and project it to the object surface. The edge of the blocks forms the grid-lines, and the intersections of every two grid-lines form the grid-points. The grid-lines and grid-points are accessible in both the projector panel and the image data.

Consider any grid-point $\mathbf{p}_p$ in the pattern panel of the projector, and the two accompanying grid-lines that compose it, as illustrated by Fig. 1. Suppose the grid-point and grid-lines induce an imaged grid-point $\mathbf{p}_c$ and two imaged grid-lines (or more correctly grid-curves, as the original grid-lines are generally modulated by the curvature of the surface in 3D and appear not as lines in the image data) on the image plane via a 3D point $\mathbf{P}$ on the object surface in space. Suppose the tangents to the grid-lines at point $\mathbf{p}_p$ in the pattern projection panel are $\mathbf{t}_{p1}$ and $\mathbf{t}_{p2}$ respectively, and the tangents to the grid-lines at point $\mathbf{p}_c$ in the image plane are $\mathbf{t}_{c1}$ and $\mathbf{t}_{c2}$.

Tangent $\mathbf{t}_{p1}$ and the grid-point $\mathbf{p}_p$ in the pattern projection panel together form a plane $\prod(\mathbf{p}_p, \mathbf{t}_{p1})$ of illumination from the light source, which is reflected by the object surface at point **P** and becomes the plane of projection $\Pi(\mathbf{p}_c, \mathbf{t}_{c1})$ to the image plane. The intersection of the two light planes $\Pi(\mathbf{p}_p, \mathbf{t}_{p1})$ and $\Pi(\mathbf{p}_c, \mathbf{t}_{c1})$ actually defines a tangent $\mathbf{t}_1$ in 3D to the object surface at point **P**. $\mathbf{p}_p$ and $\mathbf{t}_{p1}$ are fully accessible as they are entities under system design, and so are $\mathbf{p}_c$ and $\mathbf{t}_{c1}$ as they are entities observable from the image data. Thus the two light planes $\Pi(\mathbf{p}_p, \mathbf{t}_{p1})$ and $\Pi(\mathbf{p}_c, \mathbf{t}_{c1})$ are both constructible, and their intersection $\mathbf{t}_1$ can be determined. In fact the tangent $\mathbf{t}_1$ to the object surface at point **P** is merely the cross-product of the surface normals of the two light planes.

Similarly, another tangent $\mathbf{t}_2$ to the object surface at point **P** can be determined as the cross-product of the surface normals of two other light planes: $\Pi(\mathbf{p}_p, \mathbf{t}_{p2})$ and $\Pi(\mathbf{p}_c, \mathbf{t}_{c2})$, which are both accessible from design and image observations.

In other words, by simply taking one image of the object surface that is under projection of a proper pattern, for any imaged grid-point $\mathbf{p}_p$ at position (x,y) on the image plane, the surface orientation $\mathbf{n}(x,y)$ of the object surface at the associated 3D point can be determined simply as $\mathbf{n}(x,y)=\mathbf{t}_1\times\mathbf{t}_2$ from image observations $\{\mathbf{p}_c, \mathbf{t}_{c1}, \mathbf{t}_{c2}\}$ and pattern data $\{\mathbf{p}_p, \mathbf{t}_{p1}, \mathbf{t}_{p2}\}$. This is the underlying principle of this work.

3.2 The Path from Image Tangents to 3D Orientation

Here we elaborate the calculations more precisely. Suppose $\mathbf{n}_{c1}$ and $\mathbf{n}_{c2}$ are the surface normals of the light planes $\Pi(\mathbf{p}_c, \mathbf{t}_{c1})$ and $\Pi(\mathbf{p}_c, \mathbf{t}_{c2})$ on the camera side, and $\mathbf{n}_{p1}$ and $\mathbf{n}_{p2}$ the surface normals of the light planes $\Pi(\mathbf{p}_p, \mathbf{t}_{p1})$ and $\Pi(\mathbf{p}_p, \mathbf{t}_{p2})$ on the projector side. Suppose the intrinsic parameters of the camera and projector (which is modeled as another perspective camera, except that light is coming up out of it instead of going into it) have been calibrated, which are focal lengths f_c, f_p, and principal points $\boldsymbol{C}_c$ (u_{c0}, v_{c0}), $\boldsymbol{C}_p(u_{p0}, v_{p0})$. Then $\mathbf{n}_{c1}$, $\mathbf{n}_{c2}$, $\mathbf{n}_{p1}$, $\mathbf{n}_{p2}$ can be determined as:

$$\vec{\mathbf{n}}_{ci} = [(u_{ci} - u_{c0}) \quad (v_{ci} - v_{c0}) \quad -f_c] \times [\cos\theta_{ci} \quad \sin\theta_{ci} \quad 0] \quad i = 1,2 \tag{1}$$

$$\vec{\mathbf{n}}_{pi} = [(u_{pi} - u_{p0}) \quad (v_{pi} - v_{p0}) \quad -f_p] \times [\cos\theta_{pi} \quad \sin\theta_{pi} \quad 0] \quad i = 1,2 \tag{2}$$

where θ_* indicates the slant angle of the associated 2D tangent $\mathbf{t}_*$ with respect to the camera's image plane or projector's display panel as illustrated by Fig. 2a.

So the two 3D tangents $\mathbf{t}_1$, $\mathbf{t}_2$ to the object surface at point **P** that is associated with image point $\mathbf{p}_p=(x,y)$, if with reference to the camera coordinate system, can be obtained as:

$$\mathbf{t}_1 = \mathbf{n}_{c1} \times (\mathbf{R}\mathbf{n}_{p1}) \tag{3}$$

$$\mathbf{t}_2 = \mathbf{n}_{c2} \times (\mathbf{R}\mathbf{n}_{p2}) \tag{4}$$

where **R** represents the rotational relationship between the camera and projector coordinate frames.

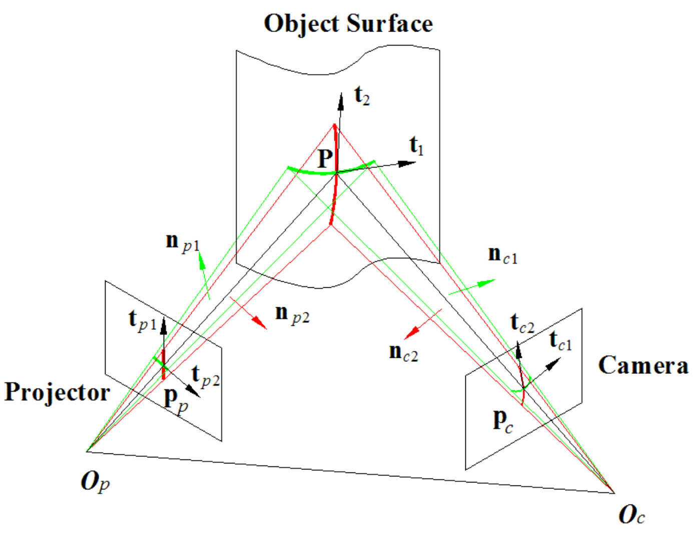
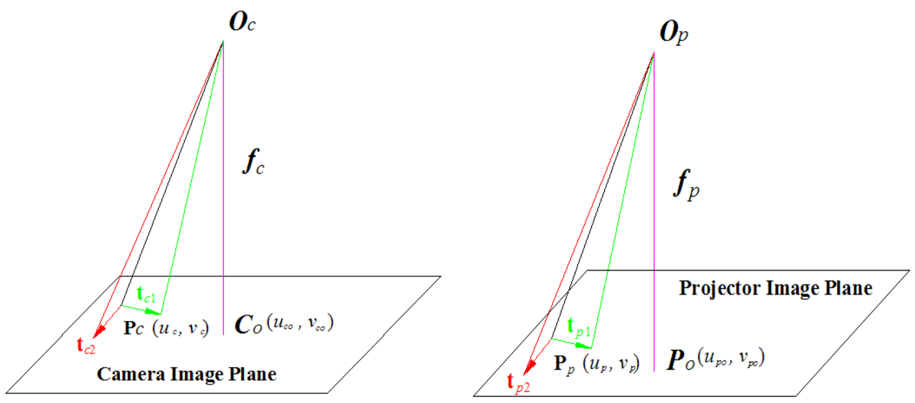


Fig. 1. Determining surface orientation from two image tangents, for object surface under pattern illumination



(a) (b)

Fig. 2. Definition of the slant angle of a 2D tangent as used in the calculation

Finally, the surface orientation to the object surface at point **P** that is associated with the image position (*x*,*y*) can be determined as:

$$\mathbf{n}(x, y) = \mathbf{t}_1 \times \mathbf{t}_2 \tag{5}$$

Notice that the determination of local surface orientation as expressed by Equation (5) is a deterministic process that requires only image information local to the specific point to operate. Unlike shading from shading and many other methods, it does not require assumption about how local surface orientations at neighboring points are related. More specifically, it requires no process of iterations to determine local surface orientations.

3.3 The Needlessness of Correspondences

The previous analysis indicates that surface orientation in 3D can indeed be determined from image tangents, but it also points out that for each imaged grid-point $\mathbf{p}_p=(x,y)$ and the accompanying image tangents $\mathbf{t}_{c1}$ and $\mathbf{t}_{c2}$, it requires also the 2D tangents $\mathbf{t}_{p1}$ and $\mathbf{t}_{p2}$ and other local information in the projector's display panel to go with them. In other words, correspondence between grid-points in the image and grid-points in the projector panel seems necessary.

However, we shall show that with the suitable projection pattern and suitable light source, it is possible that the above correspondence be avoided. Specifically, we choose to use the checker-board pattern for projection, so that the 2D tangents $\mathbf{t}_{p1}$ and $\mathbf{t}_{p2}$ to the grid-lines at each grid-point on the projector panel are all the same. Under the assumption of parallel projection, the invariant $\mathbf{t}_{p1}$ and $\mathbf{t}_{p2}$ would lead to the determination of surface normals $\mathbf{n}_{p1}$ and $\mathbf{n}_{p2}$ (normals to the light planes from the projector) which are invariant with the grid-points. With this, the necessary local entities on the projector side can all be made global, and a once-and-for-all pre-estimation step will make correspondence between the image plane and the projector panel unnecessary. In previous work such as [14][15][16], the parallel projection is assumed not only the projector but also the camera.

However, the projector illumination process may not be exactly captured by the parallel projection model, and discrepancy from it will cause bigger error to the determination of $\mathbf{n}_{p1}$ and $\mathbf{n}_{p2}$ at positions farther away from the center (i.e.., the principal point) of the projector panel. We use a simple correction mechanism to fix the problem. For each grid-point $\mathbf{p}_c$ $(=(u_c,v_c))$ in the image plane, we adopt a linear mapping, as described below, to estimate more finely the associated grid-point $\mathbf{p}_p$ $(=(u_p,v_p))$ in the projector panel:

$$[u_p, v_p] = [\frac{u_c - u_{\min}}{u_{\max} - u_{\min}} W_p, \frac{v_c - v_{\min}}{v_{\max} - v_{\min}} H_p] \tag{6}$$

where $(u_{\min}, v_{\min})$, $(u_{\max}, v_{\max})$ represent the top-left and bottom-right grid point positions in the image plane, W_p and H_p are the width and height of the pattern on the projector panel counted in projector panel's pixels. This way, a projection model closer to perspective projection than parallel projection model is used to determine $\mathbf{n}_{p1}$ and $\mathbf{n}_{p2}$ at the estimated grid-point $\mathbf{p}_p$, and this is illustrated by Fig. 2b.

3.4 Calibration of Projector-and-Camera System

While in previous work like [14][15][16] illumination and imaging are modeled as parallel projections, here they are modeled as perspective projections. With this, the proposed method can expectedly reach more precision in 3D reconstruction.

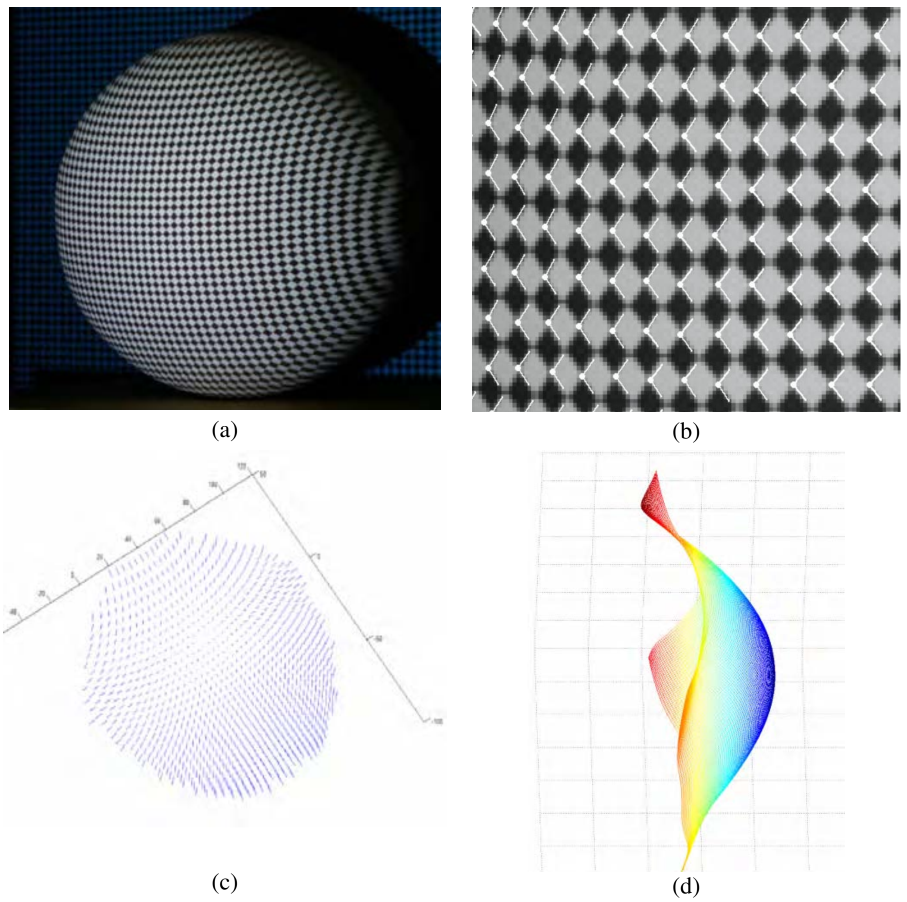


Fig. 3. Reconstruction of a spherical surface by the proposed method: (a) object surface under pattern illumination; (b) appearance of grid-points and grid-lines in the captured image; (c) reconstructed local surface orientations as viewed from one perspective; (d) interpolated surface as viewed from another perspective.

However, it is also obvious that the calibration of the intrinsic and extrinsic parameters of the projector-and-camera system has an important role to play. On this we use a calibration mechanism [18] that makes use of an LCD display panel (separate from the one of the projector) as the external reference object. The calibration mechanism has shown to be able to reach very high calibration accuracy, allowing more precise 3D reconstruction to take place.

3.5 Feature Extraction

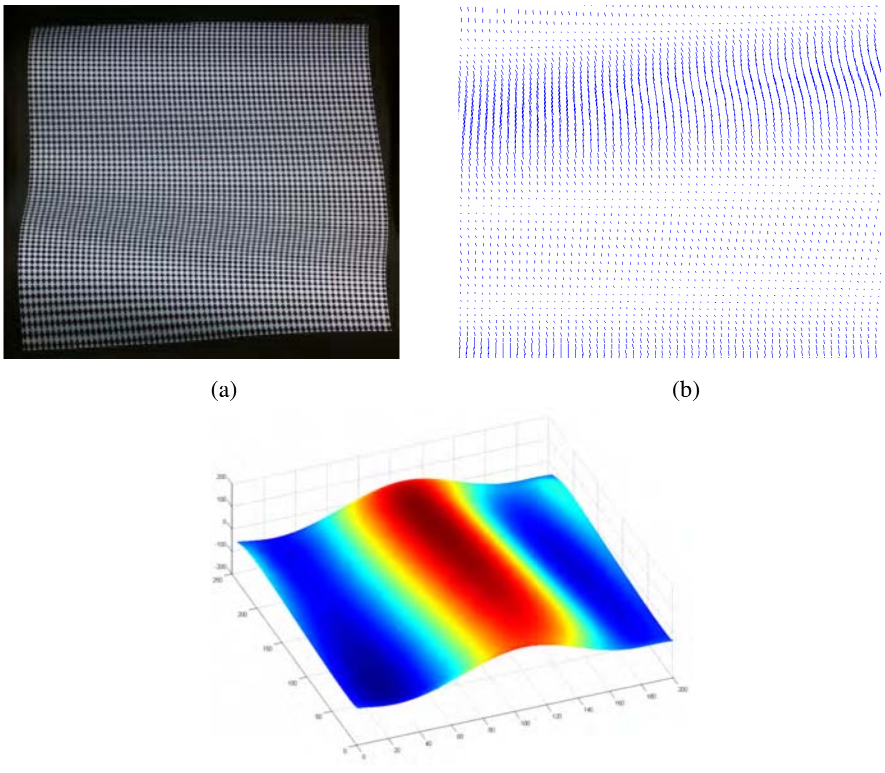
The reconstruction accuracy also depends upon how accurately the grid-lines and grid-points are located in the image, and how precisely 2D tangents to the grid-lines at the grid-points are extracted. There has been a rich body of works in the literature on the topic. In this work we use an algorithm that is of sub-pixel precision, whose description is beyond the page limit of this report and will be reported elsewhere.

3.6 Surface Interpolation

Once the local surface orientations at the grid-points are computed, a simple median filter is used to remove the outliers among them which mostly exist around the image boundary. Then the discrete data go through linear interpolation to form dense surface description, which embeds a regularization term related to smoothness. More precisely, we use the 2D integration method proposed by Frankot and Chellappa [17].

4 Experimental Results

A structured light system consisting of a DLP projector and digital camera was used for experiments. The system was first calibrated using the method described in [18] that made use of an LCD panel as the external reference object.


(a) (b)

(c)

Fig. 4. Reconstruction of a ribbon-like surface by the proposed method: (a) object surface under pattern illumination; (b) appearance of grid-points and grid-lines in the captured image; (c) reconstructed and interpolated surface as viewed from one perspective.

We used the LCD panel whose planarity was of industrial grade and which could be regarded as a perfectly planar surface to evaluate the quality of shape reconstruction. Shape reconstruction from the proposed method showed only very small deviation from planarity: of all the measured points, only an average discrepancy of 0.82° and standard deviation of 0.15° in the distribution of local surface orientations. Note that in methods like [14][15][16] and other earlier works a discrepancy in the range of 2° to 8° were reported.

We have also conducted experiment on a spherical object. The shape reconstruction result by the proposed method is as shown in Fig. 3: Fig. 3(a) shows the object under illumination of the checker-board pattern, Fig. 3(b) shows the appearance of grid-points and grid-lines in the image data, and Fig. 3(c) and (d) show the determined local surface orientations and the final interpolated shape as viewed from two viewpoints. The mean error of shape reconstruction was only 1.39°.

A ribbon-like surface was reconstructed and the result is shown in Fig. 4. Though ground truth of this surface was not available for evaluating in precise terms the reconstruction quality, visual check showed that the reconstruction was reasonable.

5 Conclusions and Future Work

Relative depth or shape description independent of scale is an important shape expression sufficient for many applications. We have described a method for its reconstruction that requires only off-the-shelf instruments which are not many more than a camera and a projector. The operation is also simple: requiring the projection of a checker-board pattern by the projector and just one image capture by the camera. The working mechanism is straightforward as well; it determines local surface orientations deterministically without the need of going through iterations, and it requires no feature correspondence between the projector panel and the image data either if relative depth is all that is desired, though feature correspondences are not excluded from the operation of the method and their presence could actually escalate the relative depth to absolute depth. Experiments show that with accurate calibration of the projector-and-camera system through system like [18], reconstruction could be of very promising precision.

As the method does not make use of raw intensities in the image data, it is more robust to imaging noise and illumination variations than intensity-based methods like shape from shading and other photometric methods, and it requires to make less assumption on the reflectance property of the object surface. Compared to other methods that are structured light based like [14][15][16], it is based upon more general models for projection and imaging, and it requires no scanning. Future work includes how additional images could boost the reconstruction precision, and how look-up table for all possible slant angles θ_{c*} can speed up the computations further.

References

1. Taylor, C.J.: Surface Reconstruction from Feature Based Stereo. In: 9th IEEE International Conference on Computer Vision, pp. 184–190 (2003)
2. Salvi, J., Pages, J., Batlle, J.: Pattern Codification Strategies in Structured Light Systems. Pattern Recognition 37(4), 827–849 (2004)

3. Kawasaki, H., Furukawa, R.: 3D Acquisition System using Uncalibrated Line-Laser Projector. In: 18th International Conference on Pattern Recognition, vol. 1, pp. 975–979 (2006)
4. Garding, J.: Direct Estimation of Shape from Texture. IEEE Trans. on PAMI 15, 1202–1208 (1993)
5. Zhang, R., Tsai, P., Cryer, J.E., Shah, M.: Shape from Shading: A Survey. IEEE Trans on. PAMI 21(8), 690–705 (1999)
6. Healey, G., Binford, T.O.: Local Shape from Specularity. In: 1st ICCV, pp. 151–160 (1987)
7. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Three-Dimensional Face Recognition. Computer Vision 64(1), 5–30 (2005)
8. Brooks, M.J., Chojnacki, W.: Direct Computation of Shape from Shading. In: 12th International Conference on Pattern Recognition, pp. 114–119 (1994)
9. Lange, H.: Advances in the Cooperation of Shape from Shading and Stereo Vision. Second International Conference on 3-D Imaging and Modeling, 46–58 (1999)
10. Woodham, R.J.: Photometric Stereo: A Reflectance Map Technique for Determining Surface Orientation from Image Intensity. In: 22nd Int. Symp. SPIE, vol. 155, pp. 136–143 (1978)
11. Shrikhande, N., Stockman, G.: Surface Orientation from a Projected Grid. IEEE Trans. on PAMI 2(6) (1989)
12. Horn, B.K.P., Brooks, M.J.: The Variational Approach to Shape from Shading. Computer Vision Graphics Image Processing 33, 174–208 (1986)
13. Sugihara, K., Okazaki, K., Kaihua, F., Sugie, N.: Regular Pattern Projection for Surface Measurement. In: 2nd International Symp. Robotics Research, pp. 17–24 (1984)
14. Winkelbach, S., Wahl, F.M.: Shape from 2D Edge Gradients. In: Radig, B., Florczyk, S. (eds.) Pattern Recognition. LNCS, vol. 2191, pp. 377–384. Springer, Heidelberg (2001)
15. Winkelbach, S., Wahl, F.M.: Shape from Single Stripe Pattern Illumination. In: Van Gool, L. (ed.) Pattern Recognition. LNCS, vol. 2449, pp. 240–247. Springer, Heidelberg (2002)
16. Asada, M., Ichikawa, H., Tsuji, S.: Determining Surface Orientation by Projecting a Stripe Pattern. IEEE Trans. on PAMI 10(5), 749–754 (1988)
17. Robert, T., Frankot, R.: A Method for Enforcing Integrability in Shape from Shading Algorithms. IEEE Trans. on PAMI 10(4) (1988)
18. Song, Z., Chung, R., Yeung, S.W.-L.: An Accurate Camera-Projector System Calibration by the use of an LCD Panel. In: Proceedings of the 2007 International Conference on Image Processing, Computer Vision, and Pattern Recognition, pp. 164–170 (2007)

Reliable Depth Map Regeneration Via a Novel Omnidirectional Stereo Sensor*

Lei He[1,2], Chuanjiang Luo[1,2], Yanfeng Geng[1,2], Feng Zhu[2], and Yingming Hao[2]

[1] Graduate School of Chinese Academy of Sciences, Beijing, China
[2] Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China
{leih,cjluo,yfgeng,fzhu,ymhao}@sia.cn

Abstract. We present a method to obtain dense 3D maps for a mobile robot that is equipped with a novel omnidirectional stereo vision sensor. The vision sensor is composed of a perspective camera and two hyperbolic mirrors. Once the system has been calibrated and two image points respectively projected by upper and nether mirrors are matched, the 3D coordinate of the space point can be acquired by means of triangulation. To satisfy the reliability requirement by mobile robot navigation, we use high-quality stereo matching algorithm – the graph cut method. An initial depth map can be calculated using efficient dynamic programming technique. With a relatively good initial map, the process of graph cut converges quickly. We also show the necessary modification to handle panoramic images, including deformed matching template, adaptable template scale. Experiment shows that this proposed vision system is feasible as a practical stereo sensor for accurate 3D map generation.

1 Introduction

A catadioptric vision system using mirrors has been a popular means to get panoramic images [1], which contains a full horizontal field of view. This wide view is ideal for three-dimensional vision tasks such as motion estimation, localization, obstacle detection and mobile robots navigation. Omnidirectional stereo is a suitable sensing method for such tasks because it can acquire images and ranges of surrounding areas simultaneously.

Mobile robot navigation using binocular omnidirectional stereo vision has been reported in [2], [3]. Such two-camera stereo systems are costly and complicated compared to single camera stereo systems. Omnidirectional stereo based on a double-lobed mirror and a single camera was developed in [4]-[6]. A double lobed mirror is a coaxial mirror pair, where the centers of both mirrors are collinear with the camera axis, and the mirrors have a profile radially symmetric around this axis. This arrangement has the merit to produce two panoramic views of the scene in a single image. But the disadvantage of this method is the relatively small baseline it provides. Since the two mirrors are so close together, the effective baseline for stereo calculation is quite small.

* This work is supported by National Science Foundation of P.R. China under granted number 60575024.

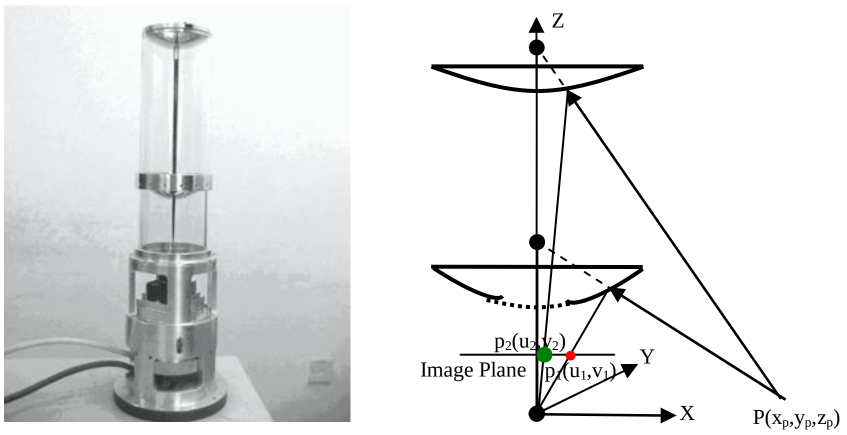


Fig. 1. The appearance of the stereo vision system and configuration of the system

We have developed a novel omnidirectional stereo vision optical device (OSVOD) based on a common perspective camera coupled with two hyperbolic mirrors, which are separately fixed inside a glass cylinder. As the separation between the two mirrors provides much enlarged baseline, the precision of the system has improved correspondingly (Fig. 1). The coaxial configuration of the camera and the two hyperbolic mirrors makes the epipolar line radially collinear, which makes the system free of the search process for complex epipolar curve in stereo matching (Fig. 3). Since the OSVOD is designed for corresponding mobile robot, it can be placed at a height of approximately 0.75 meters above the ground plane.

We will now give a brief overview of our work. After discussing related work, a full model of calibrating the system is presented in Section 3. In Section 4, we show how dense depth maps are extracted from the omnidirectional images. In Section 5 we conclude the paper.

2 Previous Work

State of the art algorithms for dense stereo matching can be divided into two categories:

Local method: These algorithms calculate some kind of similarity measure over an area [7]. They work well in relatively textured areas in a very fast speed, while they cannot gain correct disparity map in textureless areas and areas with repetitive textures, which is a unavoidable problem in most situations. In [8] a method of finding the largest unambiguous component has been proposed, but the density of the disparity map varies greatly depend on the discriminability of the similarity measure in a given situation.

Global method: These methods make explicit smoothness assumptions and try to find a global optimized solution of a predefined energy function that take into account both the matching similarities and smoothness assumptions. The energy function is always in the form of $E(d) = E_{data}(d)+\lambda * E_{smooth}(d)$, where λ is a parameter controlling the proportion of smoothness and image data. Most recent algorithms belong to this category [9], [10]. The biggest problem of global method is that the data term and the smoothness term represent two processes competing against each other, resulting in incorrect matches in areas of weak texture and areas where prior model is violated.

Although numerous methods exist for stereo matching, as to our knowledge, there are few algorithms specifically designed for single camera omnidirectional stereo. In our work, we try to adapt the promisingly performed graph cut algorithm to the peculiarities of omnidirectional images.

3 Calibrating the System

Camera calibration is a vital process for many computer vision jobs. In using the omnidirectional stereo vision system, its calibration is also important, as in the case of conventional stereo systems. Compared with the present calibrating techniques only concerning the camera intrinsic parameters, our calibrating algorithm presents a full model of the imaging process, which includes the rotation and translation between the camera and the mirror, and an algorithm to determine this relative position from observations of known points in a single image. We divide the calibration into two steps. Firstly we calibrate the camera's intrinsic parameters without the mirrors in order to reduce computational complexity. Once the camera intrinsic parameters are known, we estimate the pose parameters of the CCD camera with respect to the mirrors by LM algorithm. The function to be optimized is the sum of squared difference between coordinates of targets and the locations calculated from the targets' image points from the projection model. More details of the calibration process can be found in [11].

4 Stereo Matching

The images acquired by our OSVOD (Fig. 2) have some particularities in contrast to normal stereo pairs as follows, which may lead to poor result using traditional stereo

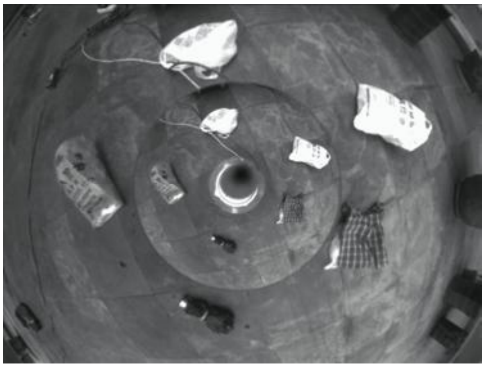


Fig. 2. Real indoor scene captured for depth map reconstruction

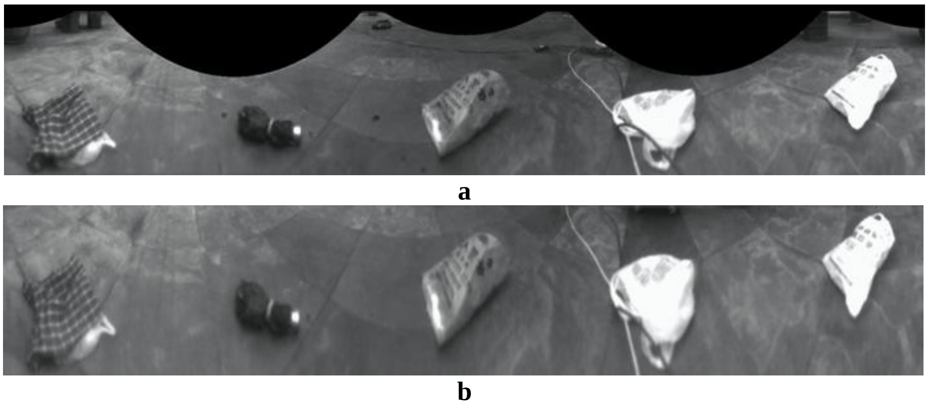


Fig. 3. Unwrapped cylindrical images, of which **a** corresponding to outer circle image and **b** inner circle image

matching methods: (1) The upper mirror and nether mirror have different focal length that the camera focal length has to compromise with the two, thus causing defocusing effect. As a result, similarity measures take on much less discriminability. (2) The resolution gets lower when moving away from the image center. The result is the farther off the center, the more unreliable the matching result is. (3) In this close-quarter imaging, the object surface is always not frontal-parallel to the virtual retina of the camera, resulting in large foreshortening effect between the outer circle image and the inner circle image.

To solve these problems, our method consists of the following steps: we first convert the raw image both to two cylindrical images and planform images corresponding to images via nether and upper mirrors respectively (Fig. 3 and Fig. 4). The vertical lines with the same abscissa in the cylindrical images are the same epipolar. We compute a similarity measurement for each disparity on every epipolar curve in the cylindrical images. The similarity measurement of a pixel pair is set as the maximum of that computed from cylindrical images and that from planform images. This is to solve problem (3), as surface perpendicular to the ground tend to have good similarity measurement on the cylindrical images and surface parallel to the ground on the planform images. Second, we propose a three-step method based on feature matching and dynamic programming to obtain an initial depth map. This method allows matching distinctive feature points first and breaks down the matching task into smaller and separate sub-problems so it could solve problem (1). Also, the relatively good initial map will prevent the following graph cut method from converging at local maximum point caused by problem (1). Finally, we use graph cut algorithm to be described in more detail later. Different weights are set to pixels with different radius to the image center. This is to solve problem (2). We also make necessary modification in the iteration of graph cut to handle panoramic images, including deformed matching template, adaptable template scale.

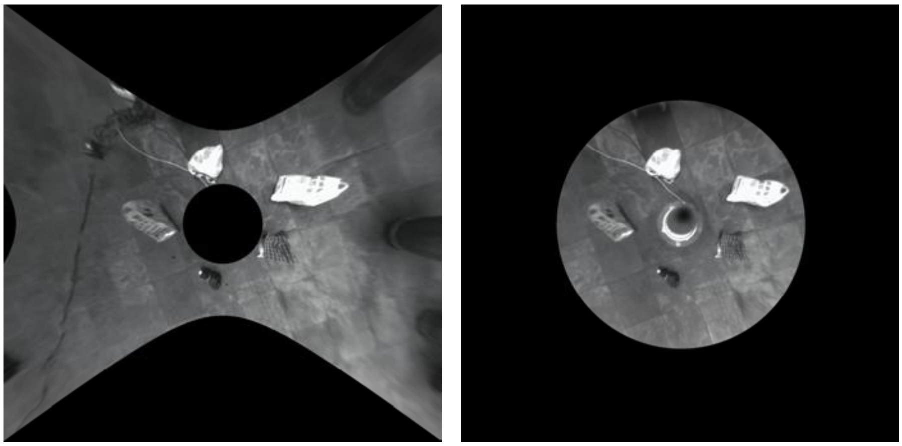


Fig. 4. Converted planform images, of which **a** corresponding to outer circle image and **b** inner circle image

4.1 MZNCC

The similarity measure we choose here is zero-mean normalized cross correlation (ZNCC), since it is invariant to intensity and contrast between two images. But directly using this measure would result in low discriminability. Chances exist that two templates with great difference in average gray-level or standard deviation which cannot be deemed as matched pair may have high ZNCC value. To avoid this possibility, we modified ZNCC (called MZNCC) by multiplying a window function as follows:

$$MZNCC(p,d) = \frac{\sum (I_a(i,j+d)-\mu_a)\cdot(I_b(i,j)-\mu_b)}{\sigma_a \cdot \sigma_b} \cdot w(|\mu_a - \mu_b|)\cdot w(\frac{\max(\sigma_a,\sigma_b)}{\min(\sigma_a,\sigma_b)} - 1) \tag{1}$$

where $w(x) = \begin{cases} 1, & x < \lambda \\ \lambda / x, & x \geq \lambda \end{cases}$, μ_a and μ_b are the average grey-level of matching window, σ_a and σ_b are the standard deviation, *d* denotes the disparity of pixel *p*.

We define our texture level of each point following the notion of bandwidth of the bandpass filter. For a given pixel and a given template centred in the pixel, we slide the template one pixel at a time in the two opposite directions along the epipolar line and stop at the location the MZNCC value of the shifted template with the primary one decrease below a certain threshold for the first time. Let *l* be the distance between the two stop points, which is inverse proportional the texture level. The definition of texture intensity can be formalized as:

$$T(u,v) = \sum_{-r\leq(i,j)\leq r} (I(u+i,v+j)-\bar{I})^2 \Big/ l^2 \tag{2}$$

where *r* is the radius of the template. With the use of this defined texture intensity and two thresholds, the whole image can be divided into three regions: strong textured, weak textured and textureless regions.

4.2 Initial Depth Map

Step1. Reliable FX-dominant Matching

This step follows the notion of FX-dominant defined by Sara [8]. The key of this notion is the uniqueness constraint which means each point may be matched with at most one point in the other image, and the ordering constraint which states the order of the matched points in the two epipolar line is the same. The latter one is not always true, but it is reasonable for most cases, especially indoor scene. The FX-region of a certain matched pair (*i*, *j*) in the MZNCC matrix is defined as the set of pairs that cannot coexist with (*i*, *j*) without violating these two constraints:

$$FX(p) = \{q = (k,l) \mid (k \geq i \wedge l \leq j) \vee (k \leq i \wedge l \geq j) \wedge q \neq p\} \tag{3}$$

It is formed by two opposite quadrants around (i, j) in the MZNCC matrix. And FX-dominant matching is to find pairs that have higher value than any pair in the FX-region. However, due to noise and distortion, the selected FX-dominant pairs still can not ensure its reliability. We only choose pairs from the FX-dominant results which satisfy the condition that the difference of the MZNCC value of the pair and the second local maximum MZNCC of $FX(p) \cup p$ is higher than a threshold (we choose 0.15).

Step2. Feature Matching and Ambiguous Removal

In this step, firstly we plot the curve of the texture intensity for a given epipolar line and choose all local maximum as feature points. For every feature point every matching pair with local maximum MZNCC higher than 0.7 is labeled as a candidate match (Fig. 5 all labeled points). Then we select a combination of candidate matching pairs that obey uniqueness constraint and ordering constraint and have the highest sum of

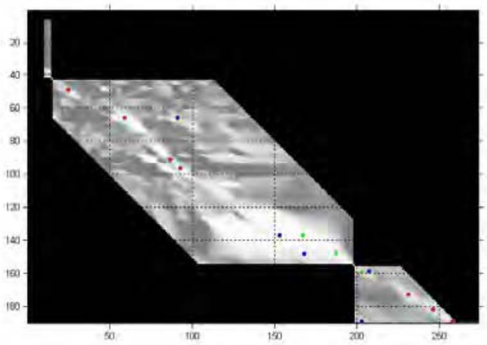


Fig. 5. Result of feature matching of a certain epipolar. The image represents the MZNCC matrix, all points labeled in the graph mean candidate match for detected features, red and green are the results chosen by maximization of sum of MZNCC and then green are removed for uneliminated ambiguous.

MZNCC (A feature point can be left unmatched with a zero contribution to the sum of MZNCC). The selected combination of illustrating epipolar shown in Fig. 5 is labeled red and green. In this selected combination, still some ambiguous match candidates exist. We mean a selected candidate is unambiguous if it is the only choice without altering other matched feature points under uniqueness and ordering constraint, otherwise it is ambiguous. We will then remove all ambiguous feature points until no matched feature is ambiguous. In Fig. 5, the ambiguous match candidates are labeled green and they are to be removed from the feature matching result.

Table 1. The penalty item

	Strong textured	Weak textured	Textureless
A	$-\lambda \bullet \mu \bullet \max((0.7 - MZNCC), 0)$	$-\mu \bullet \max((0.7 - MZNCC), 0)$	0
B	$-\sigma \bullet \lambda \bullet \mu \bullet \max((0.5 - MZNCC), 0)$	$-\sigma \bullet \mu \bullet \max((0.5 - MZNCC), 0)$	-MZNCC
C	-MZNCC	-MZNCC	-MZNCC

Step3. Dense Matching via DP

The remaining correspondences can be determined by dynamic programming. A starting and an ending point should be known at first. The matched feature points in the last step can naturally perform this role. Therefore, dynamic programming can be applied to every range between adjacent matched feature points. This objective of DP is achieved by finding a path with optimized energy function in a search space defined by the search range. The path is also restricted by the starting and ending point as well as the uniqueness and ordering constraint. The most important part is the definition of the energy function. Unlike others straightforwardly use sum of intensity difference, we define our energy function in he form of sum of MZNCC value plus a penalty item aim to assign different weights to different points based on the texture level and matching confidence. We also add a smooth item:

$$E = \sum MZNCC(i, j) + \sum penalty(i, j) + \sum E_{smooth}(i, j) \quad (4)$$

where (i, j) is in the matching route. We represent E_{smooth} as second order derivative of the depth curve, which satisfy the Markov property requirement and is appropriate for dynamic programming. To define the penalty item, we make another classification of all points. A point is belong to Class A (high confidence) if the global maximum MZNCC value is higher than 0.7, Class B (low confidence) if the global maximum MZNCC is between 0.5 and 0.7, otherwise Class C (noise). Then the penalty item is defined as Table 1, where λ is the strong texture weight, μ is penalty level and σ low confidence weight.

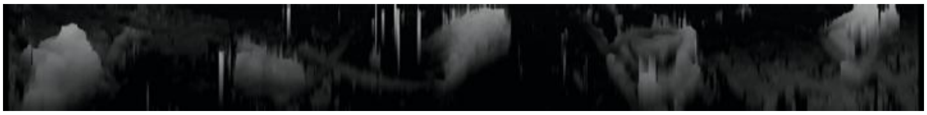


Fig. 6. Initial depth map via DP method (Fig. 3a is the reference map)

Fig. 6 shows the result via this three-step method. This depth map measures the height above the floor. The brightness of the map is proportional to the height, while black represents unknown areas. The result is much better than simple winner takes all solution. But it ignores the consistency constraint between adjacent epipolar lines, resulting in jagged contours and errors at weak textured and textureless areas. This can be improved by the following graph cut algorithm. Also, this initial depth map makes the graph cut process converge very quickly.

4.3 Graph Cuts

The graph cut method used here is a modification of the work by Kolmogorov & Zabih [10] to handle our specific omnidirectional image. Kolmogorov & Zabih choose an energy function composed of four items, E_{data}, E_{occ}, E_{smooth} and E_{unique}, thus converted the correspondence problem into an iterative energy minimization problem done by graph cut algorithm base on swap move algorithm, until convergence is reached. We here will introduce another function item E_{order} to our implementation. The ordering constraint is not always satisfied, but it is still a reasonable assumption in most cases. We use the appropriate energy function E in the following form to accommodate close-quarter measurement:

$$E(f) = E_{data}(f) + E_{occ}(f) + E_{smooth}(f) + E_{order}(f) \tag{5}$$

E_{data}(f) describes the MZNCC of corresponding pixels, and we also use the penalty item in Table 1 to distinguish pixels with different texture and reliability.

$$E_{data}(f) = \sum_{<p,q>\in A(f)} MZNCC(p,q) - penalty(p,q) \cdot \tag{6}$$

The item E_{occ}(f) adds a constant cost for each occluded pixel:

$$E_{occ}(f) = \sum_{p\in P} C_p T(|N_p(f)|=0) \, . \tag{7}$$

E_{order}(f) imposes an ordering constraint by adding an infinite penalty for pixels violating ordering constraint:

$$E_{order}(f) = \sum_{D(p)<D(q)} T(D[f(p)] > D[f(q)] \cdot \infty \tag{8}$$

where D is the Y-coordinate in the cylindrical image. $E_{smooth}(f)$ introduces a penalty for neighboring pixels having different disparity values:

$$E_{smooth}(f) = \sum_{(p,q,r)\in N} V_{pqr}(f_p, f_q, f_r) \tag{9}$$

where neighboring pixels *p*, *q* and *r* are series-wound orderly, E_{smooth} contains three variables so that it can represent the smoothness of *f* more appropriately than two-variable model does. V_{pqr} is a function that should be graph-representable. We give it as follows, C_{pqr} denotes the weight relationship between E_{smooth} and E_{data}

$$E_{smooth}(f) = \sum_{(p,q,r)\in N} C_{pqr} \cdot \min(T, g(\left|f_p + f_r - 2f_q\right| + \left|f_p - f_r\right|)) \tag{10}$$

Template Rectification and Adaptive Scale: For certain corresponding pixel pairs, it is expected that the MZNCC value of the two templates centered at these two points are very close to 1. This expectation is well satisfied when the two image templates are the projections of a single surfaces and this surface is frontal-parallel to the imaging plane of the virtual camera. When larger image templates straddle depth discontinuities, possibly including occluded regions, the MZNCC value may decrease to a value much smaller than 1. Also, if the surface is not parallel to the imaging plane, especially as the ground plane in our scene perpendicular to the imaging plane, the foreshortening effect makes the two templates differ quite significantly, also reduce the MZNCC value to an unsatisfactory amount.

In this iterative framework of graph cut, it is natural to estimate the appropriate template scale not to straddle depth discontinuities and rectify the template to compensate the foreshortening effect from current temporal result at each step. At each pixel in the image, we first use the largest template scale. We then compute the variance of the depth data in the template. If the variance exceeds an appropriately chosen threshold, it may be that the template scale is too large. Otherwise we continue to make use this template scale for MZNCC calculation.

After the scale is determined, it is ensured that the template corresponds to a single surface. We use the depth data to fit this surface to a plane in 3-D space, and then reproject this plane to the other image. Normally, the reprojected template is not a rectangle any more if the surface is not frontal-parallel. And we compute the MZNCC value between the rectangle template in the reference image and the rectified template in the other image. In this way, foreshortening effect is well compensated.

While initially these two operations may not be perfect, the graph cut algorithm is not very sensitive to noise. As long as graph cut process improves the result at each iteration, these two operations will become more and more accurate and will help improve the final depth map and speed up the convergence as well. Our empirical results bear this out.

Result. Fig. 7 shows the result of the final depth after graph cut process. It is easily seen that it improves a lot from the initial depth map by DP method. Although the ground truth map is unavailable, we randomly selected hundreds of points than can be measured and check the calculation error, finding that most (about 95%) are smaller than 20mm, only slightly higher than the calibration error. This is well satisfied the requirement of mobile robot navigation.

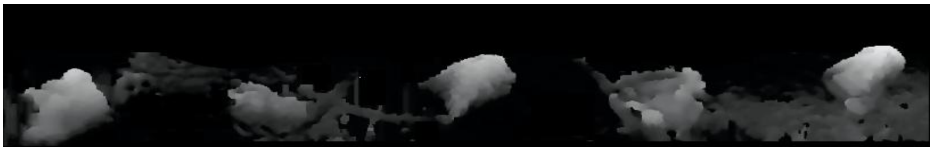


Fig. 7. Final depth map after graph cut (Fig. 3a is the reference map)

5 Conclusion

We have developed a complete framework of automatically generating omnidirectional depth maps around a mobile robot using a novel designed panoramic vision sensor. The configuration of the system makes the calculation simpler and stereo matching easier with good accuracy. We have presented a three-step method to estimate an initial depth map, which combines the advantage of feature matching and global matching. After that, a modified graph cut algorithm is applied to refine the result. This method basically solved the three major difficulties faced by our vision system and obtained convincing result.

References

1. Nayar, S.K.: Catadioptric Omnidirectional Camera. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 482–488. IEEE Computer Society Press, Los Alamitos (1997)
2. Gluckman, J., Nayar, S.K., Thoresz, K.J.: Real-time Omnidirectional and Panoramic Stereo. In: Proc. DARPA Image Understanding Workshop, pp. 299–303 (1998)
3. Koyasu, H., Miura, J., Shirai, Y.: Recognizing Moving Obstacles for Robot Navigation Using Real-time Omnidirectional Stereo Vision. Journal of Robotics and Mechatronics 14(2), 147–156 (2002)
4. Southwell, M.F.D., Basu, A., Reyda, J.: Panoramic Stereo. In: Proc. International Conf. on Pattern Recognition, pp. 378–382 (1996)
5. Conroy, T.L., Moore, J.B.: Resolution Invariant Surfaces for Panoramic Vision Systems. In: Proc. IEEE International Conf. on Computer Vision, pp. 392–397. IEEE Computer Society Press, Los Alamitos (1999)
6. Eduardo, L., Cabral José, L., de C. Jr., S., Marcos, C.H.: Omnidirectional Stereo Vision with a Hyperbolic Double Lobed Mirror. In: Proc. International Conf. on Pattern Recognition, pp. 1–4 (2004)
7. Devernay, F., Faugeras, O.: Computing Differential Properties of 3-D Shapes from Stereoscopic Images without 3-D Models. In: Proc. CVPR, pp. 208–213 (1994)
8. Sara, R.: Finding the Largest Unambiguous Component of Stereo Matching. In: Proc. 7th European Conf. on Computer Vision, vol. 2, pp. 900–914 (2002)
9. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Belief Propagation for Early Vision. International Journal of Computer Vision 70(1) (2006)
10. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusion using graph cuts. In: International Conference on Computer Vision (2001)
11. Luo, C., Su, L., Zhu, F., Shi, Z.: A Versatile Method for Omnidirectional Stereo Camera Calibration Based on BP Algorithm. In: ISNN 2006, pp. 383–389 (2006)

An Anti-aliasing Technique for Voxel-Based Massive Model Visualization Strategies

Gustavo N. Wagner, Alberto Raposo, and Marcelo Gattass

Tecgraf, Computer Science Dept., PUC-Rio, Rua Marques de São Vicente, 255, 22453-900 – Rio de Janeiro, Brazil
{gustavow,abraposo,mgattass}@tecgraf.puc-rio.br
http://www.tecgraf.puc-rio.br

Abstract. CAD models of industrial installations usually have hundreds of millions of triangles. For this reason they cannot be interactively rendered in the current generation of computer hardware. There are many different approaches to deal with this problem, including the Far Voxels algorithm, which uses a hierarchical level-of-detail structure. In this structure, voxels are used to create a coarse representation of the model when required. This strategy yields interactive rates for large data sets because it deals well with levels of detail, culling, occlusion and out-of-core model storage. The Far Voxels algorithm, however, has a severe alias problem when it is used to represent small or thin objects, which is especially visible during transitions between different levels of detail. This paper presents a new version of the Far Voxels algorithm that improves visual quality during model navigation.

1 Introduction

The visualization of CAD (Computer Aided Design) models is important for many engineering-related activities spanning from conception and design up to maintenance. There are, however, several problems in obtaining quality visualization of CAD models. One of these problems is the size of the visualization model, which can easily achieve hundreds of millions of triangles. As graphics engines become powerful enough to deal with large models, research on massive model visualization has received greater attention. Despite all recent advancements, most real CAD models still exceed the processing and memory capacity of existing hardware when interactive rates are required.

To speed up the rendering of this type of model, several algorithms have been proposed in the literature. Most of them are based on a combination of strategies that can be summarized as a combination of hierarchical levels of detail (HLOD), culling, occlusion, efficient use of the graphics pipeline and good management of disk, CPU and GPU memory. A promising algorithm to render massive models, called the Far Voxels, was presented in the paper by Gobbetti and Marton [3]. This algorithm uses a HLOD structure where intermediate (coarse) representations of sub-models are represented by voxels. This HLOD with voxel representation yields interactive rates for large data sets because it deals well with levels of detail, culling, occlusion and out-of-core model storage.

The Far Voxels algorithm, however, has a severe drawback when dealing with CAD models. CAD models usually have lines and thin objects that, even with the most detailed representation, introduce very high spatial frequency that cannot be rendered without proper anti-aliasing treatment. The aliasing caused is very disturbing during model navigation, where these thin objects seem to move between consecutive frames.

In the original Far Voxels algorithm the temporal aliasing problem is aggravated when these thin objects are converted to voxels. The voxels used to represent these objects tend to create representations which are larger than the ones of the original geometric representation, as shown in Fig. 1. In addition to creating a representation that is very different from the original model, this distortion causes very noticeable popping artifacts in the transition between different levels of detail.

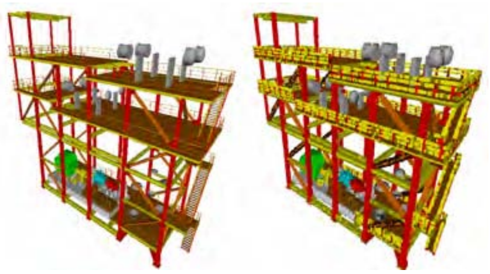


Fig. 1. Thin objects: (*left*) represented as geometry, (*right*) represented as voxels, causing visual artifacts

In this paper we propose a method for detecting this type of voxel and an alternative voxel representation that seeks to achieve images of the same quality obtained with current 3D hardware's anti-aliasing using the detailed representation of the model (triangles and lines). This detection and alternate voxel representation yields a new version of the Far Voxels algorithm.

2 Related Work

The literature on HLOD and point rendering is extensive and a complete review would be too long to be included here. For this reason, only key papers related to point rendering or improvement of its visual quality are discussed. Complete surveys on massive model rendering can be found in [3] and [10].

The idea of using points as graphics primitives is not new. In 1985, Levoy and Whitted [6] presented a seminal paper that proposes points as an efficient display primitive to render complex objects. In 1998, more than a decade later, Grossman and Dally [4] presented an efficient algorithm to render object from sampled points in the CPU. An advantage of representations based on points is that they do not require any effort to preserve the model’s topology, as it would have to be done in the frontier between meshes with different resolutions.

More recent approaches began to make use of 3D graphics hardware. In 2000, Rusinkiewicz and Levoy [9] presented the QSplat system, which uses point primitives to render very complex laser scanned models. In this system, the model is converted to a hierarchy of bounding-spheres. During visualization, the most appropriate spheres from the hierarchy are selected and rendered in 3D hardware as point primitives. In this hierarchy, points of a region of the model are grouped into a single point, which may be used instead of the cluster when viewed at distance.

In 2004 Gobbetti and Marton [2] presented an approach to organize point primitives in clusters. Their strategy reduces CPU processing, permits the clouds to be cached in graphics hardware and improves the performance of CPU-to-GPU communications. This idea evolved into the Far Voxels technique [3], which allows much larger models to be efficiently rendered with point-based primitives. The leaf nodes, which contain the most detailed representation of the model in the hierarchy, are still rendered as triangles.

Much effort has been placed in trying to improve the rendering quality of point primitives. In QSplat [9], different primitive shapes are tested to determine the one with the best quality/performance ratio. Alexa et al. [1], convert the model into a tree of higher order polynomial patches which is used to generate the point primitives in the proper resolution needed during visualization.

Zwicker et al. [11] presented a seminal paper about sampling issues in point rendering, based on the concept of Elliptical Weighted Average (EWA) resampling filters, which were introduced by Heckbert [5]. Ren, Pfister and Zwicker [8] proposed an implementation compatible with modern graphics hardware for the EWA screen filter [11], which permits high quality rendering of point-based 3D objects.

Here we approach the problem of visualizing large models using the Far Voxels algorithm, which is one of the most efficient techniques available for dealing with large CAD models. We present a solution that improves the appearance of a few specific voxels of the model, namely those representing objects whose dimensions are smaller than one voxel. Our approach to improve the rendering quality of point primitives is orthogonal to the abovementioned ones and could be used in conjunction with any one of them that uses voxels as impostors.

3 Opaque Voxels

The Far Voxels algorithm is composed of two main phases: preprocessing and model visualization. The preprocessing phase computes visibility information that will aid in the creation of the simplified voxel-based representations of parts of the model. The visualization phase uses this information to efficiently navigate through the model.

3.1 Preprocessing Phase

The preprocessing phase creates a representation of the model optimized for rendering by sampling the volume from all possible directions. This optimized representation is structured on an HLOD hierarchy, which has on its leaves the most detailed representation available for the model, and on the other nodes simplified representations of their respective regions. In this representation, nodes located near the hierarchy root will have the coarsest representations of the model, and are intended to be used to represent the model when it is viewed from far away.

The hierarchy is created by a recursive subdivision which uses axis-aligned planes positioned according to the surface-area heuristic [7]. The subdivision starts working with all triangles of the model, which are assigned to the hierarchy's root, and then starts subdividing them. Each subdivision will create a new pair of nodes on the tree, with existing triangles split between both nodes. The subdivision process ends when a node with less than a predefined amount of triangles is created. This node becomes a leaf of the hierarchy.

Simplified representations of parts of the model are created using a voxel representation. Inside a node, voxels are organized on a uniform grid, where each voxel will have, under ideal conditions, the size of one pixel when projected on the screen. Each voxel may assume different representations depending on the direction from where it is visualized, as it attempts to reproduce the appearance of the original model as closely as possible.

The simplified representations of the model's regions are created in two steps. First, all visible surfaces in that region are sampled with the use of a CPU-based raytracer. Then, the radiance of all the hits obtained from the intersection of the rays with the region's surfaces is analyzed and the appropriate shader to represent them is selected. The use of a raytracer permits selecting only the surfaces of the model that are visible, which is important to avoid that hidden surfaces create artifacts on the generated simplified representation.

The raytracer starts by defining a volume V, which corresponds to the region being sampled, and a surface S, from where the sampling rays will be shot. The distance d_{min} of surface S to the volume V is calculated as being equal to the minimum distance that the user has to be from V for this voxel representation to be used. As we can assure that the user will not be inside the region defined by surface S, all objects inside it can be used as occluders (Fig. 2a).

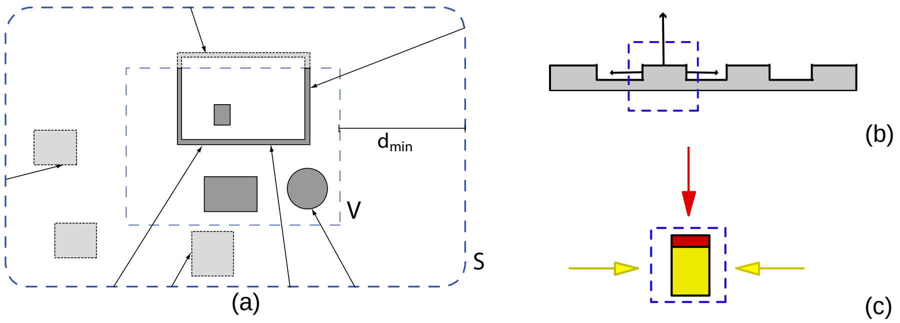


Fig. 2. (a) Example of a possible raytracing configuration. Voxel appearances may vary with viewing direction: (b) varying normals, (c) varying colors.

Many of the surfaces located inside features of the model are never hit by the sampling rays traced during the preprocessing phase and do not need to be represented, as it is assumed that this voxel representation is always viewed from outside. This is called Environmental Occlusion [3] and allows the creation of a more efficient simplified representation of the model. The amount of discarded voxels is reported to vary from 25% to 43% in the original paper and reached up to 57% in the models used here to test our implementation.

For all voxels that are hit by the sampling rays, the colors computed for all directions are stored and then analyzed to produce a radiance model for that point in space. This analysis results in a choice of the most appropriate shader model to render the voxel from the minimum distance established by the criterion that it should project in approximately one pixel. There are two types of shade models. The first one is intended to represent surfaces which are flat or almost flat, and is parameterized by a single normal and two materials, one for each direction from which the voxel may be seen. The second one is intended for more complex surfaces and is composed of 6 materials and 6 normals, each associated with one of the main viewing directions (±x, ±y, ±z). In Fig. 2b and 2c, there are examples where the normal and color of the surfaces are significantly different depending on the direction from where they are seen, and a single representation for the normal or color on the voxel would not generate satisfactory results. When the corresponding voxel is being viewed from an intermediate direction, the resulting material and normal used correspond to an interpolation of the material and normal associated with the nearest viewing directions weighted by their respective direction cosines.

The generated voxels are stored on disk to be loaded on demand during the model's visualization. The data file that is generated is divided in two parts. One part corresponds to the HLOD hierarchy, which contains all information necessary to traverse it and determine the appropriate configuration of nodes to be used. The other part contains the data that is used to represent the model, composed of lists of voxels for internal nodes of the hierarchy, and lists of triangles/triangle-strips for leaf nodes.

3.2 Visualization Phase

During visualization, all nodes of the model’s hierarchy are traversed. Each voxel node found on the hierarchy may be composed of voxels of the two existing types. For each type, the appropriate vertex shader is bound and all the voxels contained in it are drawn with a single call to glDrawArrays. The additional parameters necessary to create the voxel representation are transmitted using VertexAttributes.

Each node of the model’s hierarchy contains a possible simplified representation for its corresponding region of the model. While traversing these nodes, the viewer has to decide whether to use the representation available at that node or to continue traversing the node’s children in search of a more precise representation. This decision is based on a user-defined parameter that indicates the size that the voxel of a node must have when it is projected to the screen. The viewer keeps descending on the hierarchical structure until the existing voxels are smaller than the desired projected size or until a leaf node is found.

During the model traversal, branches of the hierarchy are tested for occlusion using Hardware Occlusion Queries. Parts of the model that are determined as being hidden do not need to be drawn.

The rendering algorithm may issue the Occlusion Query to determine if a certain branch of the graph is visible while rendering its bounding box or while rendering its full geometry. If only its bounding box is rendered and is found to be visible, its geometry will have to be rendered later. On the other hand, if the branch's full geometry is rendered, there is no advantage, at least for the current frame, in determining that it is hidden.

The visibility information obtained is used on the next frame when the renderer needs to decide how to query a node's visibility. Nodes that where hidden on the previous frame have their bounding-boxes checked for visibility before being rendered. These nodes have a good chance of remaining hidden in the current frame. The nodes that were visible on the last frame are always directly rendered as geometry or voxels.

Nodes on the scene hierarchy are visited by order of proximity to the camera, starting from the nearest ones. The results for the issued Occlusion Queries are not available instantly, so the renderer must not stop rendering while it waits for them. All performed queries are stored in a list of pending queries, which is checked by the visualization algorithm every time it needs to select a new node to be rendered. If the result of any pending query becomes available and indicates that its associated geometry is visible and has to be rendered, it is rendered immediately.

4 Detecting Problematic Voxels

When voxels are used to represent parts of the model composed of small or thin objects, the "pure" Far Voxels algorithm tends to create representations which appear "bloated" when compared to the original geometric representation. Unfortunately, thin and small objects are very common in CAD models and the first tests conducted with the original Far Voxels algorithm yielded unsatisfactory results. In addition to creating distortions in the simplified model appearance, they cause a very noticeable popping effect in the transition between two different levels of detail.

A naive solution would be breaking the voxel into smaller voxels, which would create a representation that could adjust to the model with the required precision. As in all other aliasing problems, increasing the resolution is not the best method to solve the problems arising from the presence of high spatial frequencies. Furthermore, this increase contradicts the whole idea of having voxels with the approximate size of one pixel in the screen.

A better approach results from using the pre-processing stage as a sampling stage. Voxels with small objects are problematic because they enclose high spatial frequencies that must be filtered before the rendering reconstruction occurs.

A possible implementation for filtering and rendering voxels with small objects follows the same ideas for anti-aliasing currently implemented in graphic boards. In the sampling stage they determine an opacity factor to be attributed to the voxel. The blending of a partially transparent voxel yields the blurring that results from filtering high frequencies. Note that to achieve this result it is not just a matter of enabling the 3D hardware's anti-aliasing procedures with opaque voxels. Semi-transparent voxels must be used.

In order to determine which voxels have to be represented as semi-transparent voxels, we use the visibility information generated by the raytracer during the preprocessing phase (Fig. 3a). Depending on the geometry configuration inside a voxel, the rays shot against it may hit some geometry inside it or pass through it.

When a voxel is trespassed by many rays in a certain direction, we may safely assume that the geometry contained in that voxel occupies a small cross-section area of the voxel when it is viewed from that angle. Thus, for that angle, this voxel may be rendered with a transparency scale proportional to the ratio between the amount of rays that hit a surface inside the voxel and the total number of rays that where shot against it.

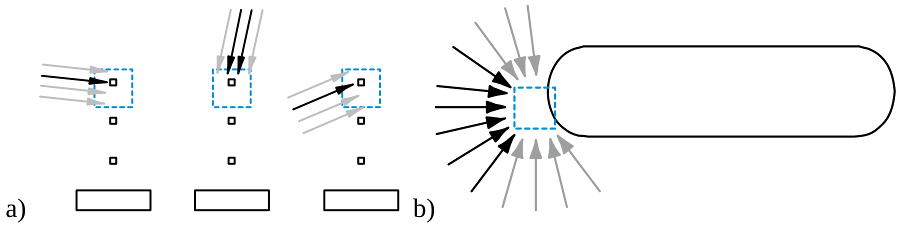


Fig. 3. a) Raytracer sampling a voxel to obtain its visibility. b) Voxels containing small parts of large objects may also have many rays passing through it.

We must be careful to use the transparency information in a different direction than the one in which it is computed. In very frequent cases, parts of large objects may occupy only a small fraction of the voxel (Fig. 3b) in a way it is hit by rays that come only from a few directions. This voxel may be rendered with transparency when viewed from the direction from where the rays that passed through it came from, but must remain opaque when viewed from all other directions. This procedure prevents the appearance of holes in large objects.

As storing visibility information for every possible viewing direction is unfeasible, the transparency information obtained from each sampling direction must be packed before it may be used in the visualization. Since the rays that crossed a certain voxel are not evenly distributed over all directions, we use a representation that allows independent transparencies for each of the six main viewing directions (±x, ±y, ±z). That is, we use a transparency model that is similar to the material color model explained in section 3.

The final voxel transparency will be an interpolation of the transparencies associated with the nearest viewing directions weighted by their respective direction cosines. If the transparency factor is the same for all directions, we may use a single transparency factor for the voxel, yielding a more efficient representation.

5 Rendering Anti-aliased Voxels

Correctly rendering any kind of transparent geometry using 3D hardware requires that it is drawn after the whole opaque geometry in the rendered scene. Also, transparent geometry has to be drawn from back to front in order to create a correct representation in cases where two or more transparent objects overlap on screen. This is exactly the opposite behavior expected for geometry being tested for occlusion with Occlusion Queries, which has to be rendered from front to back. For this reason, transparent voxels have to be handled in a different way.

As transparent voxels make up for only a small part of all voxels they may be rendered without being tested for occlusion after the whole opaque geometry, without significant impact on the overall performance. Also, most of the occluded transparent voxels are going to be discarded before being rasterized by the Early Z Culling present in current graphics cards, as most occluders will have already been rendered.

Ordering all voxels from back to front could cause performance problems with larger models, as we would have to manage these objects on a per-primitive level. As transparent voxels are well distributed in small groups throughout the model, a more relaxed approach can be used. During rendering, only the nodes of the graph that contains transparent voxels are ordered from back to front, while letting the primitives that exist inside these nodes be rendered in any order. As transparent voxels occur only in well distributed parts of the models, visual artifacts that arise from this simplification are hardly noticed, and can be further reduced if the transparent voxels are rendered with Z-Buffer writing disabled.

6 Results

The navigation application used to validate and test the developed technique was implemented in C++ using OpenGL. The models used to test the algorithm were real offshore structure models provided by Petrobras, a Brazilian Oil & Gas Company. In this paper, we present the results obtained using a single real engineering model, the P-50 FPSO (Floating Production Storage and Offloading), with 30 million triangles and 1.2 million objects. This model was chosen because it contains many regions where small or thin objects create visual artifacts when represented using voxels.

Performance tests were made over a typical walkthrough of the model. The machine used was an Athlon X2 64 4200+, with 4 GB of RAM memory, nVidia GeForce 8800 GTX graphics card with 768 MB of memory running on Windows XP Professional 32-bit.

The model was preprocessed on the same machine, but using Windows XP Professional 64-bit to allow the preprocessor to have access to all the available system memory. The entire model was preprocessed in 27 hours on a single machine.

Transparency information, which was added to all voxels of the model, only increased the total processed model size in 5%, from 1.56 GB to 1.64 GB. The rendering performance of the developed method was evaluated during a walkthrough that followed the path outlined in Fig. 4.

Frame rate information was recorded for each frame rendered. The data recorded compares the performance of the walkthrough over a model that uses the anti-aliased

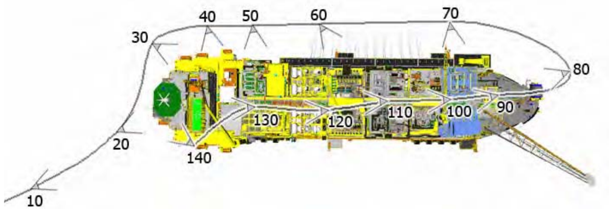


Fig. 4. Path followed on the walkthrough around the P-50 model

voxels developed with the performance obtained on a model that uses only the voxels presented in the original Far Voxels technique and occlusion queries. The frame rate obtained during the walkthrough over these two models is plotted in Fig. 5. Comparing the results obtained with these two different types of model allows verifying the efficiency of the developed anti-aliasing method.

The graph shows that the performance does not decrease significantly when the new types of voxels are used. This indicates that the developed voxel anti-aliasing

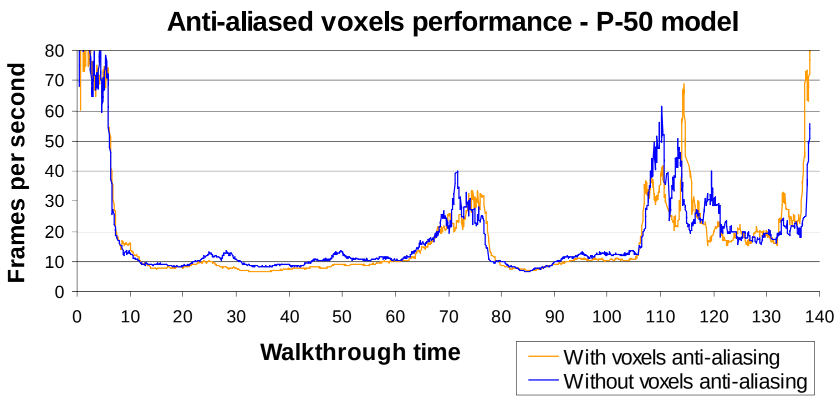
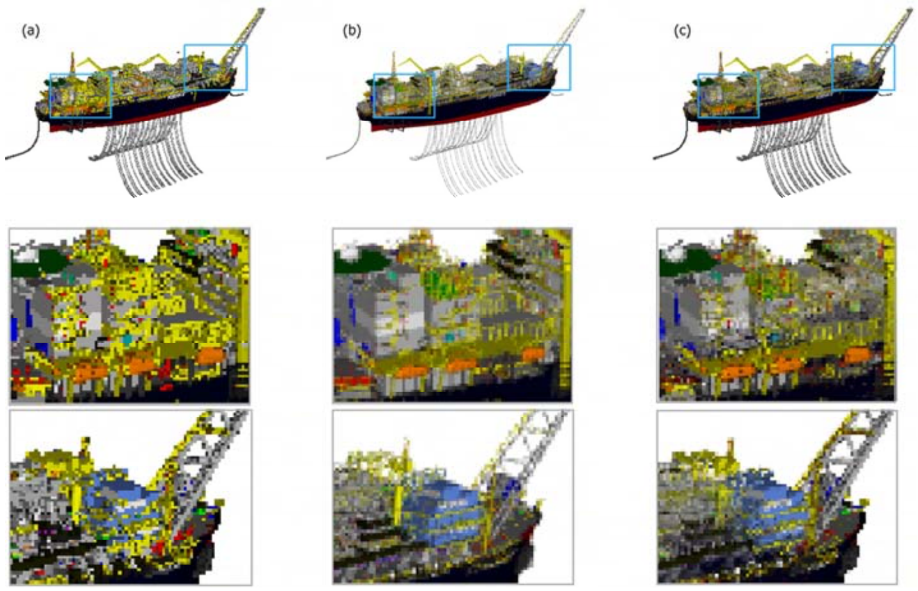


Fig. 5. Walkthrough performance with and without anti-aliased voxels



(a) voxels without the anti-aliasing technique (b) triangles only, no voxels (c) voxels with the anti-aliasing technique

Fig. 6. Results obtained using the developed technique compared with the ones obtained the original voxel strategy. The zoomed views are intended to be pixelated.

technique is efficient enough to be applied on larger models. Regarding visual quality improvement, Fig. 6 illustrates the difference between the voxels with and without anti-aliasing, comparing them to the ideal representation which uses only triangles.

7 Conclusions

This paper evaluated the use of the Far Voxels technique on models of offshore structures. Due to particular characteristics of these models, their visualization presented problems that were not treated in the original algorithm. To improve the visual representation of very small and thin objects, a detection method and an alternative voxel representation were implemented. This technique was implemented with a small simplification in the way transparent voxels are ordered, aiming at not increasing the CPU processing required to prepare the scene for rendering.

The results were evaluated using real oil & gas platform models, and the algorithm performance with the anti-aliasing filter was not significantly inferior to the performance obtained with our implementation of the original Far Voxels algorithm, which by itself is far more efficient than using simpler optimizations. This is especially true for very large models.

References

1. Alexa, M., Behr, J., Cohen-or, D., Fleishman, S., Levin, D., Silva, C.T.: Point set surfaces. In: Proceedings of the conference on Visualization 2001, pp. 21–28. IEEE Computer Society Press, Los Alamitos (2001)
2. Gobbetti, E., Marton, F.: Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. Computers & Graphics 28(6), 815–826 (2004)
3. Gobbetti, E., Marton, F.: Far voxels: a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. ACM Trans. Graph. 24(3), 878–885 (2005)
4. Grossman, J.P., Dally, W.J.: Point Sample Rendering. In: 9th Eurographics Workshop on Rendering, pp. 181–192 (1998)
5. Heckbert, P.: Fundamentals of texture mapping and image warping. M.sc. thesis, University of California, Berkeley (1989)
6. Levoy, M., Whitted, T.: The use of points as a display primitive, Tech. Rep. TR 85-022, University of North Carolina at Chapel Hill (1985)
7. MacDonald, J.D., Booth, K.S.: Heuristics for ray tracing using space subdivision. The Visual Computer 6(6), 153–165 (1990)
8. Ren, L., Pfister, H., Zwicker, M.: Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. Computer Graphics Forum 21(3), 461–470 (2002)
9. Rusinkiewicz, S., Levoy, M.: QSplat: a multiresolution point rendering system for large meshes. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 343–352. ACM Press/Addison-Wesley Publishing Co., New York (2000)
10. Yoon, S., Salomon, B., Gayle, R., Manocha, D.: Quick-VDR: Interactive View-Dependent Rendering of Massive Models. In: Proceedings of the IEEE Conference on Visualization 2004, pp. 131–138. IEEE Computer Society Press, Los Alamitos (2004)
11. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface Splatting. In: Computer Graphics, SIGGRAPH 2001 Proceedings, pp. 371–378 (2001)

Photo-Realistic Depth-of-Field Effects Synthesis Based on Real Camera Parameters

Huei-Yung Lin and Kai-Da Gu

Department of Electrical Engineering,
National Chung Cheng University,
168 University Rd., Min-Hsiung
Chia-Yi 621, Taiwan, R.O.C.
lin@ee.ccu.edu.tw, da0326@yahoo.com.tw

Abstract. Depth-of-field (DOF) is an important visual cue used for computer graphics and photography to illustrate the focus of attention. In this work, we present a method for photo-realistic DOF simulation based on the characteristics of a real camera system. Both the depth-blur relation for different camera focus settings and the nonlinear intensity response of image sensors are modeled. The camera parameters are calibrated and used for defocus blur synthesis. For a well-focused real scene image, the DOF effect is generated by spatial convolution with a distance dependent circular Gaussian mask. Experiments have shown that the difference between the images synthesized using the proposed method and the real images captured by a camera is almost indistinguishable.

1 Introduction

In 3D computer graphics and visualization applications, the choice of camera models plays an important role for the development of image synthesis algorithms. Most commonly used approaches adopt a perspective projection model to simulate a real camera system. Although the ray tracing techniques can be readily applied for real-time rendering, several important characteristics of practical imaging systems are not explicitly taken into account. For example, geometric and chromatic aberrations of the optics, the aperture shape and size of the camera system, and nonlinear intensity response of the CCD image sensors cannot be derived from a simple pinhole camera model. Furthermore, it is not possible to generate the defocus blur phenomenon present in the human vision and real optical systems, which is de facto an important visual cue used for depth perception.

For more realistic scene generation, it is clear that the imaging model with a real lens should be used for image synthesis [1]. One of the major successes in the past few decades is the simulation of depth-of-field (DOF) effect due to the finite aperture size of a real camera system. To create the DOF effect for a given 3D scene, different amount of defocus blur is generated for each image pixel based on the distance to the focused position in the scene. The existing techniques in the literature include distributed ray tracing [2,3], image blending using the accumulation buffer [4,5], blurring with layered depth [6], and post processing by image filtering [7,8], etc. Although some of the above approaches focused on accurate DOF simulation and others emphasized the capability of real-time rendering, all of them used ideal camera models for synthetic image generation. The defocus blur computations are not based on the actual DOF of practical optical systems. Thus, the realism of the rendered scenes is usually limited to the pure computer generated virtual environments.

To simulate the DOF effects for the applications with mixture of real scenes and synthetic objects, the visual inconsistency might become prominent in the rendered image due to different imaging models and camera parameters used for the real and virtual cameras. In addition to the global illumination issues commonly addressed in the augmented reality (AR) research field [9,10], a more realistic camera model should also be applied for photo-consistent image synthesis. More specifically, the synthetic image blur due to the optical defocus should be derived based on the parameter settings of a real camera system. Currently, most popular approaches use the so-called "circle of confusion" (CoC) introduced by out-of-focus of a thin lens camera model to calculate the defocus blur [11,12]. However, the simulation results are still approximations from an ideal image formation model with oversimplified parameter settings.

In this paper, we present a photo-realistic image synthesis technique in terms of depth-of-field modeling. For a given focus setting of a real camera system, the relationship between the blur extent and the object distance is first derived by a defocus estimation process. A simple yet robust blur extent identification method using a solid circle pattern is proposed. The amount of defocus blur used for synthetic DOF generation is based on the calibrated blur-distance curve of the actual camera system. The DOF effect simulation for a given focused image is then achieved by spatial convolution with a distance dependent circular Gaussian mask. By creating the individual camera profiles with defocus blur characteristics, synthetic virtual objects can be inserted into the real scenes with photo-consistent DOF generation. Experiments have demonstrated that the proposed method is capable of photo quality rendering for the real scene images.

2 Camera Model and Image Formation

For a general optical system, the image formation process is usually divided into two separate steps: the light rays collected by the lens and the image intensities generated by the photosensitive elements. Thus, the defocus blur introduced by a digital camera system can be modeled by the geometric optics and the nonlinear intensity response of the image sensor.

2.1 Geometrical Optics and Defocus Blur

As shown in Fig. 1, an optical system consisting of a single convex lens with focal length f is used to derive some fundamental characteristics of focusing based on

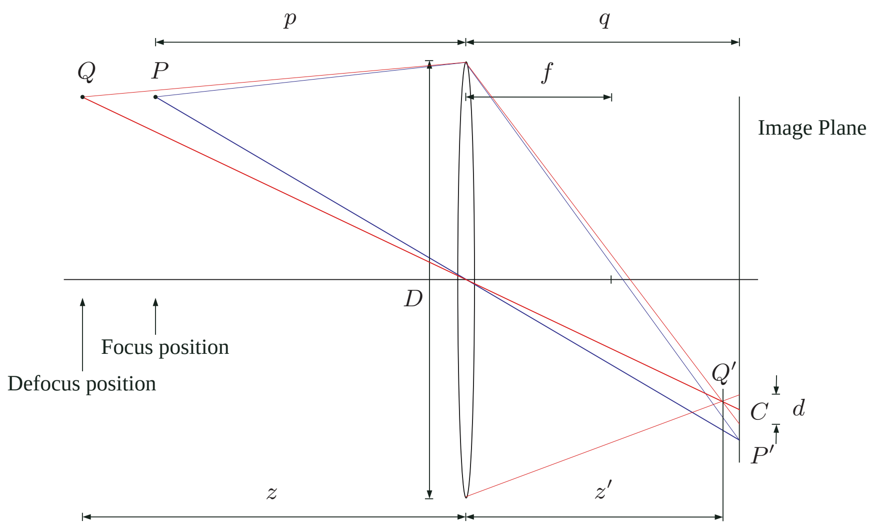


Fig. 1. Camera model for the circle of confusion

geometric optics. Let p represent the focused distance for a given set of camera parameters, then the diameter of CoC for the scene point located at the distance z from the camera is given by

$$d = \frac{Dpf}{p-f}\left(\frac{1}{p} - \frac{1}{z}\right) \tag{1}$$

where D is the diameter of the lens.

It is clear that the size of the CoC depends only on the depth z if fixed camera settings of D, f and p are given. Thus, Eq. (1) can be written as

$$d = c\left(1 - \frac{p}{z}\right) \tag{2}$$

where c is a camera constant represented by

$$c = \frac{Df}{p-f} \tag{3}$$

From the above equations, the diameter of the CoC is linearly related to inverse distance of the object. Furthermore, the constant c given by Eq. (3) represents the maximum size of the CoC since, from Eq. (2), $d \to c$ as the object distance z approaches infinity.

In most cases, it is not possible to obtain the accurate camera parameters such as aperture diameter and focal length without an elaborate calibration process. Thus, the constant c will be given by the blur extent estimated from the scene at the infinity, instead of the direct computation using Eq. (3). It is shown in the experiments that the relationship between the object distance and the corresponding CoC basically follows the curve described by Eq. (2), except for a scale factor given by different values of the constant c. The proposed model is therefore suitable for deriving the blur-distance characteristics of a practical optical system.

2.2 Nonlinear Camera Response Function

During the image acquisition process, it is usually assumed that the image intensity increases linearly with the camera exposure time for any given scene point. For a practical camera system, however, nonlinear sensors are generally adopted to have the output voltage proportional to the log of the light energy for high dynamic range imaging [13,14]. Furthermore, the intensity response function of the image sensors is also affected by the aperture size of the camera. Thus, the defocus blur of an out-of-focus image cannot be simply characterized with only the lens system. The nonlinear behavior of the image sensor response should also be taken into account.

In the previous work, the intensity response curve of a real camera system is modeled as an inverse exponential function of the integration time t, and verified experimentally with back and white printout patterns [15]. For any pixel in the image, the intensity value versus exposure time is given by

$$I(t) = I_{\max}(1 - e^{-kt}) \tag{4}$$

where $I_{\max}$ is the maximum intensity (e.g., 255 for an 8-bit greyscale image) and k is a constant for a given set of camera parameter settings. Since the constant k can be obtained from multiple intensity-exposure correspondences and curve fitting, the intensity response function of a given camera can be fully described by Eq. (4). In general, the value of the constant k is inversely proportional to the aperture size. From the same image acquisition viewpoint, the dark image region captured with a large aperture setting tends to appear even brighter compared to the one captured with a small aperture setting.

If the nonlinear behavior of the image sensor is modeled for the image formation process, the intensities of the defocus blur introduced by the optics are further adjusted according to Eq. (4). Thus, an intensity or irradiance correction step using the corresponding inverse function should be applied prior to image blurring or deblurring. This nonlinear modification of image intensities, however, does not affect the amount of blur extent derived from the geometric optics. The diameter of the circle of confusion for any out-of-focus distance is not altered by this nonlinear intensity adjustment. After the blurring or deblurring process, the synthesized image should recover the nonlinear property of the image sensor using Eq. (4).

3 Camera Profiling and Defocus Blur Synthesis

Based on the image formation model proposed in the previous section, synthetic image generation with depth-of-field effect includes the defocus blur

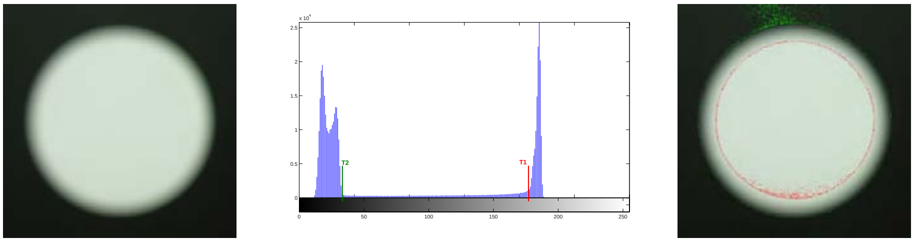


Fig. 2. The real scene image (left figure) and graylevel histogram (middle figure) used for blur identification. The red and green dots in the right figure represent the pixels with intensity thresholds T_1 and T_2, respectively. The camera focus range is set as 30 cm. The pattern is placed at 200 cm from the camera.

identification and modeling for different focused positions and scene distances, the derivation of intensity response curve for the nonlinear image sensors, and the defocus blur synthesis for different object depths.

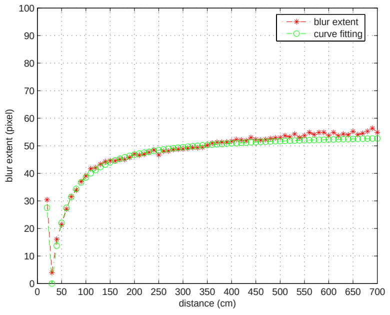
3.1 Blur Identification and Modeling

For any focus range setting given by a camera system, the amount of out-of-focus blur or CoC in the image is a function of scene distance according to Eq. (1). To establish the relationship between the distance and the defocus blur, a black solid circle pattern is used to identify the blur extent for various object positions. Different from the previous approaches which estimate the blur parameter based on a Gaussian point-spread function [16], the blur extent is measured directly from the defocused black and white image in this work. An image graylevel transition identification technique based on histogram analysis is used to identify the blur extent robustly.

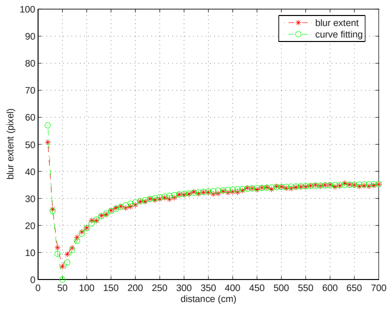
Fig. 2 shows the real image captured with certain degree of defocus blur (left figure) and the corresponding graylevel histogram (right figure). It is clear that two clusters on both sides of the histogram represent the black and white regions in the image. Furthermore, the blur pixels in the images correspond to the histogram areas with intensity values between the two clusters, since the optical defocus process introduces a gradual intensity transition for the black and white images. Let T_1 and T_2 be the upper and lower bounds of the left and right clusters in the histogram, respectively. Then the number of pixels with intensity values below T_1 and T_2 represent the non-white and black regions in the image, respectively. Ideally, these two regions should be bounded by an inner and an outer circle with radii, say R_1 and R_2, respectively. Thus, the blur extent defined by the intensity values between T_1 and T_2 can be derived from

$$b = \varDelta R = R_2 - R_1 = \frac{1}{\sqrt{\pi}}(\sqrt{A_2} - \sqrt{A_1}) \tag{5}$$

where A_1 and A_2 are the number of pixels below thresholds T_1 and T_2, respectively.



(a) The camera is focused at 30 cm.



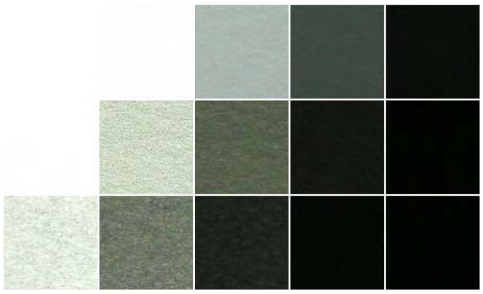
(b) The camera is focused at 50 cm.

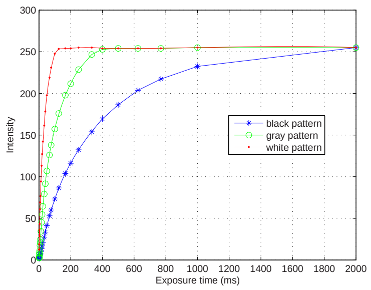
Fig. 3. The blur extent versus object distance

In the above algorithm for blur extent estimation, T_1 and T_2 in the histogram are the only parameters to be identified. These two parameters can be obtained by finding the abrupt pixel count changes for two consecutive graylevels with a given threshold. Since T_1 and T_2 correspond to the upper and lower bounds of the blur regions, they are identified by searching from the middle of the histogram to the left and right, respectively. In practice, due to the acquisition noise and digitization, the image pixels with the intensities T_1 and T_2 might not be perfect circles, as illustrated in Fig. 2 (right figures). Thus, a circle fitting algorithm based on Hough transform is used to derive the radii R_1 and R_2 for blur extent computation. It should be noted that the proposed method does not need to identify the centers of the circles explicitly, which is usually not possible for the defocused images.

Fig. 3 shows the blur extent (in pixel) versus object distance obtained from real images with different camera focus settings. For all cases, the blur extent approaches to an upper limit as the object moves beyond a certain range. The blur extent at the infinity and the distance with minimum defocus blur represent the camera constant c and the focus range p as suggested by Eq. (2), respectively. These two parameters can be derived from the least-squares fitting using the equation and the calibrated blur-distance pairs. The red and green marks illustrated in Fig. 3 represent the identified blur extents using the above algorithm and the curve fitting results, respectively.

It should be noted that the blur extent also depends on the intensities of the captured image pattern. Thus, there might exist a scale factor for the blur extent obtained from a different scene or a different illumination condition. Since the scaling is constant in terms of the scene distance, it can be easily identified by the ratio of the blur extents at the infinity for different scenes. The blur-distance relation curve can then be established and used to synthesize the defocus blur for any given depth.


(a) Images captured with different exposures.



(b) Intensity response functions.

Fig. 4. The images and intensities under different exposures

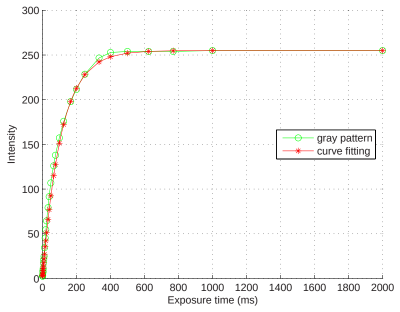
3.2 Derivation of Camera Response Functions

To model the nonlinear behavior of the image sensor and obtain the intensity response function for a given camera parameter setting, an image printout with black, gray and white patterns is used as a test object. Fig. 4(a) shows the images captured with several different exposures for the white, gray and black patterns (from the top to the bottom rows). The average intensity values for a small image region under different camera exposures are shown in Fig. 4(b). The red, green and blue curves corresponds to the intensities of the white, gray and black image patterns, respectively. It is clear that, prior to saturation, the intensity value increases nonlinearly with the exposure time. Since the intensity response curves are equivalent up to an exposure scaling, all of the curves can be modeled by Eq. (4). Fig. 5(a) shows the intensity response function of the gray image pattern (marked in green) and the curve fitting result with $k = 0.009$ (marked in red).

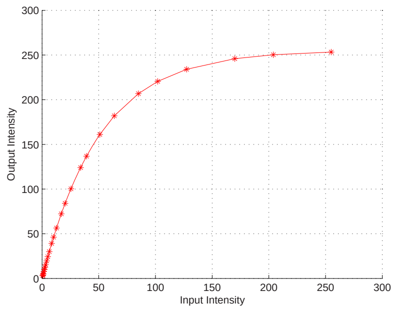
Most image processing algorithms use the intensity value to represent the scene radiance under the assumption of linear camera response function. To incorporate the nonlinearity derived from the photometric calibration, an intensity input-output mapping is created based on the fitting curve as shown in Fig. 5(a). Since the linear model assumes that the intensity increases linearly with the exposure time, the intensity correction can be derived from the full range mapping as illustrated in Fig. 5(b). In the implementation, a lookup table is created based on the intensity input-output relation and applied to the image manipulation pipeline.

3.3 Defocus Blur Image Synthesis

Our algorithm for the synthesis of DOF effect is a post-processing technique. Similar to most previous approaches, a convolution mask is applied to the intensity of each pixel with appropriate point-spread function. However, we have also taken several important characteristics of the real camera model into account,



(a) Curve fitting for intensity response.



(b) Irradiance correction function.

Fig. 5. Intensity response function and irradiance correction

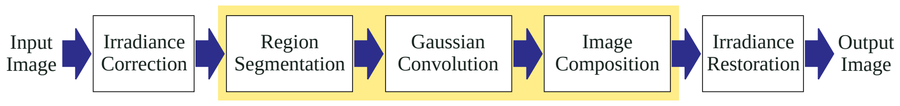
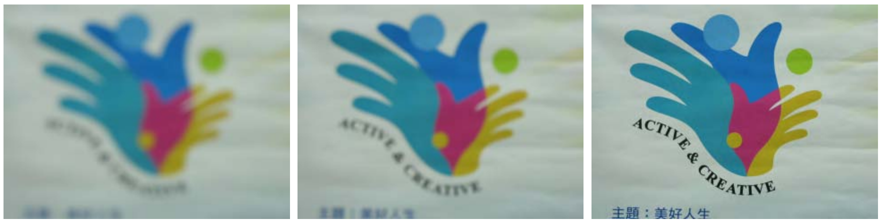


Fig. 6. Synthetic defocus blur generation pipeline

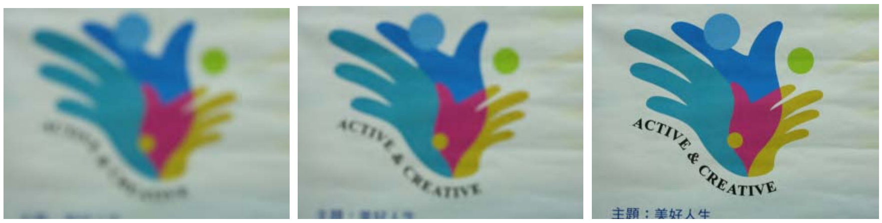
rather than applying simple image filtering with arbitrary parameter settings. The calibrated blur-distance relation is used to create a depth dependent convolution mask. The mask is designed with circular shape to better simulate the real aperture model and eliminate the block effect of the synthesized image. For the nonlinear behavior of the intensity accumulation, the image is pre-processed and post-processed based on the calibrated camera response function.

Fig. 6 illustrates the pipeline of the synthetic defocus blur generation process. The input is an image captured by a real camera system with a small aperture setting, and the output is a synthetic image with depth-of-field effect for a given focus range. At the first stage of the pipeline, an irradiance corrected image is created with the nonlinear intensity mapping given by the input-output relation shown in Fig. 5(b). At the second stage, image regions with different scene distances are separated for depth dependent defocus blur generation. The blur extent for a given distance is derived from Eq. (2) or the curves shown in Fig. 3 at the third stage. The circular Gaussian convolution masks with the sizes of identified blur extents are then created and applied to each image region. Stages four and five are the reverse processes used to blend the individually blurred regions and restore the nonlinear irradiance phenomenon, respectively.

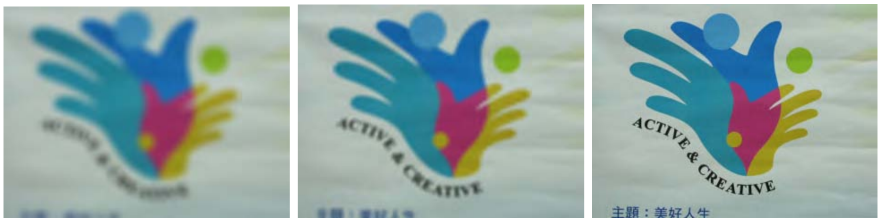
For the DOF generation of a virtual scene, the Z-buffer value of each pixel is used to derive the level of blurring. The synthetic image can then be rendered by light tracing techniques. It should also be noted that the blur-depth relation is calibrated using an intensity based method, and there might be a scale factor for the blur extent under different illumination conditions. Since the scale factor is



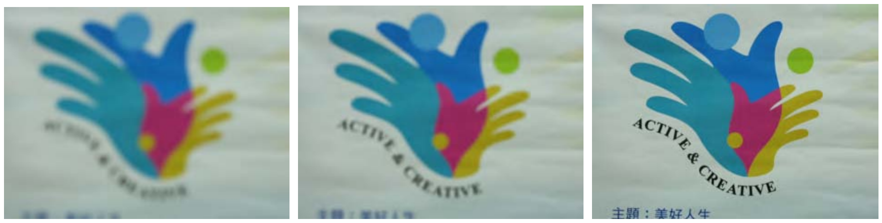
(a) Real scene images captured with different focus ranges.



(b) Synthetic images with irradiance and blur-depth corrections.



(c) Synthetic images without irradiance correction.



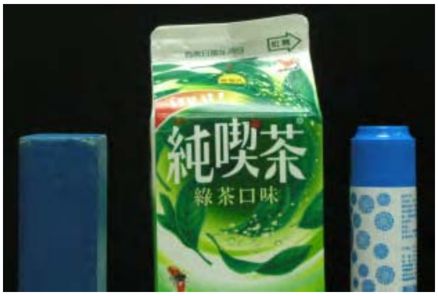
(d) Synthetic images without blur-depth correction.

Fig. 7. Experiments with single depth. The F-number is set as f/1.8 for all of the real scene image captures. The overall execution times for the left, middle and right images are given as follows. Fig. 7(b): 111 sec, 21 sec, 3 sec, Fig. 7(c): 129 sec, 24 sec, 2 sec, Fig. 7(d): 80 sec, 17 sec, 2 sec.

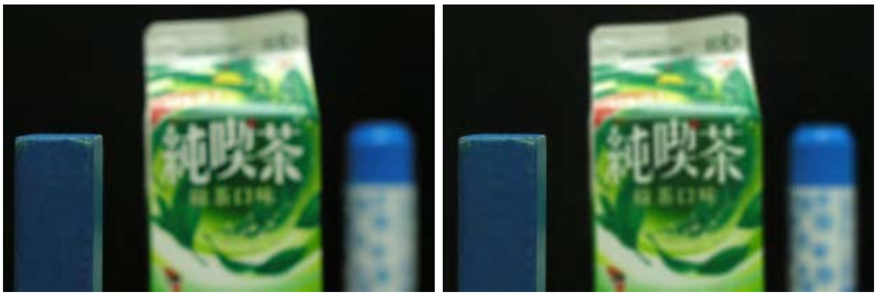
a constant for all distances, including the point at the infinity, it can be derived from an on-site calibration for the maximum blur extent.

4 Results

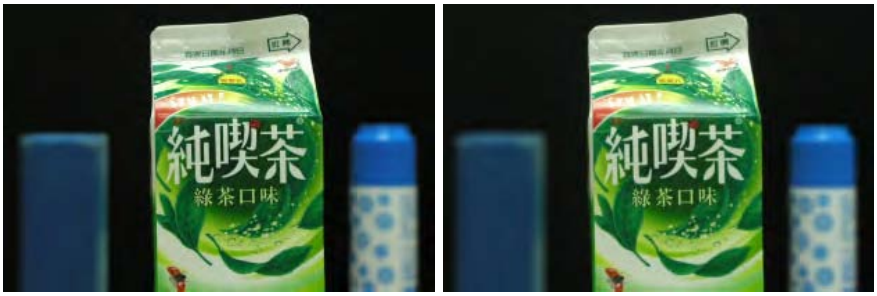
The proposed method for depth-of-field generation has been implemented on the real scene images. For the first experiment, a planar object is placed at 120



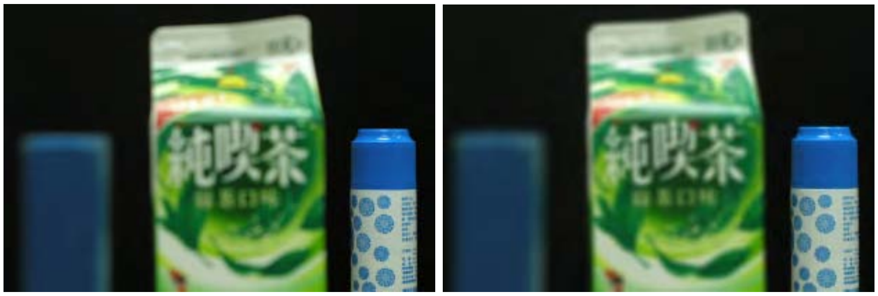
(a) Real image captured without DOF.



(b) The camera focus is on the left object.



(c) The camera focus is on the middle object.



(d) The camera focus is on the right object.

Fig. 8. Experimental results of multiple objects. The overall execution times for Fig. 8(b), Fig. 8(c) and Fig. 8(d) are 196 sec, 46 sec and 29 sec, respectively.

cm in front of the camera. Fig. 7 represents the images captured or synthesized with focus ranges of 30 cm (left), 50 cm (middle) and 100 cm (right). The real images captured by the camera, synthetic images with and without irradiance correction are shown in Fig. 7(a), 7(b) and 7(c), respectively. Fig. 7(d) shows the results without the on-site derivation of the maximum blur offset. Although the blur extents are not correct, the synthesized images fairly resemble the real scene images and are acceptable in terms of visual consistency. The lack of irradiance correction, as illustrated in Fig. 7(c), makes the synthetic images appear darker and much more different from the real photographs.

For the second experiment, three real objects are placed at 30 cm, 50 cm and 100 cm in front of the camera, as shown in Fig. 8 (from the left to the right). The focused image captured with a large F-number (f/10) and used to simulate the depth-of-field effect is shown in Fig. 8(a). To create the depth-of-field for the real scenes, the F-number is set as f/1.8 for all of the image captures. Fig. 8(b) – Fig. 8(d) illustrate the real scene images (left) and the synthetic images with depth-of-field effect generation using our approach (right) for the focus range of 30 cm, 50 cm and 100 cm, respectively.

The processing time for the proposed depth-of-field synthesis method mainly depends on the image resolution and the Gaussian mask size for the convolution operation. The original image resolution for all experiments is 1280×960. For the results shown in Figs. 7(b) and 7(c), the mask sizes of 47×47, 25×25, and 3×3 are applied for the left, middle and right images, respectively. The mask sizes used for Fig. 7(d) are 42×42, 19×19, and 3×3, from the left to the right images. For the synthesized images shown in Fig. 8, various mask sizes between 23×23 and 47×47 are applied on the objects with different depths. The overall execution times reported on a laptop computer with an Intel 2 GHz CPU are shown in the figures.

5 Conclusion

Depth-of-field phenomenon is commonly used in photography or computer graphics to illustrate the focus of attention and depth perception. Most of the existing techniques for DOF simulation adopt a simple camera model and do not consider the characteristics of practical imaging systems. Thus, the realism of the rendered scenes is usually limited to the pure computer generated virtual environments. In the paper, we present a method for the generation of photo-realistic DOF images based on the parameters of a real camera. The defocus blur is modeled by the geometric optics and nonlinear intensity response of the image sensors. Experimental results demonstrate that our approach has taken one step towards photo-realistic synthesis of DOF effect.

Acknowledgments

The support of this work in part by the National Science Council of Taiwan, R.O.C., under Grant NSC-96-2221-E-194-016-MY2 is gratefully acknowledged.

References

1. Kolb, C., Mitchell, D., Hanrahan, P.: A realistic camera model for computer graphics. In: SIGGRAPH 1995. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 317–324. ACM Press, New York (1995)
2. Chen, Y.C.: Lens effect on synthetic image generation based on light particle theory. In: CG International 1987 on Computer graphics 1987, pp. 347–366. Springer, New York (1987)
3. Shinya, M.: Post–filtering for depth of field simulation with ray distribution buffer. In: Graphics Interface 1994, pp. 59–66 (1994)
4. Haeberli, P., Akeley, K.: The accumulation buffer: hardware support for high-quality rendering. In: SIGGRAPH 1990. Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pp. 309–318. ACM Press, New York (1990)
5. Křivánek, J., Zara, J., Bouatouch, K.: Fast depth of field rendering with surface splatting. In: Computer Graphics International, pp. 196–201. IEEE Computer Society Press, Los Alamitos (2003)
6. Scofield, C.: $2\frac{1}{2}$-d depth-of-field simulation for computer animation. In: Graphics Gems III, AP Professional, pp. 36–38 (1992)
7. Potmesil, M., Chakravarty, I.: A lens and aperture camera model for synthetic image generation. In: SIGGRAPH 1981. Proceedings of the 8th annual conference on Computer graphics and interactive techniques, pp. 297–305. ACM Press, New York (1981)
8. Kraus, M., Strengert, M.: Depth-of-Field Rendering by Pyramidal Image Processing. In: Proceedings Eurographics 2007 (2007)
9. Debevec, P.: Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In: SIGGRAPH 1998. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 189–198. ACM Press, New York (1998)
10. Jacobs, K., Loscos, C.: Classification of illumination methods for mixed reality. Computer Graphics Forum 25(23), 29–51 (2006)
11. Rokita, P.: Generating depth-of-field effects in virtual reality applications. IEEE Computer Graphics and Applications 16, 18–21 (1996)
12. Mulder, J.D., van Liere, R.: Fast perception-based depth of field rendering. In: VRST 2000. Proceedings of the ACM symposium on Virtual reality software and technology, pp. 129–133. ACM Press, New York (2000)
13. Debevec, P.E., Malik, J.: Recovering high dynamic range radiance maps from photographs. In: SIGGRAPH 1997. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 369–378. ACM Press, New York (1997)
14. Schanz, M., Nitta, C., Bussman, A., Hosticka, B.J., Wertheimer, R.K.: A high-dynamic-range cmos image sensor for automotive applications. IEEE Journal of Solid-State Circuits 35, 932–938 (2000)
15. Lin, H., Chang, C.: Photo-consistent motion blur modeling for realistic image synthesis. In: Advances in Image and Video Technology, pp. 1273–1282 (2006)
16. Chaudhuri, S., Rajagopalan, A.: Depth from Defocus: A Real Aperture Imaging Approach. Springer, Heidelberg (1998)
17. Kakimoto, M., Tatsukawa, T., Mukai, Y., Nishita, T.: Interactive simulation of the human eye depth of field and its correction with spectacle lenses. In: Proceedings Eurographics 2007 (2007)

Anisotropic Potential Field Maximization Model for Subjective Contour from Line Figure

Osamu Hirose and Tomoharu Nagao

Graduate School of Environment and Information Sciences,
Yokohama National University,
79-7, Tokiwadai, Hodogaya-ku, Yokohama, Kanagawa, 240-8501, Japan
hirose@nlab.sogo1.ynu.ac.jp, nagao@ynu.ac.jp

Abstract. The subjective contour generation in the human brain depends on the interaction of local and comprehensive processes realize this mechanism. The purpose of this paper is to propose a model that outputs subjective contours from line figures. This model, for a local process, detects endpoints and generates potential fields that represent probability of an occluding contour's existence. Each field is anisotropic and spreads perpendicularly to the line figure. Then, for a comprehensive process, the directions of potential fields are corrected to the degree their intersections are maximized in the image. Finally, it fixes subjective contours by tracking potential ridgelines and outputs a result image. The generated subjective contour is smoothly curved and the shape is appropriate compare to what we perceive.

1 Introduction

The visual information that we get is not only the information received optically and what physically exists [1][2]. For example, we clearly perceive a circle in Fig. 1(a) and 1(c), and a vertical line segment in Fig. 1(b) and 1(d). The contour, which is perceived in this phenomenon, is called subjective contour, and many hypotheses of this mechanism in a brain have been suggested.

In physiological research, cells that react to subjective contours were discovered in the visual cortex area V2 [3]. This function suggests that fundamental function and local processing, such as partial edge detection, are important for the mechanism. On the other hand, a psychological study has concluded that our brain aggressively creates information to interpolate the defective region to be consistent in the entire image from the viewpoint of depth perception and others [4][5]. Then, the inducing figure's shape affects the perception [1][2]. Those reports suggest that a comprehensive visual information process in the higher levels of the brain is also important. Although a decisive theory has not yet proven, it is assumed that the interaction realizes the phenomenon.

The purpose of this paper is to propose a model that simulates the subjective contour from line figures on digital images. We tried to reproduce occluding contours by paying attention to depth perception. As a local process, this model computes the probability of an occluding contour's existence (potential) around

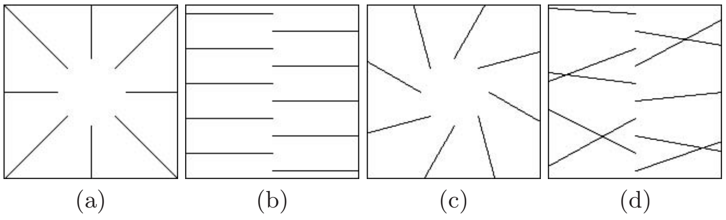


Fig. 1. Figures that induce a subjective contour

the line figure's endpoints, without concluding the subjective contour's shape. The generated potential field anisotropically spreads around the endpoint. As a comprehensive process, the model corrects the direction of the field's distribution in order that the output contour's shape is simple. Finally, when the potential ridgeline connects two endpoints smoothly and continuously, it fixes the ridgeline as a subjective contour. The comprehensive process makes it possible to create the appropriate contour's shape.

2 Prior Studies

Ullman investigated subjective contour's shapes and suggested the shape will consist of two circular arcs, and these curvature radii should be minimal [6]. Then, he suggested that a simple neural network model would reproduce such a shape. Nevertheless, he did not show experimental results.

Kass and Witkin proposed a model that segments the image boundary with smooth curved contours [7]. The model makes a closed contour with a spline-curve, and the curve changes the shape and fits along the object's edges. In addition, the improvement model similar to this is also proposed [8]. Although these models well reproduce a subjective contour in a specific case, the contour must close.

Williams and Jacobs proposed a stochastic model [9]. This model uses particles that can move and decay. The particles start from edges, spread around a specific direction according to the Random-Walk rule. As a result, the model outputs a graded gray level image. However, boundaries that divide a region are not drawn clearly, and differ from the model we propose in that they did not take the comprehensive process of an entire image into consideration.

Kim and Morie devised a system that extends the edge of block figures using the cellular neural network [10]. Although only linear and simple contours are generable, they argued that such a method is suitable for LSI realization.

Sanchez proposed a model using a Log-Gabor filtering and learning system. This system learns the frequency and orientation contained in an image. As a result, the extension of block figure's edges or the continuity of positions of line figure's endpoints is extracted. However, the resulting image does not clearly represent contour.

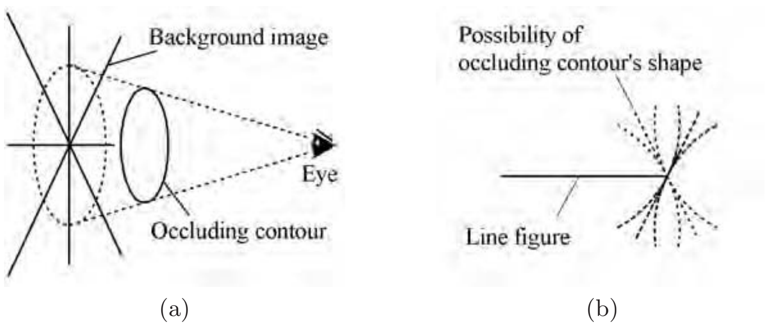


Fig. 2. Relationship of a background image and an occluding contour

Besides, there are models that reproduce subjective contours by high-pass filtering or a neural network system [12][13]. These were appropriate to relatively simple images, for example, in case that the models output only straight lines.

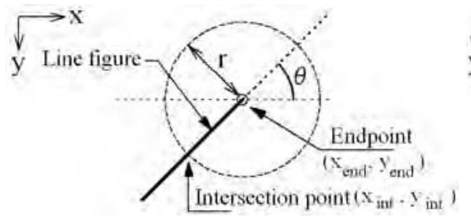
3 Algorithm of this Model

3.1 Fundamental Assumption

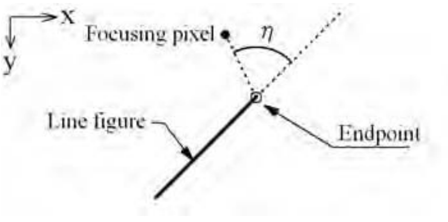
When we visually detect an endpoint of a line figure in an image, it is rational if thinking that our brain feels that the extension line is covered by an occluding figure, as in Fig. 2(a). Given this rule, it is consequential that the subjective contour is perceived to be nearer than the image. We can therefore define that an occluding contour always passes at endpoints first. Then, the line figure and the occluding contour have the possibility to intersect with various angles. Additionally, the occluding contour can become straight and curved, and has the possibility to make various shapes, as indicated in Fig. 2(b). However, the strength of perception from Fig. 1(c) is lower than Fig. 1(a). Therefore, it seemed reasonable to think that the probability of an occluding contour's existence will be intensified in the direction which is perpendicular to a line figure. Nevertheless, in the case of perception of the circle in Fig. 1(c), it has a simple shape similar to the circle from Fig. 1(a). This issue suggests that the direction of potential is not always generated perpendicular to a line figure. In such a case, it seems that a comprehensive process in the brain works in such a way that the perceived contour's shape becomes a relatively simple shape.

3.2 Endpoint and Direction Detection

In this paper, we define inducing figures as line figures with a width of a single pixel. Therefore, if the number of times of change into black from white or white from black is two when investigating the 8-neighbors of a figure's pixel, it detects the pixel as an endpoint. If such a single width line figure cannot be obtained,



(a)



(b)

Fig. 3. Definition of θ and η

a thinning process should be applied beforehand. Such preprocessing enables detection of an endpoint.

At the same time, it takes the direction θ of the line figure to decide the direction of the potential field distribution. We computed the θ by detection at the intersection point of an extension of the line figure and a circle, where the radius is r and the center is the endpoint, as shown in Fig. 3(a). The direction is calculated as $\theta = \arctan((y_{\text{int}} - y_{\text{end}}) / (x_{\text{end}} - x_{\text{int}}))$. We set $r = 7$ pixels.

3.3 Initial Potential Field Generation

The potential field is a function that describes an anisotropic distribution around an endpoint. Shipley examined the relationship of strength of perception and the gap distance of inducing figures [14]. We see in this report that the decrease in strength shows a characteristic behavior similar to a Gaussian distribution as the gap distance widens. Accordingly, we adopted an approach in which the potential field can be described as a Gauusian distribution with standard deviation σ. In addition, for applying a directional function around an endpoint, we assumed that the potential is proportional to $|\sin\eta|^{\alpha}$, where α is a parameter to adjust the attenuation speed of the potential, and η is the angle between an extension of the line figure and a straight line connecting from the endpoint to a focusing pixel, as shown in Fig. 3(b). The initial potential P at a pixel (x, y) is computed by Equations (1) - (3).

$$G_n(x,y) = \frac{1}{\sqrt{2}\sigma} \exp\left(\frac{-\sqrt{(x-X_n)^2 + (y-Y_n)^2}}{2\sigma^2}\right) \tag{1}$$

$$\eta_n = \arctan\left(\frac{y - Y_n}{X_n - x}\right) - \theta_n \tag{2}$$

$$P(x,y) = \sum_{n=1}^{N} G_n(x,y) \mid \sin\eta_n \mid^{\alpha} \tag{3}$$

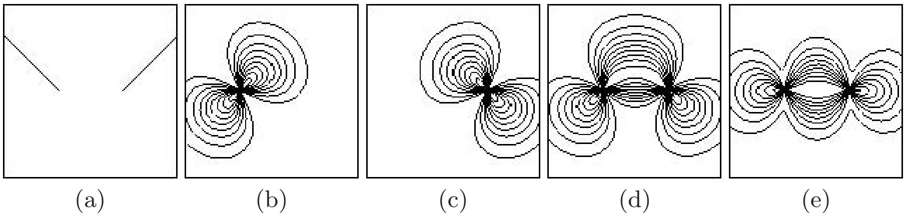


Fig. 4. Multiple potential fields and their composite (a) Line figures (b) Potential field from left figure (c) Potential field from right figure (d) Composed potential field (e) Maximized potential field

Where suffix $n(1 \leq n \leq N)$ is the endpoint number; X and Y are coordinates of endpoints;

We show an image of a potential field in Fig. 4(a) - 4(c). Fig. 4(d), which is composited from Fig. 4(b) and 4(c), shows that the ridgeline draws a smooth curve. The gap distance between the endpoints was 28 pixels, and the parameters were $\sigma = 16.0$ and $\alpha = 2.0$. At this stage, the potential field has a distribution perpendicular to the line figure. The potential is drawn as level curves divided into ten levels.

3.4 Potential Field Maximization

This step corrects the direction of the distribution of the potential fields to ensure that the overlap of potential fields in the image is maximized. This operation is important to make the conclusive subjective contour's shape simple. We compute the degree of overlap P_{sum} according to Equation (4).

$$P_{\text{sum}} = \sum_y \sum_x P(x,y)^2 \tag{4}$$

Next, we show the specific procedures required to maximize P_{sum}.

1. Vary η for an endpoint from $+\delta$ to $-\delta$. Then compute P^+_{sum} and P^-_{sum} according to Equation (4).
2. Compare P^+_{sum} and P^-_{sum} and present value P_{sum}. Then, set η to the angle which produces the largest value.
3. Repeat 1. - 2. for all endpoints.
4. Repeat 1. - 3. until P_{sum} does not increase.

Where δ is the resolution to correct the potential field direction. In this paper, we fixed $\delta = 1.0$ degree. Fig. 4(e) shows a corrected potential field from Fig. 4(d). The simplest shape, which connects both endpoints, will be straight from the viewpoint of the ridgeline's length and curvature. According to the assumption,

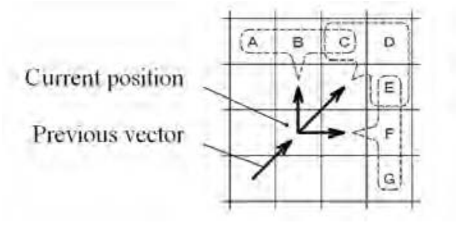


Fig. 5. Potential ridgeline tracking

we see that the ridgeline between the two endpoints was corrected to become a straight line.

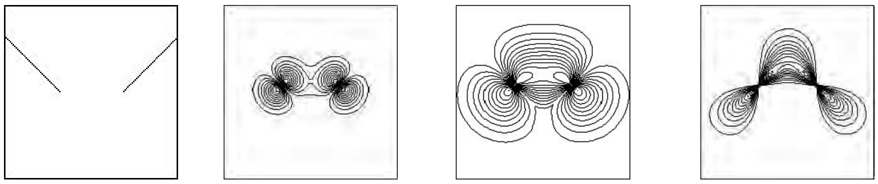
3.5 Ridgeline Tracking and Fixing the Subjective Contour

This step fixes a subjective contour from a potential field. Fundamentally, in brief, when tracking the ridgeline of a potential field and detecting connection continuously from an endpoint to another one, it defines the track as a subjective contour.

We adopted the method of starting from an endpoint and repeating the movement to the maximum potential pixel among the neighboring pixels. However, the change of vector for every movement is limited within $\pm\pi/4$ rad. The reason for limiting the bend range of vector is that the subjective contour is quite unlikely to bend rapidly. Moreover, when choosing the next pixel, this model does not compare the surrounding pixels' potentials directly, but the combined potentials of the three pixels following on from the surrounding pixels. For example, in Fig. 5, when comparing $A + B + C$, $C + D + E$ and $E + F + G$, and if the largest value was $E + F + G$, the next pixel would be right (pixel F). This operation makes it possible to make a smoothly curved shape without rapidly bending. And the track is fixed as a subjective contour if focusing pixel reaches another endpoint. If potential drops below a threshold or returns an already passed pixel, it stops tracking. Incidentally, we set the threshold at 0.1 for all experiments in this paper. This is equivalent to 1/10 of the potential at the endpoints.

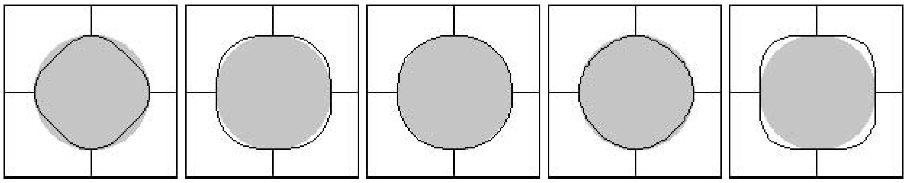
3.6 Examination of Parameters

Here, it is necessary to consider the influence of the parameters σ and α of this model. Fig. 6(b) - 6(d) shows the potential fields, using the respective parameters from Fig. 6(a). Fig. 6(b) and 6(c) show that the ridgeline cannot connect the endpoints if σ is too small. On the other hand, Fig. 6(c) and 6(d) show the effects of α on the shape of the generated contour. Therefore, we examined the parameter conditions that would generate the smoothest possible contour.

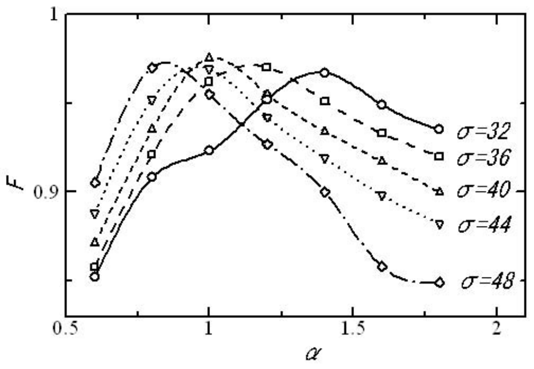


(a) Input image (b) $\sigma = 5$ $\alpha = 1.0$ (c) $\sigma = 10$ $\alpha = 1.0$ (d) $\sigma = 10$ $\alpha = 10.0$

Fig. 6. Parameters and shape of potential field



(a) $\sigma = 32$ $\alpha = 0.6$ (b) $\sigma = 32$ $\alpha = 1.8$ (c) $\sigma = 40$ $\alpha = 1.0$ (d) $\sigma = 48$ $\alpha = 0.6$ (e) $\sigma = 48$ $\alpha = 1.8$



(f) Fitness of generated contour

Fig. 7. Parameters and generated contour's shape

We evaluated the parameters by examining the curvature of the generated contour from four opposing line figures, as shown in Fig. 7(a) - 7(e). The inducing figures and the generated contour are drawn in black, while the ideal circle is shaded in gray. The radius of the circle is 40 pixels. We defined that the fitness F of these shapes is calculated by $F = 1 - (S_\mathrm{d}/S_\mathrm{c})$, where S_c is the area of the ideal circle, and S_d is the difference in number of pixels between the ideal circle and the figure surrounded by a generated contour. Hence, if the generated contour's shape is similar to the ideal circle, then F is likewise close to 1.0.

The variation of F is shown in Fig. 7(f). The model generated stable contours in the range of $32 \le \sigma \le 48$, and the maximum F was 0.975 at $\sigma = 40$ and

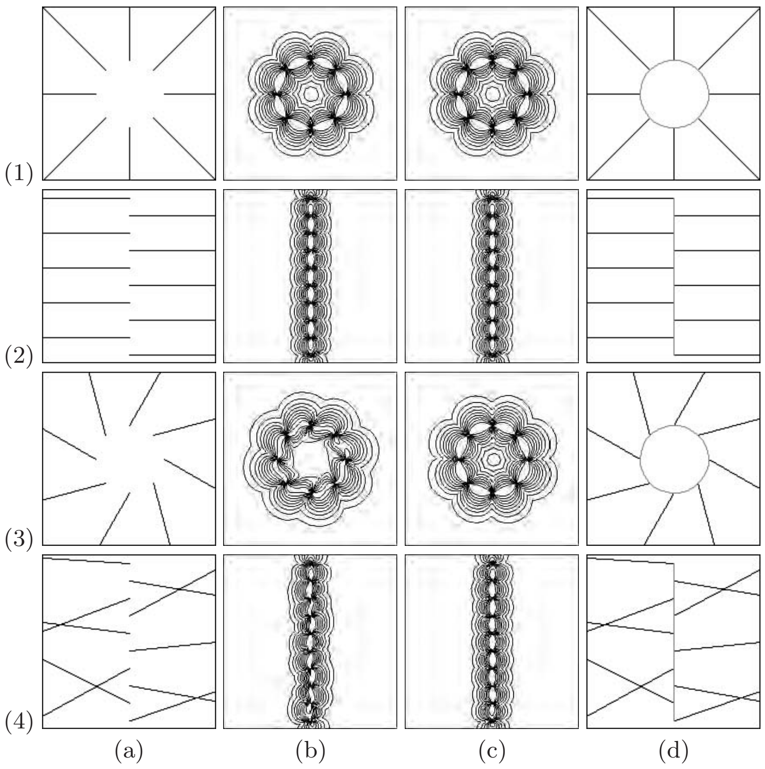


Fig. 8. Simulation of images in Fig. 1 (a) Input image (b) Initial potential field (c) Maximized potential field (d) Output image

$\alpha = 1.0$. Since σ mainly determines the size of a potential field, it should be set such that the gap distance of endpoints will be connected mutually. The distance of the endpoints in this case is 57 pixels. Therefore, in subsequent experiments, we set σ to be 70 percent of the gap distance of the endpoints, which will be connected, and $\alpha = 1.0$.

4 Experimental Results

Fig. 8 shows the simulation results for Fig. 1. The parameters in this experiment are $\sigma = 17$ and $\alpha = 1.0$ for Fig. 8(1) and 8(3), and $\sigma = 11$ and $\alpha = 1.0$ for Fig. 8(2) and 8(4). Fig. 8(1) and 8(2) are images in which the subjective contour crosses perpendicularly to the line figures. Hence, the maximized potential field is the same as the initial potential field. In these cases, this model outputs appropriate subjective contours only by local processing. On the other hand, in Fig. 8(3) and 8(4), the ridgeline meanders and does not output a suitable contour

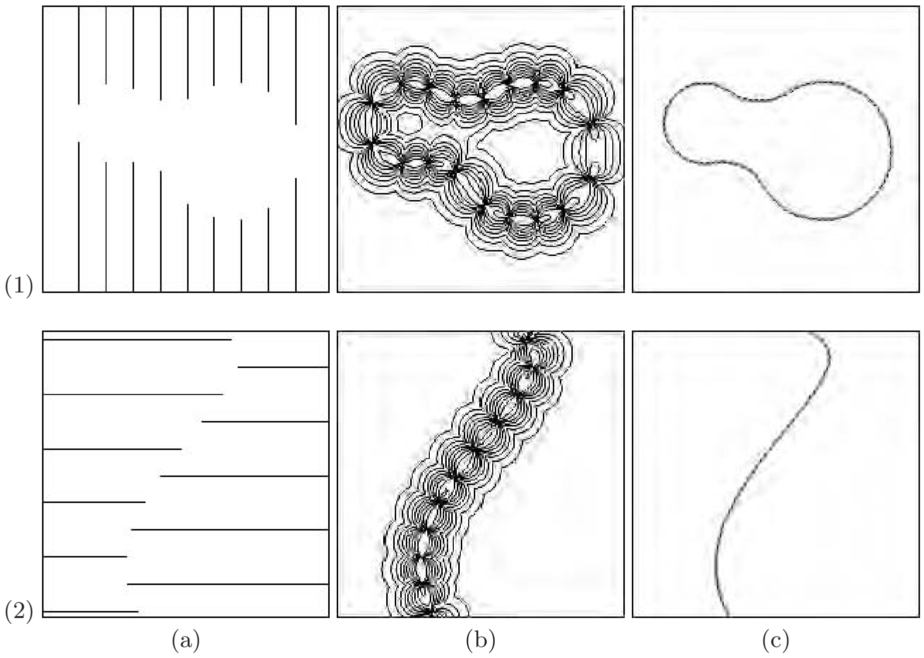


Fig. 9. Simulation of an image which outputs a complicated contour (a) Input image (b) Maximized potential field (c) Overlay of generated contour and occluding contour (*gray line*)

after local processing. By applying potential field maximization, the subjective contour's shape becomes proper.

To consider more complicated contour's shapes, we present the simulation results in Fig. 9. These images are 200 × 200 pixels, and the parameters are $\sigma = 16.0$ and $\alpha = 1.0$ for both images Fig. 9(1) and 9(2). We made the input images in Fig. 9(a) by putting occluding figures, which are drawn in a gray line in Fig. 9(c), over grating images. The shape of Fig. 9(1) consists of circular arcs, and Fig. 9(2) is a spline curve. Fig. 9(b) shows that the potential field's ridgeline makes a smooth curve. Additionally, we made images to compare the occluding contour's shape and the generated contour's shape, shown in Fig. 9(c). The shapes correspond well and the maximum deviation of the two figures was 2 pixels for both cases. This result indicates that the model can reproduce occluding contours properly.

5 Conclusion

We proposed a subjective contour model that is applicable to line figures. The model's fundamental aim is to compute the probability of an occluding contour's existence. This model outputs not only smoothly curved contours,

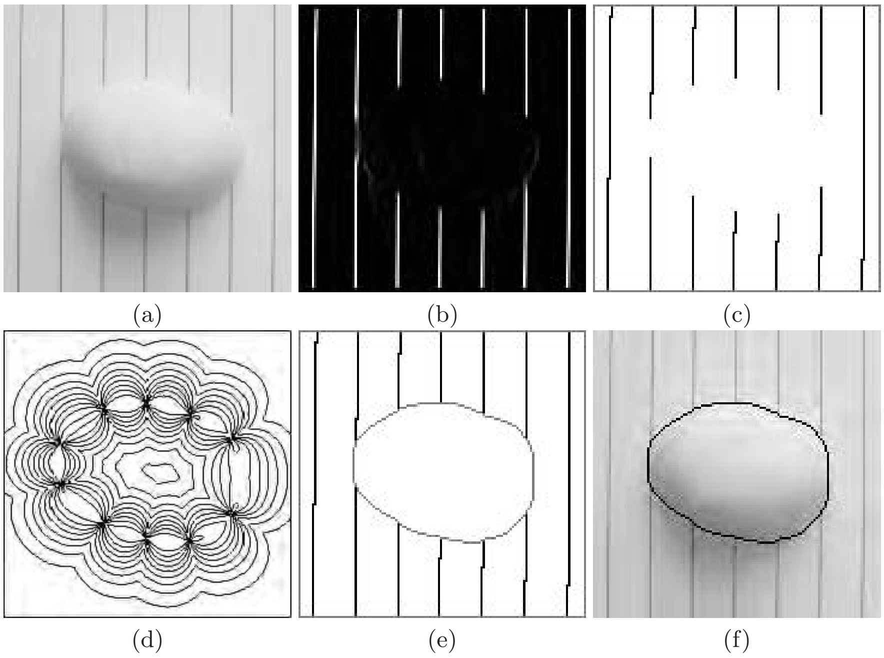


Fig. 10. Application of a real image(a) Input image (*A marshmallow on the tray with grating*) (b) Extracted grating by horizontal Laplacian filter (c) Binarized grating by thresholding and thinning (d) Maximized potential field (e) Output image (f) Overlay of (a) and generated contour

but also shapes which correspond well to what we perceive. Furthermore, the model can both output closed contours and segments. We expect that we can apply this study to complement the incomplete area of various types of images.

Here, we show an example of the application of the proposal technique. Fig. 10(a) shows a piece of white food placed on a white tray. In general, manufacturers tend to use white trays or white conveyor belts for handling food. This makes finding dirt and mold easy. In a food manufacturing process, image-processing systems inspect size or calculate mean positions so that food may be handled by a robotic arm. However, in many cases, finding an object's outline is difficult, such as in this image. In this case, by drawing a grating pattern on the background and applying the proposed technique, estimating the object's outline becomes possible. We show this in Fig. 10(b) - 10(f).

It is, however, necessary to adjust parameters for an expected result. We have to adjust σ according to the gap distance of endpoints. If the potential field' distribution is too small, a ridgeline cannot connect endpoints or the generated contour becomes short and stright; if too broad, the output shape will becomes bended line. Probably, in order to improve this model, the size of potential field's distribution should be automatically adjusted corresponding to the gap distance. This issue may be the next theme we have to solve.

References

1. Kanizsa, G.: Subjective Contours. Scientific American 234(4), 48–52 (1976)
2. Kanizsa, G.: Organization in Vision: Essays on Gestalt Perception. Praeger, New York (1979)
3. von der Heydt, R., Peterhans, E.: Mechanisms of Contour Perception in Monkey Visual Cortex. Jounal of Neuroscience 9, 1731–1748 (1989)
4. Coren, S.: Subjective Contour and Apparent Depth. Psychological Review 79, 359–367 (1972)
5. Rock, I., Anson, R.: Illusory Contours as the Solution to a Problem. Perception 8, 665–681 (1979)
6. Ullman, S.: Filling-in the Gaps: The Shape of Subjective Contours and a Model for Their Generation. Biological Cybernetics 25, 1–6 (1976)
7. Kass, M., Witkin, A., Terzopouls, D.: Snakes: Active Coutour Models. International Journal of Computer Vision 1, 321–331 (1987)
8. Zhu, W., Chan, T.: Illusory contours using shape information. UCLA CAM Report, 3–9 (2003)
9. Williams, L.R., Jacobs, D.W.: Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Salience. In: International conference on computer vision, pp. 408–415 (1995)
10. Kim, Y., Morie, T.: Subjective contour generation using a pixel-parallel anisotropic diffusion algorithm. In: International congress series, vol. 1291, pp. 237–240 (2006)
11. Rodriguez-Sanchez, R., Garcia, J.A., Fdez-Valdivia, J., Fdez-Vidal, X.R.: Origins of illusory percepts in digital images. Pattern Recognition, 33, 2007–2017 (2000)
12. Ginsburg, A.P.: Is Illusory Triangle Physical or Imaginary? Nature 257, 215–220 (1975)
13. Skrzypek, J., Ringer, B.: Neural Network Models for Illusory Contour Perception. Conputer Vision and Pattern Recognition, 681–683 (1992)
14. Shipley, T.F., Kellman, P.J.: Strength of visual interpolation depends on the ratio of physically specified to total edge length. Perception & Psychophysics 52(1), 97–106 (1992)

Surface Reconstruction from Constructive Solid Geometry for Interactive Visualization

Doug Baldwin

Department of Computer Science
SUNY Geneseo

Abstract. A method is presented for constructing a set of triangles that closely approximates the surface of a constructive solid geometry model. The method subdivides an initial triangulation of the model's primitives into triangles that can be classified accurately as either on or off of the surface of the whole model, and then recombines these small triangles into larger ones that are still either entirely on or entirely off the surface. Subdivision and recombination can be done in a preprocessing step, allowing later rendering of the triangles on the surface (*i.e.*, the triangles visible from outside the model) to proceed at interactive rates. Performance measurements confirm that this method achieves interactive rendering speeds. This approach has been used with good results in an interactive scientific visualization program.

1 Introduction

Constructive solid geometry (CSG) is a technique for modeling three-dimensional solids as set-theoretic combinations of simple primitive shapes [1]. Common primitives are such shapes as cylinders, cones, spheres, polyhedra, *etc.* Combining operations typically include union, intersection, and difference or complement. CSG was first used to represent solid models for computer aided design and manufacturing, and has since found applications in computer graphics and other areas.

The work reported in this paper is motivated by a need to render CSG-defined geometries in certain particle physics visualizations. Specifically, a visualization tool named IViPP [2] is being developed to support visual analysis of results from the MCNPX [3] simulator. MCNPX simulates reactions between subatomic particles, using a form of CSG to describe the physical environment within which the reactions occur. IViPP needs to display both particle data and the geometry of the surrounding environment. It must update its displays at interactive rates, fast enough for users to smoothly rotate, zoom, and similarly manipulate their view.

This paper's main contribution is a method for constructing a small set of triangles that closely approximates the surface of a CSG model. The set of triangles can be constructed in a preprocessing step, and subsequently rendered at interactive rates. Section 2 compares this approach to previous ways of rendering

CSG, while Section 3 describes the method itself. Section 4 presents data regarding the performance and effectiveness of this approach. Section 5 summarizes the work's status and suggests directions for further research.

2 Background and Previous Work

The method described in this paper builds a mesh of triangles representing a CSG model's surface; similar approaches have also been pursued by other researchers. For example, the "constructive cubes" algorithm [4] adapts the marching cubes isosurface construction algorithm [5] to approximate the surface of a CSG model. The ACSGM approach [6] is also based on marching cubes, but is considerably more sophisticated than constructive cubes in how it approximates the surface of the CSG model. Chung [7] uses a three-stage method, consisting of spatial subdivision followed by triangulation proper followed by triangle refinement to preserve sharp edges and corners. More recently, Čermák and Skala describe a method for triangulating implicit surfaces [8] that could be adapted to CSG. The method proposed in the present paper is conceptually simpler than ACSGM or Chung's approach, and unlike any of the previous efforts, detects unnecessarily small triangles and combines them into larger ones to reduce the total number of triangles. Like constructive cubes and ACSGM, this paper's method can trade image quality for computing resources. However, the primary resource consumed by this paper's method is time, whereas constructive cubes consumes significant amounts of both time and memory (the resource requirements of ACSGM are not discussed in [6]).

Triangulating the surface of a CSG model can be an unacceptable bottleneck for CAD applications in which users want to edit models and see the results in real time [9]. However, in visualization, geometry is often static, and users need only simple real-time interactions (*e.g.,* changes of viewpoint). Triangulated surfaces are attractive in such settings, because the triangulation itself can be rendered very quickly, and it only needs to be constructed once—after construction, different renderings of the same set of triangles display the surface from whatever viewpoints are needed.

One of the oldest alternatives to surface triangulation for CSG rendering is ray tracing [10]. However, classic ray tracing cannot be done at interactive speeds, due to the need to compute multiple ray-primitive intersections for every pixel in the display. Speeds can be improved by dividing the CSG model into spatial subregions in such a way that only a few primitives lie in each region [11]; recent research has divided the model in such a way that the primitives in each subregion can be ray traced in the GPU [12]. GPU-assisted ray tracing achieves interactive speeds on small to medium-size models, but does require custom GPU programming. In contrast, the approach introduced in this paper reduces CSG rendering to drawing triangles, something that is well-supported by modern graphics hardware without custom programming.

Goldfeather [13] proposed a method that uses hardware depth and color buffers to evaluate and render CSG models. Wiegand [14] adapted this idea to more widely available graphics hardware, using a depth buffer (albeit capable of being saved to and restored from main memory) and a stencil buffer, and showed how to access that hardware via standard APIs such as OpenGL. Subsequent work [15,16] improved the asymptotic execution time of depth-and-stencil-buffer rendering algorithms, and showed how to perform the necessary buffer comparisons in the GPU [9,17]. These improvements have yielded interactive rendering speeds for some models, but the algorithms still require considerable computation for each frame, and are sensitive to hardware characteristics (e.g., depths of buffers, memory-to-frame-buffer bandwidths, GPU capabilities).

Liao and Fang describe a volumetric approach to CSG rendering [18] that builds a three-dimensional texture map from a CSG model and then renders the model by drawing slices through that texture map. Because this approach can build the texture map prior to rendering, it can run in constant time per frame, and can easily achieve interactive speeds. However, it requires large amounts of memory for the texture map, and care must be taken to avoid aliasing in the rendered images.

3 Surface Reconstruction by Triangle Subdivision

A small set of triangles that approximates the surface of a CSG model can be constructed by the process illustrated in Figure 1. The process begins with triangulations of the model's primitives, created without concern for how primitives interact in the overall model. Each triangle from a primitive is then subdivided into smaller triangles. Vertices of these subtriangles coincide as much as possible with intersections between the edges of the triangle and the surface of the model. Subdivision continues until the triangles are small enough to be classified as either inside the modeled object, on its surface, or outside the object without producing visual anomalies. Only those triangles on the model's surface need be drawn in order to render the model. The total number of such triangles is reduced by recombining subtriangles into their parent triangle whenever the parent's subtriangles are either all on the surface or all off (inside or outside) it.

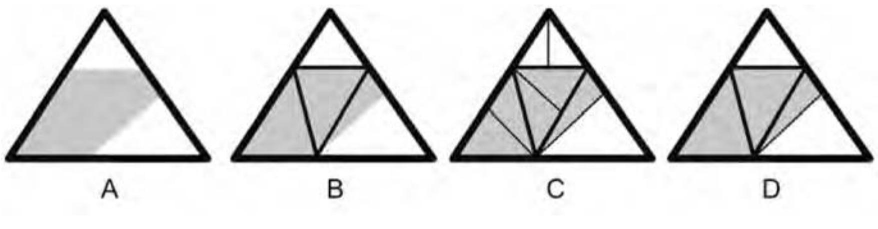


Fig. 1. Triangle subdivision. Initial triangle, with portion on model's surface in gray (*A*); subtriangles after one (*B*) and two (*C*) levels of subdivision; subtriangles are recombined into their parent if all all are on or all are off the surface (*D*).

3.1 Triangle Subdivision

The visual quality of the images produced by triangle subdivision, and the speed with which they can be rendered, depend on how triangles are divided in order to approximate the CSG model's surface. Specifically, the division scheme should yield a small number of triangles, while faithfully representing the surfaces, edges, and corners of the model. Triangle subdivision is therefore driven by changes in the classification of triangle edges relative to the CSG model (*i.e.,* whether an edge is inside the modeled object, outside it, or on its surface). Changes in classification can only happen at intersections between the edge and the surface of the CSG model, which are found in the standard manner [10]: solve the equations for the points at which edges intersect primitives, split edges into segments at these points, and then combine segments according to the Boolean operations used to combine primitives. To avoid the need for extreme numeric accuracy to determine that a segment lies on a surface of a primitive, triangle edges are considered to lie on a primitive if either that primitive is the "source" for the edge's triangle (primitive p is the source for triangle t if t is one of the triangles produced by triangulating p, or if t is a subtriangle of a triangle whose source is p), or if the entire edge lies within a small tolerance of the primitive's surface. Once the points at which classifications change are known, a triangle is divided into subtriangles according to rules 1 through 4 below. Figure 2 summarizes these rules, using dots to indicate points at which an edge's classification changes.

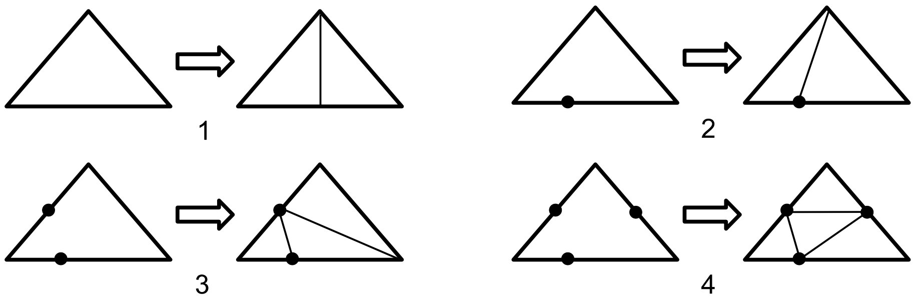


Fig. 2. Triangle subdivision rules

1. If no edge of the triangle changes classification, split the triangle into two subtriangles along the line from the center of the longest edge to the opposite vertex. Note that even if no edge changes classification, the triangle's interior might; this rule ensures that subdivision continues until any changes in classification anywhere in a triangle must intersect an edge.
2. If exactly one edge changes classification, split the triangle into two subtriangles along the line from one of the points at which that edge's classification changes to the opposite vertex.

3. If two edges change classification, split the triangle into three subtriangles, along a line connecting one classification-change point from each edge, and along the line from one of these points to the opposite vertex. If either edge changes classification multiple times, use the classification-change points closest to the edges' common vertex.
4. If all three edges change classification, split the triangle into four subtriangles along lines between one change point from each edge. For two of the edges, use the change points closest to the common vertex, as in Rule 3; for the third edge, choose a change point arbitrarily.

These rules keep the overall number of triangles small by maximizing the likelihood that different subtriangles will lie on different sides of an edge of the CSG model (rules 2, 3, and 4), or by making progress towards reducing the size of triangles to the point where further splitting is unnecessary (rule 1).

3.2 Subdivision Order and Stopping Criteria

Both the order in which triangles are considered for subdivision and the criteria for stopping subdivision can be tuned to reduce the number of triangles produced and the time required to do so. Tuning is supported by storing triangles awaiting subdivision in a priority queue. Triangles are subdivided when they are removed from this queue, with subtriangles placed back in the queue. The goal of the priority function is to divide visually important triangles before unimportant ones, but different functions can reflect different measures of importance. For example, triangles may be prioritized by size (divide larger, and thus visually more prominent, triangles before smaller ones), age (divide older triangles, likely to be cruder approximations to the model's surface, before newer triangles), *etc.* In all cases, a priority of 0 or less indicates that a triangle should be classified without further subdivision. Priority functions can take advantage of this fact to limit the time or other resources used by triangle subdivision: setting priorities to 0 when the allocated resources have been exhausted stops further subdivision and forces the triangles currently in the priority queue to receive "best guess" classifications.

When a triangle no longer needs subdividing, it is classified as either on the model's surface or off the surface, based on the heuristic that triangles are on the surface if and only if more than half of the total length of their edges is.

It is sometimes possible to stop subdividing a triangle sooner than the priority function would. In particular, if all of a triangle's edges are on the surface, or all are off, and no edge changes its classification, then subdivision can stop as soon as the shortest edge is shorter than the model's minimum feature size. This early end to subdivision is permissible because when a triangle's shortest edge is shorter than the model's minimum feature, any feature that affects the classification of the triangle's interior must also intersect an edge. If no such intersections occur, as indicated by no edge changing its classification, then the entire triangle has the same classification as its edges. Minimum feature size must be estimated, but simple estimates work well. For example, the implementations discussed below estimate minimum feature size as half the size of the smallest feature in any primitive.

Finally, triangles that are "small" by various measures are discarded instead of being further divided. In particular...

– Subtriangles that cover only a small fraction of their parent's area leave large siblings to be processed by future subdivisions and are visually insignificant. To avoid creating such subtriangles, changes in edge classification in either the first or last 0.05% of an edge are ignored when dividing triangles.
– Triangles with nearly collinear vertices are numerically unstable. Therefore, triangles are discarded if the sum of the lengths of the two short edges is nearly equal to the length of the longest edge.
– Triangles with nearly equal vertices are also numerically unstable. Therefore, triangles are discarded if, for any edge, the ratio l/d is close to the machine epsilon for floating point numbers, where l is the length of the edge and d is the distance from the origin to one of the edge's endpoints (l/d close to epsilon indicates that changes in vertex coordinates across the edge will be difficult to recognize numerically).

3.3 The Complete Algorithm

Figure 3 summarizes the complete algorithm for triangulating and rendering a CSG model. This figure presents two central functions, "triangulate" and "draw." "Triangulate" generates a list of triangles that approximates the surface of a CSG model, while "draw" renders these triangles to some display device. Calls on "triangulate" and "draw" may be separated. In particular, the time-consuming triangulation can be done once when the model is created (or when it changes), with only the faster drawing done for each frame.

4 Results

A prototype program for CSG rendering by triangle subdivision has been coded using C++ and OpenGL. Figure 4 shows two CSG models rendered by this program. For both models, triangles were prioritized for subdivision by the length of their longest side, and subdivision stopped when that length was less than the equivalent of five pixels on the display device.

The left-hand image in Figure 4 is a perforated shell, constructed by subtracting a number of cylinders (the perforations) from a difference of two spheres (the shell). Table 1 presents the performance of triangle-subdivision rendering on several variations on this model. The variations differ in the number of cylinders removed from the shell, and thus in the total number of primitives in the model. The "Triangles Created" and "Triangles Drawn" columns are the total number of triangles created during subdivision, and the number actually used for drawing after recombination, respectively. Recombination is able to remove about 90% of the triangles generated. "Frames per Second" is the rate at which the triangulated model is rendered. Speeds of over 10 frames per second were achieved

```
triangulate( CSG model m )
    list of triangles l = triangulations of primitives in m
    priority queue q = priority queue containing triangles in l
    while q is not empty
        dequeue triangle t from q
        if t.priority <= 0
            v = sum of lengths of parts of edges of t on m’s surface
            d = sum of lengths of edges of t
            if v > d / 2
                t.classification = ON
            else
                t.classification = OFF
        else if t’s shortest edge is shorter than m’s minimum feature
                  and no edge of t changes its classification
            if all edges of t have classification ON
                t.classification = ON
            else
                t.classification = OFF
        else if t is not "small"
            t.subtriangles = divide t per rules in Figure 2
            for each triangle s in t.subtriangles
                enqueue s in q
            t.classification = UNKNOWN
    end while
    for each triangle t in l
        recombine( t )
    return l

draw( triangle t )
    if t.classification == UNKNOWN
        for each triangle s in t.subtriangles
            draw( s )
    else if t.classification == ON
        render t to display

recombine( triangle t )
     if t.classification == UNKNOWN
         for each triangle s in t.subtriangles
             recombine( s )
         if all members of t.subtriangles have same classification, c
             t.classification = c
```

Fig. 3. Triangulating and rendering a CSG model

in all cases. All data was collected on an Apple MacBook Pro with 2.16 GHz Intel Core 2 Duo processor and ATI Radeon X1600 video chipset, running under Mac OS X 10.4.10 and OpenGL 2.0. Frame rates were measured using Apple’s OpenGL profiler.

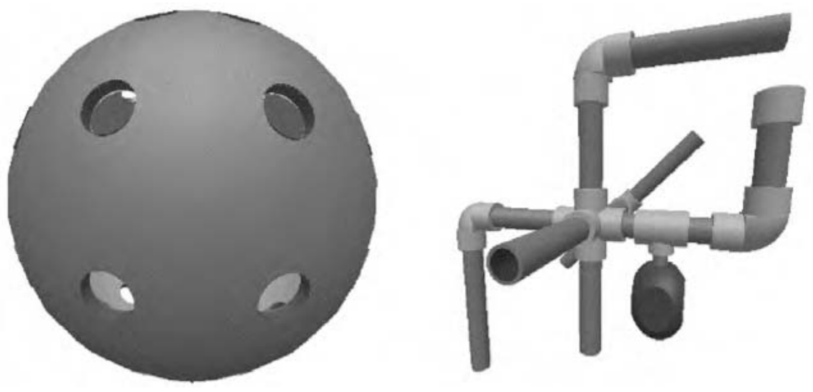


Fig. 4. Some CSG models rendered by triangle subdivision: perforated shell (left) and pipe network (right)

Table 1. Triangulation and rendering data for perforated shells

Primitives	Triangulation Time (Sec)	Triangles Created	Triangles Drawn	Frames per Second
8	2.58	30260	3990	89.7
16	9.20	55166	6692	54.2
28	27.12	93336	10597	34.2
44	65.57	142558	15973	22.5
64	141.44	209060	22596	16.1
88	270.11	285209	30594	12.0
116	420.13	332448	35426	10.2

CSG rendering by triangle subdivision has also been incorporated into IViPP, which is beginning to see production use as a visualization tool for physics research. Typical geometries in this setting consist of several tens of "cells," each described by a CSG expression containing on the order of 1 to 3 primitives. IViPP treats each cell as a separate CSG model. Triangle subdivision is used to triangulate each cell, and cells are rendered by rendering their surface triangles. A standard depth buffer then handles occlusions of one cell by another, occlusions of particle tracks by cells and cells by tracks, *etc.* IViPP prioritizes triangles for subdivision by size and the time that has elapsed since beginning triangulation: during the first second, a triangle's priority is the length of its shortest edge; after one second priorities are always 0, preventing further subdivision. IViPP is successfully visualizing models as large as a nuclear reactor consisting of 71 cells collectively containing 138 primitives. IViPP takes about 1 minute to triangulate this model, and renders it at about 12 frames per second (using the same computer configuration that produced the performance data for the perforated shell).

5 Conclusions

A new way of rendering CSG models at interactive speeds has been described. Distinguishing features of this method include using triangle subdivision followed by recombination to generate a small set of triangles that closely follows the surface of the CSG model, a priority mechanism that allows image quality to be balanced against resource usage, and the ability to separate time-consuming analysis of the CSG model from rendering. Rendering speeds of over 10 frames per second have been achieved on models containing over 100 primitives, an entirely adequate speed for interactive applications. The method is in successful production use in the IViPP visualization program.

The factor that ultimately limits the utility of this method is triangulation time, not rendering time. However, time can be a factor in prioritizing triangles for subdivision, allowing triangulation time to be controlled. Experience with IViPP suggests that such time limits do not seriously compromise image quality.

CSG rendering by triangle subdivision can be extended in a number of ways. Perhaps most significantly, it assumes that triangulations of primitives are easy to generate. While this is true for many primitives, some CSG systems support primitives whose triangulations are not obvious (*e.g.,* general quadric surfaces). The method also needs to be tested on very large CSG models, *i.e.,* ones containing thousands of primitives. Since triangles divide independently of each other, the concurrency provided by multicore processors seems a promising way to speed triangle subdivision, but has not yet been investigated. Finally, additional combinations of priority schemes and resource limits for subdivision may yield further improvements in image quality and triangulation time.

Acknowledgments

This work was supported in part by a grant from the U. S. Department of Energy through the University of Rochester Laboratory for Laser Energetics. Ryan Kinal, Heather Warren, and Justin Hagstrom are largely responsible for incorporating the rendering method described here into IViPP. Dr. Sharon Stephenson and her students at Gettysburg College are early adopters of IViPP, and provided the reactor model mentioned in Section 4. Thanks to Dr. Stephen Padalino of SUNY Geneseo for support of the entire IViPP project.

References

1. Requicha, A.A.G.: Representations for rigid solids: Theory, methods, and systems. ACM Computing Surveys 12, 437–464 (1980)
2. Baldwin, D.: Architecture of the IViPP particle visualization program (2004), Available at `http://cs.geneseo.edu/~baldwin/ivipp/ivipparch.html`
3. Los Alamos National Laboratory: MCNPX^{TM} Users Manual (LA-UR-02-2607) (2002), Available at `http://mcnpx.lanl.gov/opendocs/versions/v230/MCNPX_2.3.0_Manual.pdf`

4. Breen, D.E.: Constructive cubes: CSG evaluation for display using discrete 3-D scalar data sets. In: Proceedings of Eurographics 1991, pp. 127–142. Elsevier Science Publishers, Amsterdam (1991)
5. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 163–169. ACM Press, New York (1987)
6. Purgathofer, W., Tobler, R.F., Galla, T.M.: ACSGM: An adaptative CSG meshing algorithm. In: Proceedings of CSG 1996, Information Geometers Ltd. (1996)
7. Chung, C.W., Chuang, J.H., Chou, P.H.: Efficient polygonization of CSG solids using boundary tracking. Computers and Graphics 21, 737–748 (1997)
8. Čermák, M., Skala, V.: Adaptive edge spinning algorithm for polygonalization of implicit surfaces. In: Proceedings of Computer Graphics International 2004, pp. 36–43. IEEE Computer Society Press, Los Alamitos (2004)
9. Hable, J., Rossignac, J.: CST: Constructive solid trimming for rendering BReps and CSG. IEEE Transactions on Visualization and Computer Graphics 13 (2007)
10. Roth, S.D.: Ray casting for modeling solids. Computer Graphics and Image Processing 18, 109–144 (1982)
11. Chuang, J.H., Hwang, W.J.: A new space subdivision for ray tracing CSG solids. IEEE Computer Graphics and Applications 15, 56–62 (1995)
12. Romeiro, F., Velho, L., Figueiredo, L.H.d.: Hardware-assisted rendering of CSG models. In: Proceedings of the XIX Brazilian Symposium on Computer Graphics and Image Processing, pp. 139–146. IEEE Computer Society Press, Los Alamitos (2006)
13. Goldfeather, J., Molnar, S., Turk, G., Fuchs, H.: Near real-time CSG rendering using tree normalization and geometric pruning. IEEE Computer Graphics and Applications 9, 20–28 (1989)
14. Wiegand, T.: Interactive rendering of CSG models. Computer Graphics Forum 15, 249–261 (1996)
15. Stewart, N., Leach, G., John, S.: Linear-time CSG rendering of intersected convex objects. In: Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 437–444 (2002)
16. Stewart, N., Leach, G., John, S.: Improved CSG rendering using overlap graph subtraction sequences. In: Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia – GRAPHITE 2003, pp. 47–53. ACM Press, New York (2003)
17. Guha, S., Krishnan, S., Munagala, K., Venkatasubramanian, S.: Application of the two-sided depth test to CSG rendering. In: Proceedings of the 2003 Symposium on Interactive 3D Graphics, pp. 177–180. ACM Press, New York (2003)
18. Liao, D., Fang, S.: Fast volumetric CSG modeling using standard graphics system. In: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, pp. 204–211. ACM Press, New York (2002)

Interactive Glyph Placement for Tensor Fields

Mario Hlawitschka[1], Gerik Scheuermann[1], and Bernd Hamann[2]

[1] Image and Signal Processing Group, University of Leipzig, Germany
[2] Institute for Data Analysis and Visualization (IDAV) and
Department of Computer Science, University of California, Davis, USA

Abstract. Visualization of glyphs has a long history in medical imaging but gains much more power when the glyphs are properly placed to fill the screen. Glyph packing is often performed via an iterative approach to improve the location of glyphs. We present an alternative implementation of glyph packing based on a Delaunay triangulation to speed up the clustering process and reduce costs for neighborhood searches. Our approach does not require a re–computation of acceleration structures when a plane is moved through a volume, which can be done interactively. We provide two methods for initial placement of glyphs to improve the convergence of our algorithm for glyphs larger and glyphs smaller than the data set's voxel size. The main contribution of this paper is a novel approach to glyph packing that supports simpler parameterization and can be used easily for highly efficient interactive data exploration, in contrast to previous methods.

1 Introduction

A variety of methods exists for the visualization of tensor fields. In medical imaging applications, for example, tensor data is most commonly visualized via pseudo–colored slices. In the context of diffusion tensor (DT) imaging, the color of a pixel typically depends on the directional information and fractional anisotropy (FA) of a tensor. Due to the human perception of the color space, this method does not provide full coverage of all tensor information. Tensor glyphs are another simple method, widely used especially in engineering that is capable of displaying all intrinsic information of the tensor. However, this visualization is only informative when it is not obscured, i.e., all glyphs can be seen properly. Other methods use the continuous properties of interpolated fields to display structures, such as texture–based methods like line integral convolution (LIC) [1]. While LIC itself can only provide a single directional information without scaling attributes, enhancements of LIC to tensors such as HyperLIC [2] and metric interpretations of tensor fields [3] overcome this problem. Hyperstreamlines [4] show line features in the data and can be combined with additional attributes, but when applied to 2D slices of 3D tensor data, they provide incorrect information when integration is restricted to a plane. Nevertheless, integration–based methods provide a continuous view of the data.

In contrast to vector field visualization, where glyph placement tries to minimize the number of glyphs that describe the vector field [5], for medical tensor fields, a full coverage of the field is important as features, such as tumor, may be small. This is partly due to boundaries playing an important role in medical imaging data sets and behavior along these boundaries is of interest, and partly because noise changes the data locally. Because of the noise, clustering the data into areas of similar behavior does not work here. Kindlmann and Westin [6] combined this continuous nature of the field as done by LIC or hyperstreamlines with the discrete sampling of glyphs by densely packing the glyphs to strengthen the continuity of the field. The major improvement of this method compared to discrete sampling of the glyphs is that glyphs no longer overlap and visual artifacts induced by the regular sampling of glyphs are reduced. In contrast to glyph–based vector field representations where a reduction of glyphs is the most important method to reduce visual clutter, a dense packing is enforced in measured tensor data to not overlook small features.

We use Kindlmann and Westin's "glyph packing" as a basis for our algorithm but have enhanced his method. Our main contributions are

– the use of a different, parameter–less acceleration structure that does not require good selection of bin sizes like the structure proposed before,
– an improved initial placement of glyphs to reduce overall computation time,
– a re–usable acceleration structure and placement for interactive manipulation of the planes the glyphs are drawn in,
– the exchange of parameters by self–explanatory ones that directly influence the visualization in an intuitive way, and
– an implementation where the glyphs follow a pointer such that a plane in the data set or a region of interest can be chosen interactively by the user while the algorithm provides feedback at interactive frame rates.

2 Glyph Packing

Kindlmann and Westin introduced glyph packing as an iterative approach to optimize the placement of tensor glyphs. Their method uses a particle model where each particle induces a potential field and therefore, particles attract or repel each other, depending on their distance measured in a metric space distorted by the tensor shape. The movement of particles is described by the differential equation

$$\sum_{a \neq b} f_{ab} - C_{drag}\frac{d\mathbf{p}}{dt} + C_{ext} f_{ext} = \frac{d^2\mathbf{p}}{dt^2}. \tag{1}$$

C_{drag} and C_{ext} are scaling parameters, f_{ab} are the inter–particle forces, and $\mathbf{p}$ is the location of the particle a. These forces are illustrated in Fig. 1. The drag force prevents the field from oscillating while the external force f_{ext} forces some additional layout parameters. Kindlmann and Westin use, e.g., $f_{ext} = \nabla M(\mathbf{p})$ where M is a scalar mask on the data to prevent the glyphs from leaving the domain of the data set.

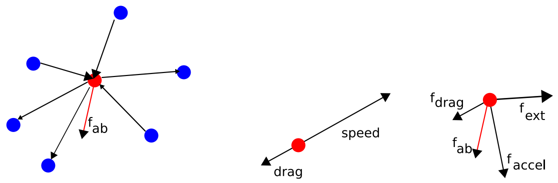


Fig. 1. Illustration of the particle interactions resulting in the force f_{ab} (left) the drag force f_{drag} (middle) and the resulting acceleration f_{accel} (right) from Eq. 1. The forces are computed for a glyph located at the red point. The blue points are positions of glyphs that influence the current point, resulting in an interparticle force. The drag force (middle) always antagonizes the current speed causing fast particles to slow down which prevents oscillation. Note that the particle's speed is the first derivative and the drag force influences the second derivative. Finally, all forces are summed up resulting in an acceleration on the particle (right).

2.1 Computing the Initial Distribution

When analyzing the force function of Kindlmann and Westin, there is a distance of strong repulsion, followed by a strong attraction that decays with increasing distance. Therefore, glyphs that are farther away have less influence than nearby glyphs. In addition, glyphs that have almost the optimal distance to each other have strong forces for keeping that distance fixed. If there is a hole in the data where no glyph is located while the glyphs outside are aligned at an almost optimal distance, there are only weak forces that try to close that hole. This example shows how crucial good initial seeding is for the algorithm. An example of the described effect is shown in Fig. 2. The initial sampling is a uniformly distributed random placement but obviously the same effect can occur when Neumann's rejection method is applied as done by Kindlmann and Westin. To prevent these cases, we compute an improved initial distribution compared to the one proposed by Kindlmann and Westin by using one of the two variants of the following approach, depending on the size of glyphs relative to the cell size of the original data set. The choice of the size of glyphs partly depends on the available screen resolution. To avoid visual clutter, glyphs have to be large

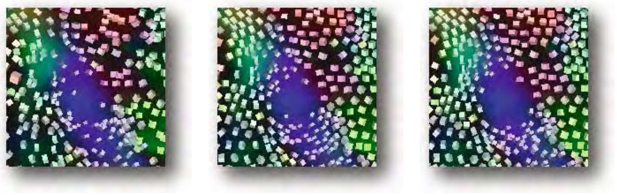


Fig. 2. A worst–case scenario for the initial seeding: The randomly placed initial glyphs (left) do not fill the area completely, but leave a hole at one point. Because of the distance across the hole, there are only week forces trying to close that hole, while there are strong forces to stabilize the layout around that hole. The holes stay stable throughout the optimization process (middle and right image).

enough to distinguish them and identify their properties, e.g., different scaling along the eigen–directions. On the other hand, they have to be small enough to keep a certain amount of glyphs on the screen to provide the interpolation like effect of dense packing and to provide information on the behavior around the glyph. Therefore, when zooming into one area, there have to be more glyphs per cell than when looking at the data from far away. Considering most commonly used medical data sets, cell size is fixed by the scanner's resolution where every slice of the data set provides a uniform, rectilinear grid containing 128×128 or 256×256 tensor values. When inspecting the data set in a full–screen mode, we found that 1000 glyphs for areas of interest and up to 65000 glyphs for whole data sets are enough for providing all features of dense packing and still avoiding visual clutter. Therefore, less than one glyph per cell needs to be drawn. When inspecting certain areas more closely – especially in printing media, we end up with areas of about 10×10 cells, but still want to draw the same amount of glyphs. In this case more than ten glyphs on the average gather in one cell. This is taken into account in our second sampling approach, where stratified sampling is used to increase the quality of the sampling for more than one glyph per cell. Let v be a cell and let A_v be the (average) area covered by a glyph in the cell. The ratio between the cell size A_v and the estimated glyph area $p_v = \frac{A_v}{a_v}$ provides an estimate of the number of glyphs in the voxel. Obviously, we overestimate the number of glyphs, but this is not a problem. By summing up p_v for all cells, i.e., $P = \sum_v p_v$, we get an estimate P of the number of glyphs we need to cover the whole data set. Furthermore, the probability for a new glyph to be added in cell v will be p_v/P. We then can use the inversion method [7] to get a mapping from a uniform distribution to the distribution of glyphs and estimate the initial placement of glyphs using the following algorithm:

```
Algorithm: Initial Glyph Placement

A[] = cellSizes();
a[v] = estimated size of glyph in cell v
p[v] = A[v]/a[v]
// integrate density
for (v = 1; v < #cells; ++v)
    p[v] = p[v-1]+p[v]
P = p[#cells-1]
for (v = 1; v < #cells; ++v)
    p[v] = p[v]/P
for (i=0; i < P; ++i)    // place P random glyphs
    d = random()         // in [0..1)
    // binary search for cell with p[v-1] <= d < p[v] where p[0]=0
    i = findCellFor(d)
    place glyph randomly in cell i
```

For cells outside the data set or in areas where no glyphs should be drawn, p_v can be set to zero to avoid initial seeding of glyphs. The more glyphs there have to be seeded in a single voxel, the more important becomes a proper placement within the cell. As described before, high–quality close–up views of the data set require many glyphs seeded in a single cell. We address this by placing glyphs using stratified sampling [8], i.e., we sample the cells independently to guarantee a better distribution within the cells: When there are voxels where it is likely that no glyph is placed, the shown algorithm is used to estimate the number of glyphs placed in every voxel. When we already know, by checking, e.g., the values of p_v for larger than one everywhere, that more than one glyphs will likely be placed in every voxel, we use a rounded value of p_v as the number of glyphs. Now every voxel is filled with a uniform distribution of this amount of glyphs where the location of the glyphs are obtained from a list of strata.These strata can be pre–computed sets of points, e.g., following a Poisson–disc distribution [8] or other randomly created samples of points that have been checked to have low energy between neighboring points for the isotropic case. To avoid regular patterns during the first steps of the optimization, we implemented interleaved sampling [9] using a set of strata and access different sets of points through a hash map using the cell index as key. Instead of using pre-computed strata, pseudo–random sequences can be used and applied with different seeds. Good pseudo–random sequences automatically fill the holes in–between points and therefore can be used incrementally, i.e., if the resolution changes, more points can be added using the same sequence.

2.2 Modified Force Function

While Kindlmann and Westin use a force function based on the ellipsoidal tensor model, we propose the use of an alternative function to improve the packing for different types of glyph representations. We describe the force on a particle directly from its scaled distance to its neighboring glyphs, i.e.:

$$y_{ab} = p_a - p_b \tag{2}$$

$$d_{opt}(a,b) = g_a(-y) + g_b(y) + c_{offset} \tag{3}$$

$$f_{ab} = \phi' \left(\frac{y_{ab}}{d_{opt}(a,b)} \right) . \tag{4}$$

Here, c_{offset} is a parameter describing an absolute offset between the glyphs. In most cases $c_{offset} = 0$ is a good choice, but to reduce the number of glyphs while keeping their size constant, an absolute offset may be given here. $g_a(y)$ denotes the surface function of glyph a evaluated at direction y. The sign in the argument is only important if the glyphs would be asymmetric which is not the case in medical data. A simple example of g for ellipsoidal glyphs, i.e., the surface described by the sphere S deformed by the tensor D by DS would be $g(y) = \frac{\alpha \|y\|}{\|D^{-1}y\|}$ which is the distance from the glyph's center to its surface measured along direction y. In contrast to the previous paper [6], in our version the function g can be chosen arbitrarily, therefore glyphs can be painted as boxes, superquadrics or ellipsoids and the force function will always try to avoid glyphs overlapping which is not the case in the previous approach. If g is chosen to be the projection of the glyph on the plane we can overcome the problem of overlapping glyphs in 2D slices that is mentioned in Kindlmann and Westin's results section. ϕ denotes the force function where ϕ', i.e., its derivative, is chosen to be

$$\phi'(r) = \begin{cases} r-1 & 0 < r < 1 \\ (r-1)(1+\gamma-r)^2/\gamma^2 & 1 \leq r \leq 1+\gamma \\ 0 & r > 1+\gamma \end{cases} , \tag{5}$$

where the optimum would be the zero–crossing at $r = 1$. All glyphs farther away than $r = 1 + \gamma$ do not influence this glyph.

2.3 Acceleration Using a Delaunay Triangulation

Because glyphs that are farther away in the metric distance have no influence on each other, computation of distances between them should be avoided. We do this by introducing a Delaunay triangulation of our sample points that is updated after moving the points. Building such a triangulation for the relevant amount of glyphs is fast. Timing can be found in Table 1. In general, between two steps, the grid does not change much. Therefore, the computational cost can be reduced by reusing the existing data structure which is only useful when a lot of glyphs are drawn and the computational complexity of the Delaunay triangulation surpasses the time of the force computation. The advantange of this data structure is that searches for neighboring points can be performed easily. A breadth–first–search can be performed to get the closest points in terms of edge distance. Usually, these points are the ones with the largest influence on the vertex. While using neighbors with a two edge distance seems to have slightly better results, direct edge neighbors are enough to make our algorithm converge.

2.4 User Parameters

Our implementation has two modes: First, it is possible to distribute a certain amount of glyphs in the data set and let the algorithm spread them in a large enough area, and second, one can select an area, where glyphs should be placed and the algorithm selects the number of glyphs needed by itself. This is an advantage to the approach previously presented, as in both cases, the user only has to select visual properties while the previous approach only lets you select the number of glyphs and the parameters for the forces, but does not provide a reasonable mean to compute one parameter from the others.

2.5 Addition and Removal of Glyphs

If a certain area of the data set needs to be filled with glyphs, and the pre–estimated amount of glyphs is not enough to fill this area, we need to add glyphs. A glyph is added when there is an area, where the distance between neighboring glyphs is large. These areas can be found by looking for triangles that have at least two long edges. In contrast, in areas where too many glyphs are present, glyphs need to be removed. We mark glyphs for removal that only have high repelling forces on their neighbors. As in dense areas, we do not want to remove all glyphs at once, we remove a certain set of glyphs randomly. Re-evaluation of the edge forces in this step provides a hint of the quality after removing the glyphs which may be used to improve the selection.

2.6 Adding Group Movement for Interactive Glyph Clouds

In addition to the mode presented in 2.4, an interactive mode is supported in our implementation, where a set of glyphs is placed around a pointing device and will follow the pointer loosely. Instead of having fixed positions related to the pointing device, the glyphs will be in motion to reorganize for the best visualization. Here, the user selects the number of glyphs to have displayed. We initially place the glyphs randomly around the cursor as described before. Then the grid is optimized while the glyphs are already displayed, which gives the user interactive feedback. When the cursor is moved, all points are moved the same way as the cursor and the optimization process keeps on moving them relatively to each other to avoid overlapping. By fixing one glyph to the cursor, we prevent the cloud to diffuse randomly into the data set, but the cloud may become asymmetric. To prevent this, additional small forces can be applied pointing to the cursor.

2.7 The Integration Step

The actual optimization is done in an Euler iteration [10] where a stepsize of 0.2 has empirically proven to be appropriate for all data sets for which we have tested our method.

```
Algorithm: Iteration

while not converged or maximal stepsize reached
    sample tensor values at particle positions
    compute all particle forces
    update particle speed
    move particles
    recompute Delaunay triangulation
    if( change in number of glyphs is allowed by user )
        check grid for triangles with large distances and
          attracting forces and place glyphs in those cells
        check grid for vertices with large repelling forces on all
           edges and mark these vertices for removal
        remove random subset of these vertices
```

3 Evaluation and Results

We implemented the proposed algorithms in our visualization system and used CGAL [11] for computing the Delaunay triangulation. We applied our algorithm to data acquired with a three–Tesla Siemens scanner on a healthy volunteer.

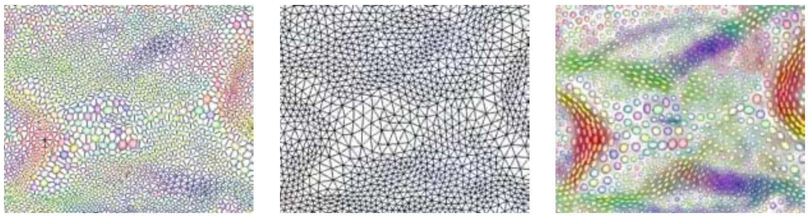


Fig. 3. Result of glyph packing of 2000 glyphs with grid (left), Delaunay triangulation only (middle) and glyphs shown on a color–mapped slice (right)

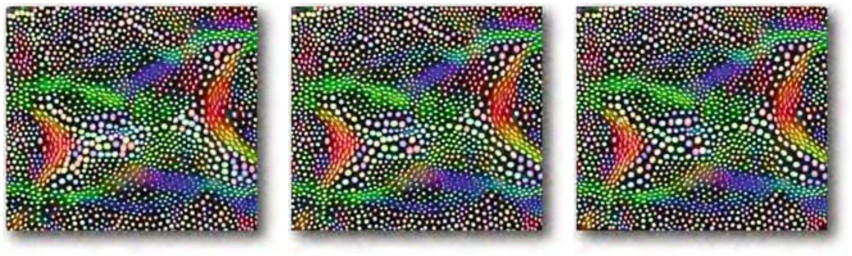


Fig. 4. Different steps of the iterative algorithm using 2000 glyphs. An initial random placements of seeds and steps 20 and 30 are presented from top left to bottom right.

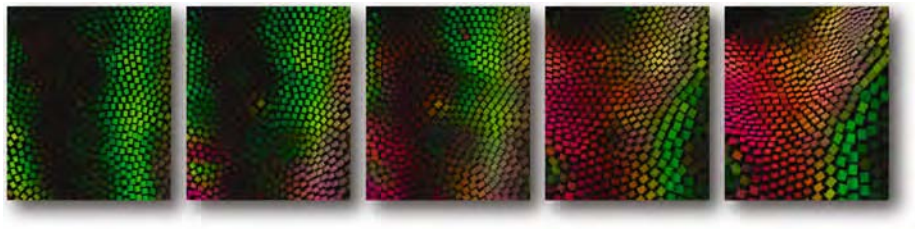


Fig. 5. An interactive sweep through a data set. The glyphs realign at interactive frame rates to prevent overlapping.

60 diffusion weighted images were acquired using three times averaging and 21 baseline images. The data has been converted to second order tensors using linear least squares fit [12]. Data acquisition takes about 20 minutes with an in–slice resolution of 128×128 voxel and 72 slices on a $1.7 \times 1.7 \times 1.7mm^3$ grid.

Fig. 3 shows the Delaunay triangulation used for neighborhood calculation, the resulting packing of glyphs and an overlay on a pseudo–colored slice. While the grid only shows direct neighbors, all neighbors having an edge distance of two are taken into account in the calculation to improve stability when edges of the triangulation flip. The evolution from the initial seeding throughout the first 30 steps is shown in Fig. 4. The energy function between glyphs decreases

Table 1. Timings of different configurations as absolute time, time for constructing the Delaunay triangulation in every step, time for computing the forces in every step and time for interpolation of the tensor values in every step. Cells marked with a dash are below the threshold of 0.01 seconds. The glyph packing usually becomes stable after several hundred steps, more than one thousand steps have never been required in our experiments. The detailed timings show that most of the time is spend in computing the inter-particle forces.

Glyphs	Iterations	Depth	Time[s]	Delaunay[s/step]	Force[s/step]	Interpol.[s/step]
100	10000	2	15	-	-	-
200	10000	2	35	-	-	-
400	10000	2	80	-	-	-
1000	100	2	2	-	0.03	-
2000	100	2	5	-	0.06	-
4000	100	2	15	-	0.14	-
10000	100	2	55	0.08	0.46	0.03
10000	10	2	5	0.02	0.21	0.02
20000	10	2	17	0.34	1.44	0.06
40000	10	2	61	1.36	4.4	0.1

tremendously during the first steps which results in an almost stable packing after about 15 steps. Fig. 5 shows a sequence of images out of an interactive sweep from the singuli to the corpus callosum.

We performed timing experiments on different scenarios in a single threaded environment. It turns out that up to 8000 glyphs can be optimized using the interactive approach on a desktop PC. For complete slices, a calculation time of about one minute is needed to get first results but up to four minutes are required until they become stable. A comparison of times can be found in Table 1. We used 10000 iterations for small amounts of glyphs to get comparable times, but far less than 100 iterations are required to obtain stable results there.

4 Conclusions

The main contribution of this paper is the development of a modified glyph packing algorithm that overcomes several drawbacks we found in the algorithm presented by Kindlmann and Westin [6]. First, we introduced a parameter–less acceleration structure that has small computational complexity and allows interactive tensor visualization using glyph placement. Second, we implemented an improved initial seeding that reduces the number of steps required by the iterative optimization and makes the algorithm more likely to converge to visually good results.To provide the user, i.e., the radiologist or surgeon, with an easy to use user interface usable in surgery, we reduced the number of parameters to two, namely the scaling of the glyphs and the distance between neighboring glyphs. Furthermore, we provided an interactive user interface where areas of interest can be selected interactively using a pointer, e.g., the mouse or a tracked 3D pointing device as used in surgery, and the glyphs automatically align around this pointer in a way that reduces visual clutter. Because our approach does not require any pre–computation or pre–processing, it can be used as an interactive probe to inspect areas of interest and can be used supplementary to existing visualization techniques.

Acknowledgments

We thank our cooperation partners from the Max Plack Institute for Human Cognitive and Brain Science in Leipzig, especially Harald E. Möller, Marc Tittgemeyer, Alfred Anwander and Thomas Knösche for discussion, evaluation of our results and for providing the data sets. We thank Christian Heine for his ideas for optimizing the algorithm.

References

1. Cabral, B., Leedom, L.C.: Imaging Vector Fields Using Line Integral Convolution. In: SIGGRAPH 1993. Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, pp. 263–270. ACM Press, New York (1993)
2. Zheng, X., Pang, A.: HyperLIC. In: Proceedings of IEEE Visualization 2003, pp. 249–256. IEEE Computer Society Press, Los Alamitos (2003)
3. Hotz, I., Feng, L., Hagen, H., Hamann, B., Joy, K.I.: Tensor field visualization using a metric interpretation. In: Weickert, J., Hagen, H. (eds.) Visualization and Processing of Tensor Fields, pp. 269–281. Springer, Heidelberg (2006)
4. Delmarcelle, T., Hesselink, L.: Visualizing second-order tensor fields with hyperstreamlines. IEEE Computer Graphics and Application 13(4), 25–33 (1993)
5. Griebel, M., Preusser, T., Rumpf, M., Schweitzer, M.A., Telea, A.: Flow field clustering via algebraic multigrid. In: Rushmeier, H., Turk, G., van Wijk, J.J. (eds.) Proceedings of IEEE Visualization 2004, pp. 35–42. IEEE Computer Society Press, Los Alamitos (2004)
6. Kindlmann, G., Westin, C.F.: Diffusion tensor visualization with glyph packing. In: Gröller, E., Pang, A., Silva, C.T., Stasko, J., van Wijk, J. (eds.) Proceedings of IEEE Visualization'06, pp. 1329–1335. IEEE Computer Society Press, Los Alamitos (2006)
7. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. ninth Dover printing, tenth GPO printing edn. Dover, New York (1964)
8. Sobol, I.M., Messer, R.: Monte Carlo Method (Popular Lectures in Mathematics). University of Chicago Press (1975)
9. Keller, A., Heidrich, W.: Interleaved sampling. Rendering Techniques, 269–276 (2001)
10. Bronstein, I., Semendjajew, K., Musiol, G.: Mühlig, H.: Taschenbuch der Mathematik — 5., überarbeitete und erweiterte Auflage. Verlag Harri Deutsch, Thun und Frankfurt am Main (2001)
11. The CGAL Consortium: CGAL, Computational Geometry Algorithms Library (2007), `http://www.cgal.org`
12. Basser, P., Mattiello, J., LeBihan, D.: Estimation of the effective self–diffusion tensor from the NMR spin echo. Journal of Magnetic Resonance 3, 247–254 (1994)

Tensor Lines in Tensor Fields of Arbitrary Order

Mario Hlawitschka[1], Gerik Scheuermann[1], Alfred Anwander[2], Thomas Knösche[2], Marc Tittgemeyer[3], and Bernd Hamann[4]

[1] University of Leipzig, Germany
[2] Max Planck Institute for Human Cognitive and Brain Science, Leipzig, Germany
[3] Max Planck Institute for Neurological Research, Cologne, Germany
[4] Institute for Data Analysis and Visualization (IDAV) and Department of Computer Science, University of California, Davis, USA

Abstract. This paper presents a method to reduce time complexity of the computation of higher–order tensor lines. The method can be applied to higher–order tensors and the spherical harmonics representation, both widely used in medical imaging. It is based on a gradient descend technique and integrates well into fiber tracking algorithms. Furthermore, the method improves the angular resolution in contrast to discrete sampling methods which is especially important to tractography, since there, small errors accumulate fast and make the result unusable. Our implementation does not interpolate derived directions but works directly on the interpolated tensor information. The specific contribution of this paper is a fast algorithm for tracking lines tensor fields of arbitrary order that increases angular resolution compared to previous approaches.

1 Introduction

Tensor data have a long history in engineering. The studies of Basser et al. [1,2] introduced the diffusion tensor to medical imaging of the human brain. In human brain imaging applications, it is now possible to measure anisotropic diffusion of hydrogen that relates to the major neural fiber bundles *in vivo*. More recent methods focus on improving the diffusion model by using the higher angular resolution of scanners that are now available even in clinical environment due to the sustaining decrease in scanning times. While second–order diffusion tensor imaging is only meaningful in isotropic areas or in areas where a voxel contains a single fiber bundle, many voxels in human brain scans contain multiple fiber bundles [3]. To overcome these well–known limitations, multi–tensor models have been proposed that match the sum of more than one tensor in every voxel, where usually the number of fibers has to be known in advance. In contrast to this, other methods, including Frank's high angular resolution diffusion imaging (HARDI) [4,5], Tuch's q–ball imaging [6,7], Alexander's PASMRI [8] and higher–order tensor approaches by Özarslan et al. [9] do not base on a–priori knowledge. They all derive a spherical function that is used to estimate the direction of fibers. Depending of the approach, this function is described by higher–order tensors, spherical harmonics or discrete, spherical sample points.

Fiber tracking has become popular in recent years, and many groups started implementing fiber tracking for second–order tensors [10,11] or higher–order methods [3]. Most of these methods split computation of directions from the tracking itself which leads to problems in areas, where the direction changes rapidly. The only method that is in some way comparable to our approach has been presented by Weinstein et al. [12]. Their method uses an advection–diffusion progress to propagate lines through second–order tensor fields. While their major intention is to use the advection as a smoothing process to provide tracking in areas of isotropic diffusion or noisy data, our method may be interpreted by its function – but not its physical meaning – as the advection step, but uses this step to compute the exact directions that can not be found analytically. The result is a line that is tangential to the directions at every position sampled by the line which is true for Weinstein's method only in constant tensor fields.

We use methods of Özarslan and Frank and describe the function as higher–order tensors and spherical harmonics. Both representations describe the same function space and can be used interchangeable as a basis set for our algorithm. While their method sets the basic idea for the storage of the local data, we show, how directions of fibers can be found efficiently and consistent tracking is performed using the higher–order information in every tracking step. Furthermore, all smooth, scalar, square integrable functions on the sphere can be approximated by this basis set, therefore, it can be easily adapted to methods such as PASMRI where the "persistent angular structure" has similar function like the orientation distribution function (ODF) computed in q–ball imaging that we use. While the method demonstrated to work on data acquired using medical imaging, the underlying theory is independent of the application and the tracking can be extended to other fields of science, e.g., mechanical engineering, where higher–order tensors are used to describe material properties.

In the next section we present the basics of higher–order tensor lines previously presented by Hlawitschka et al. [13] that are needed to understand this paper. Then we improve the previously presented method using gradient descend methods to make them computationally efficient. The resulting algorithm is capable of painting lines in about the same speed as second–order tensor methods and have a much higher angular precision than grid based methods. The increased local accuracy improves fiber tracking tremendously as drawing integral lines is highly sensitive to small local errors.

2 Higher–Order Tensor Lines

Higher–order tensor lines [13] are a generalization of streamlines and second–order tensor lines to tensors of arbitrary order. They have been defined as lines, following the maxima, minima or saddle points on the scalar surface function describing local properties. The function can be defined, for example, by the tensor T and a direction $\boldsymbol{g}$ on the unit sphere S^2 as parameter by

$$f_s(\boldsymbol{g}) = T_{i_1 \dots i_n} \boldsymbol{g}_{i_1} \cdots \boldsymbol{g}_{i_n}. \tag{1}$$

If T is a second–order tensor, i.e., $n = 2$ this is the standard case found in mechanical engineering, e.g., as derivative of vector fields as well as in second–order diffusion tensor imaging. Higher order tensors are mainly used in medical imaging which is the major motivation for this formulation. The function f_s can be represented by a spherical height–field as used later in Fig. 3, resulting in a surface that can be used as a glyph representation. For simplicity, we call the field tensor field, even though it is not limited to the tensor formulation provided in Eq. 1, but $f_s(\boldsymbol{g})$ may be written in spherical harmonics basis, too, eventually leading to the same formulation of lines.

Let $\mathbf{T}$ be a tensor field defined on $D \subseteq \mathbb{R}^3$. If the field is not degenerate there exist open, connected sets $U_m \subseteq D$ and functions $\hat{g}_m(x), x \in U_m$ where $\hat{h}(x)$ is a vector valued function indicating the direction of the m-th local maximum of f_s. We already have shown in [13] that the function can be chosen that $\hat{g}(x)$ is C^1 continuous on U_m. The function $\hat{h}$ can be interpreted as a vector field on U_m. Then a major tensor line through the point x_0 on the field defined by $\hat{h}(x)$ is the curve $c(t)$ where

$$c(0) = x_0 \qquad \text{and} \qquad \frac{\partial c(t)}{\partial t} = \hat{h}(x) \quad \text{for} \quad t \geq 0 \tag{2}$$

It is important to see that as there may exist many local maxima m at any position x there may exist a large number of different, C^1–continuous vector fields $U_{m_1} \ldots U_{m_n}$ containing the same point x which makes many different lines go through the same point x. As the structure of these neighborhoods is quite complex and there currently is no method to compute them analytically, it is not possible to extract the vector fields first, followed by painting streamlines in these vector fields. Although computing multiple vectors locally and finding the best matching direction does not provide good results because direction information between sample points has to be interpolated. Obviously, before interpolating the data, it has to be guaranteed that all vectors interpolated belong to the same locally defined vector field, which cannot be computed by only taking information of the local directions into account. Instead, it is possible to use the property of C^1 continuity of these vector fields to fill this gap. The problem of finding the locally defined field $\hat{h}$ efficiently is addressed first. Then we present an algorithm using the continuity property of the field to extract higher–order tensor lines by implicitly constructing the local vector field.

2.1 Search for Local Directions

All algorithms described in literature base on evaluation of the tensor function on a discrete set of points and, in some cases after applying a sharpening transform, search for local maxima based on these sample points. The result is the same when the grid is seen as a linearly interpolated field. Up to now, this method is state of the art in computer graphics and medical visualization and has been used in many publications, among them Descoteaux et al. [14].

Although this method leads to good results and, given a reasonable amount of sample points, the directions derived are good enough for most cases, we

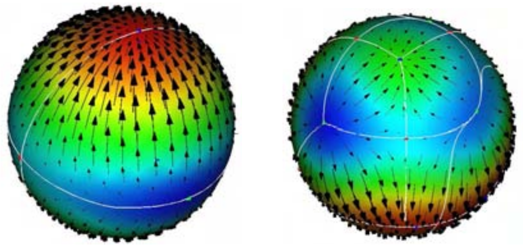


Fig. 1. Simple glyphs out of a data set. Black arrows indicate the gradient field. Vector field topology is indicated with sources (green) saddles (red) and sinks (blue). The separatrices (white lines) segment the left surface in four parts of similar behavior. A pseudo color–map indicates the underlying scalar function from deep blue (lowest value) via green to red (highest value). Left: second–order tensor. Right: more complex fourth–order tensor based on the same data set.

improve the directions using the underlying interpolation of the tensor model. Assuming, we already have a good starting point that is close to the current fiber direction, Euler's algorithm provides a fast way of finding this maximum. Because the number of steps is small and the field is well conditioned, Euler's approach provides good results. Higher–order approaches such as Runge Kutta integration would lead to more evaluations of the gradient function for small amount of steps. If the angles become larger, this is clearly a tradeoff in precision versus speed, but as the gradient field is a smooth potential vector field, this simple approach already provides good results. As the algorithm depends on the derivative of the scalar field, vector field topology [15] of the derivative of the scalar field can be used to visualize the areas of influence of every maximum as it is shown in Fig. 1.

2.2 The Algorithm

We start as all previous methods by sampling the function on a predefined grid and looking for local maxima on this grid. Instead of repeating the search at every position, the following integration algorithm is used:

```
function integrateLines( position p, initial direction ):
  for every direction found at position p
    optimize direction using Euler's algorithm
    do
        go a step in the direction
        try newdir = Euler's algorithm ( direction )
        if Euler failed: decrease step size and revert last step
        else
            update position
            update direction = newdir
        check directional change and increase step size if possible
    while step size > eps and position still inside grid
```

In contrast to most algorithms where the directions are pre–computed and directions are interpolated, our algorithm computes the directions on–the–fly. Therefore, we are able to use the feedback between finding the directions and the line integrator to change the step size of the integrator as shown in Fig. 2. A smaller step size of the integrator ensures that the inner Euler algorithm finds the maximum in less time but on the other hand increases the number of steps of the integrator. In our tests, we found that this parameter is not critical for the algorithm, but we kept the step size small to ensure that we find the next direction in at most ten steps. Using adaptive step size further reduced the number of steps in the inner loop and finally improved the precision of the direction. Because of the smoothness of the glyphs, the maximal step size can easily be adapted in a way that no significant maximum is missed and the algorithm always converges.

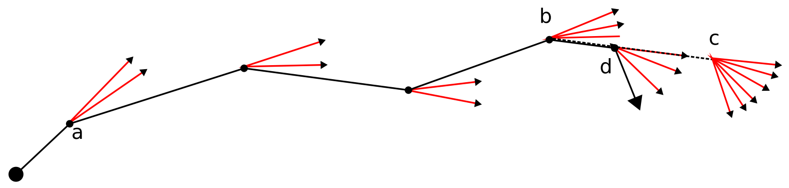


Fig. 2. Illustration of the iteration process using Euler's integration with adaptive step size. The black line shows the actual integration path. Red arrows show the iteration on the glyph for finding the new direction. The dashed line depicts a path that is tried but rejected. After the first step in point a, the step size is increased because of the angular criterion. In point b, a step (dotted line to c) is tried, but finding the direction in c failed because too many iterations are required and the angle becomes large. Therefore, the step size is reduced and the new direction is found in point d. The iteration continues with the smaller step size.

2.3 Gradient Calculation for Tensor Representations

The described algorithm needs the surface gradient of the scalar function f_s that can be calculated analytically. Let T be a symmetric tensor and let $f(g)$be a tensor–function as defined in Eq. 1. For simplicity, most of the analysis is done on the sphere $\bar{g} \in S^2 \subset \mathbb{R}^3$, $\|\bar{g}\| = 1$. The derivatives along the coordinate axes of the scalar function f are

$$\boldsymbol{v}(g) = \nabla f(g) = T_{i_1 \dots i_\ell} g_{i_1} \cdot \ldots \cdot g_{i_{\ell-1}} \tag{3}$$

The radial derivatives are given by projection of the derivative on the radial direction g

$$\begin{aligned} \boldsymbol{v_r}(g) &= \frac{\partial}{\partial r} f(g) = \langle \boldsymbol{v}(g), g \rangle \\ &= T_{i_1 \; i_\ell} g_{i_1} \cdot \ldots \cdot g_{i_{\ell-1}} \bar{g}_{i_\ell} = \|g\|^{\ell-1} f(\bar{g}). \end{aligned} \tag{4}$$

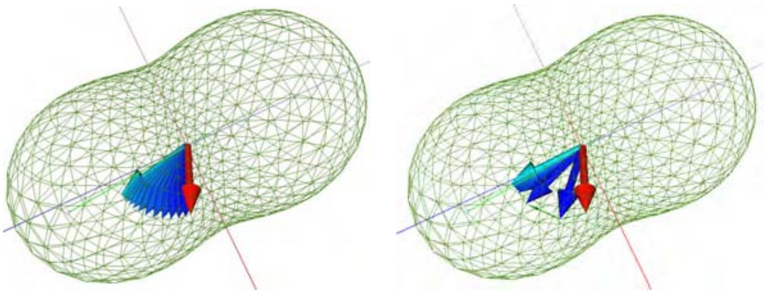


Fig. 3. Gradient descend on the surface function f_s. The red arrow indicates starting direction of the iteration. The green line indicates the correct new direction which is reached by the cyan vector, too. The green grid shows the scalar function as an offset from the center and is not used in the calculation but only shown to visualize the function. Left: fixed step size, right: Adaptive step size yields faster convergence. The angle between the starting direction and the result is exaggerated here as our algorithm would not allow that large angles.

If $\|g\| = 1$ the normalization factor vanishes and the gradient on the surface of the unit sphere can be calculated using

$$\boldsymbol{v}_s = \boldsymbol{v} - \boldsymbol{v}_{\boldsymbol{r}} \cdot \bar{g} = T_{i_1 \dots i_\ell} \bar{g}_{i_1} \cdot \ \dots \cdot \bar{g}_{i_{\ell-1}} - T_{j_1 \dots j_\ell} \bar{g}_{j_1} \cdot \ \dots \cdot \bar{g}_{j_\ell} \bar{g}_{i_\ell} \tag{5}$$

2.4 Gradient Calculation for Spherical Harmonics

An alternative representation that can be used to describe the tensor field is the spherical harmonics representation. Here, a finite set of spherical harmonics basis functions [16], i.e., a smooth set of globally defined basis functions on the sphere that can be seen as a spherical analogue to the Fourier basis in 2D is used to represent the measured data. Usually the lowest orders of the spherical harmonics are used to approximate the data, where the order defines the angular resolution. As spherical harmonics are eigenfunctions of the Laplace operator, their derivatives can be represented in the spherical harmonics basis, too and therefore are easy to compute. Using numerical evaluation, one must be aware of instabilities near the z-Axis. The instabilities arise from the unbounded Legendre associated polynomials $P_\ell^1(\cos(\theta))$. This can be circumvented by rotating the spherical harmonics by 90° before computing the derivatives. The rotation in spherical harmonics basis representation can be performed using a vector matrix multiplication where the rotation matrix is a sparse block structure matrix as described, e.g., by Ivanic [17,18]. Using this approach, derivatives can be computed on spherical harmonics representations of the spherical field with high numerical precision. The matrix to rotate the spherical harmonics depends on the degree of the spherical harmonics and is pre–computed once and is applied to every data point when needed. A 3×3 rotation matrix is then used to rotate the resulting direction back into the original frame of reference.

3 Evaluation and Results

We applied our algorithm to data acquired with a three Tesla Siemens scanner on a healthy volunteer. 60 diffusion weighted images were acquired at $b = 2500$ and six baseline images. The same gradient information is used to compute a test data set for single fiber distributions. In all cases, the data is prepared using spherical harmonics based q–ball imaging and converted to orientation distribution function using Descoteaux's [19] description of the Funk–Radon–Transform for the spherical harmonics basis. The in–slice resolution is 128×128

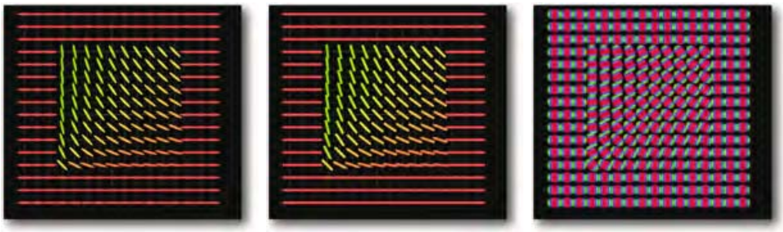


Fig. 4. From left to right: second order ellipsoidal glyphs, second order superquadric glyphs, and fourth–order glyph on q-ball data

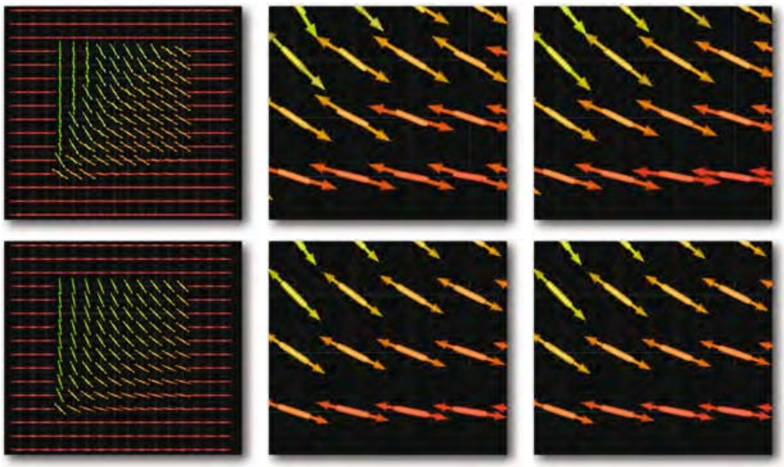


Fig. 5. A comparison of grid based search (top row) and iterative improvement (bottom row) for sampling on a grid containing 60 points (left) and zoomed into the bottom row for 240 points (middle) and 960 points (right). The points were constructed from an icosahedron subdivision. While the arrows indicate the directions found by the fourth–order q–ball imaging, superquadric glyphs are drawn for comparison as they show the true tensor direction. For 240 sample points, the deviation of the naive implementation is clearly visible, but even for 960 points it is visible in the bottom row of arrows. Iterative refinement (bottom row) gives the same results for all three cases, i.e., it is independent of the initial resolution.

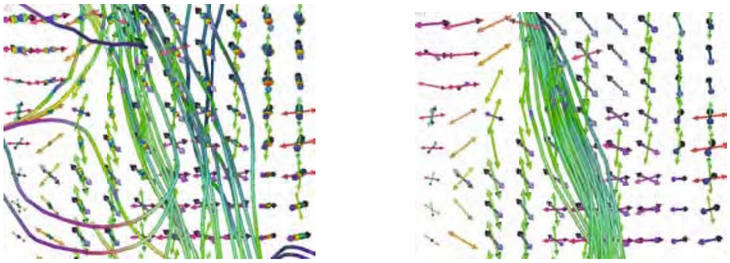


Fig. 6. Lines on second–order tensor field (left) and fourth–order data (right). The second order lines are not able to follow the right directions because of low fractional anisotropy, while higher–order lines find the dominant anterior–posterior fiber bundle in this area. The arrows indicate the grid–based directions at the sample points to get an approximate overview of the data, but are not used for the calculation.

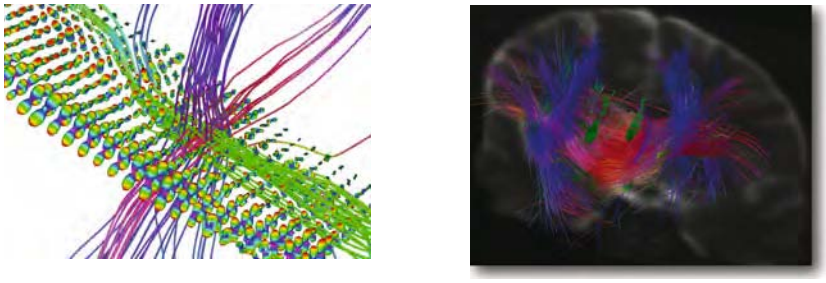


Fig. 7. Left: Crossing of three fibers out of a measured data set. Right: Fiber tracts in the same data set. The data has been acquired using a three–Tesla scanner at $b = 2500$.

voxels and 72 slices are acquired on a $1.7 \times 1.7 \times 1.7mm^3$ grid. We first test our algorithm for simple test data sets. Fig. 4 shows a simple second–order data set. The data was created from second–order tensors showing a single fiber direction and stored as raw data. For the evaluation the data was reconstructed using linear least–squares fitting to second–order tensors and q–Ball imaging on a spherical harmonics basis of order four. We chose a second–order model here, because visualization of second–order glyphs can be used to evaluate the precision of our algorithm as there, the behavior of the fiber tracts is well–known. We compared iterative refinement to brute–force search for three different resolutions. While 60 sample points on a regular spherical grid obviously lead to low angular resolution, 240 sample points are still unusable, but even for 960 sample points, our algorithm performs better. As the grid is symmetric, for grid sampling 30, 120 and 480 evaluations of the spherical harmonics are required respectively. Our algorithm usually converges in five steps to an error of approximately 1e-5 in polar angles, therefore, a fixed upper limit of ten steps can be used. In this case, the algorithm would need 50 evaluations of spherical harmonics to compute the maxima, i.e., 30 to compute a rough estimate on the grid and another twenty because computation of the derivative has the same complexity as evaluating the spherical harmonics twice and can be written as a spherical harmonics itself. This means a reduction of evaluations of spherical harmonics from 960 to 50, which improves the overall speed tremendously and still increases the angular resolution. A comparison of the two methods is shown in Fig. 5.

On the measured data set of a healthy subject, low FA values make the second–order tensor lines fail while higher–order lines are able to extract the structures (Fig. 6). Our algorithm is able to extract multiple fiber crossings as shown in Fig. 7. There crossings of two fibers can be found, e.g., in the area where the corpus callosum and the pyramidal tract meet.

4 Conclusions

We presented a method for fast calculation of higher–order tensor lines in both higher–order tensor representations as well as spherical harmonics representations of high angular resolution data. As the number of evaluations of the local data is small, the speed of our algorithm supersedes the speed of previous algorithms and is comparable to implementations of second–order tensors lines, where checks of the eigenvector orientation and direction corrections slow down the integration. We have shown that our implementation outperforms previous algorithms in both speed and precision. Especially in areas where maxima lie close together, our algorithm provides much higher resolution as sampling on a grid. Furthermore, it does not have problems with maxima artificially induced by the sampling structure as it is based on a continuous and smooth surface model. Thus, our main contribution is a fast and reliable algorithm that performs tracking of tensor lines in tensor fields of arbitrary order.

Acknowledgments

We thank Enrico Kaden and Davide Imperati for the fruitful discussion and the FAnToM-team, especially Alexander Wiebel at the University of Leipzig, Germany, Xavier Tricoche at Purdue University, West Lafayette, IN, USA, and Christoph Garth at the University of California, Davis, CA, USA for contributing code to the software framework used in this work. We thank the Max Planck Institute for Neurological Research, Cologne, Germany and in particular to Timm Wetzel for providing the diffusion–weighted MR data set.

References

1. Basser, P.J., LeBihan, D.: Fiber orientation mapping in an anisotropic medium with NMR diffusion spectroscopy. In: 11th Annual Meeting of the SMRM, Berlin, vol. 1221 (1999)
2. Basser, P., Mattiello, J., LeBihan, D.: Estimation of the effective self–diffusion tensor from the NMR spin echo. Journal of Magnetic Resonance 3, 247–254 (1994)

3. Alexander, D.C.: An introduction to computational diffusion MRI: the diffusion tensor and beyond. In: Weickert, J., Hagen, H. (eds.) Visualization and Processing of Tensor Fields, pp. 83–106. Springer, Heidelberg (2006)
4. Frank, L.R.: Anisotropy in high angular resolution diffusion-weighted MRI. Magnetic Resonance in Medicine 45, 935–939 (2001)
5. Frank, L.R.: Characterization of anisotropy in high angular resolution diffusion-weighted MRI. Magnetic Resonance in Medicine 47, 1083–1099 (2002)
6. Tuch, D.S., Reese, T.G., Wiegell, M.R., Makris, N., Belliveau, J.W., Wedeen, V.J.: High angular resolution diffusion imaging reveals intravoxel white matter. Magnetic Resonance in Medicine, 577–582 (2002)
7. Tuch, D.S.: Diffusion MRI of Complex Tissue Structure. PhD thesis, Massachusetts Institute of Technology (2002)
8. Alexander, D.C.: Persistent angular structure: new insights from diffusion magnetic resonance imaging data. Inverse Problems 19, 1031–1046 (2003)
9. Özarslan, E., Mareci, T.H.: Generalized diffusion tensor imaging and analytical relationships between diffusion tensor imaging and high angular resolution diffusion imaging. Magnetic Resonance in Medicine 50, 955–965 (2003)
10. Vilanova, A., Zhang, S., Kindlmann, G., Laidlaw, D.: An introduction to visualization of diffusion tensor imaging and its applications. In: Weickert, J., Hagen, H. (eds.) Visualization and Processing of Tensor Fields, Springer–Verlag Berlin Heidelberg, pp. 121–153. Springer, Heidelberg (2006)
11. Blaas, J., Botha, C.P., Vos, F.M., Post, F.H.: Fast and reproducible fiber bundle selection in DTI visualization. In: Silva, C.T., Gröller, E., Rushmeier, H. (eds.) Proceedings of IEEE Visualization 2005, pp. 59–64. IEEE Computer Society Press, Los Alamitos (2005)
12. Weinstein, D.M., Kindlmann, G.L., Lundberg, E.C.: Tensorlines: Advection-diffusion based propagation through diffusion tensor fields. In: Proceedings of IEEE Visualization 1999, pp. 249–253. IEEE Computer Society Press, Los Alamitos (1999)
13. Hlawitschka, M., Scheuermann, G.: HOT–lines — tracking lines in higher order tensor fields. In: Silva, C.T., Gröller, E., Rushmeier, H. (eds.) Proceedings of IEEE Visualization 2005, pp. 27–34. IEEE Computer Society Press, Los Alamitos (2005)
14. Descoteaux, M., Deriche, R., Lenglet, C.: Diffusion tensor sharpening improves white matter tractography. In: SPIE Medical Imaging, San Diego, California, USA (2007)
15. Tricoche, X., Scheuermann, G.: Topological methods for tensor visualization. In: Hansen, C., Johnson, C.R. (eds.) Visualization Handbook, Elsevier, Amsterdam (2004)
16. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. ninth dover printing, tenth gpo printing edn. Dover, New York (1964)
17. Ivanic, J., Ruedenberg, K.: Rotation matrices for real spherical harmonics. Journal of Physical Chemistry 100, 6342–6347 (1996)
18. Ivanic, J., Ruedenberg, K.: Corrections of rotation matrices for real spherical harmonics. Journal of Physical Chemistry A 102, 9099 (1999)
19. Descoteaux, M., Angelino, E., Fitzgibbons, S., Deriche, R.: Regularized, fast, and robust analytical q-ball imaging. Magnetic Resonance in Medicine 58(3), 497–510 (2007)

Image Compression Using Data-Dependent Triangulations

Burkhard Lehner[1], Georg Umlauf[1], and Bernd Hamann[2]

[1] Department of Computer Science, University of Kaiserslautern, Germany
{lehner,umlauf}@informatik.uni-kl.de
[2] Institute for Data Analysis and Visualization (IDAV) and Department of Computer Science, University of California, Davis, USA
hamann@cs.ucdavis.edu

Abstract. We present a method to speed up the computation of a high-quality data-dependent triangulation approximating an image using simulated annealing by probability distributions guided by local approximation error and its variance. The triangulation encodes the image, yielding compression rates comparable to or even superior to JPEG and JPEG2000 compression.

The specific contributions of our paper are a speed-up of the simulated annealing optimization and a comparison of our approach to other image approximation and compression methods. Furthermore, we propose an adaptive vertex insertion/removal strategy and termination criteria for the simulated annealing to achieve specified approximation error bounds.

1 Introduction

Storing and transmitting images often requires large amounts of memory or high bandwidths. Good compression algorithms are necessary to reduce these bottlenecks. Lossless compression of images does not provide the necessary compression rates. Therefore, often lossy compression algorithms are used that achieve high compression rates, but cannot reproduce the image data exactly.

The approach presented in this paper uses a piecewise linear, C^0 approximation of the image by a triangulation of the image domain. Storing or transmitting this triangulation requires only little space or bandwidth. It has the advantage that affine transformations of the image can be performed very efficiently by applying the transformation to the vertices of the triangulation only. Therefore, the encoded image can easily be displayed on screens of different resolutions. Small screens of mobile phones or other hand-held devices and large screens like power walls can be used. Since the approximation is a C^0-continuous vector based format, scaling the image to a large resolution does not cause sawtooth distortions. Smoothing of the image is not necessary.

Computing a data-dependent triangulation as high-quality approximation, i.e. low approximation error, of an image is a hard problem, which can be solved using simulated annealing (SA). However, this requires a large number of local, possibly rejected modifications of the triangulation, causing SA to converge slowly. To speed it up we propose modified probability distributions for the atomic modifications guided by the local approximation error and its variance.

In Section 2 we review related work on piecewise linear approximation and image compression. We outline in Section 3 the basic notation and introduce in Section 4 the concept of SA. In Section 5 we show how this concept is used to approximate images. Here, also the concept of adapted probability distributions, so-called guides, methods to control the number of triangles adaptively, and termination conditions for SA are discussed. In Section 6, we present experimental results of our method compared to other approaches.

2 Related Work

In 3D a 2-manifold can be approximated by a triangle mesh. In [1] an initial mesh is modified optimizing vertex positions and their connectivity to minimize a global approximation error. In [2] a fine mesh is simplified by collapsing edges according to a local criterion. Both methods cannot be applied to images, since they only work on the geometry of the mesh and not on attributes such as color.

Piecewise linear approximation of a 2D scalar field induced by a triangulation requires an error norm to measure the approximation quality. Computing a low-error triangulation is usually done iteratively. *Greedy triangulations* (GT) are computed by minimizing the error in every step. Since they are relatively simple, they are fast, but may fail to find a good approximation because of local minima.

In [3,4] a *greedy refinement strategy* is used. Starting with a Delaunay triangulation vertices are inserted at pixels of maximum L^∞-error or Sobolev-error, emphasizing regions of large changes in the image. The vertices are inserted by subdividing the corresponding triangle.

A *greedy decimation strategy* works by removing vertices from a fine initial triangulation, e.g., progressive meshes [5]. Although defined for 2-manifolds, in [5] it is also applied to color images. The error is measured by the L^2-norm, and a vertex is removed by an edge collapse. The method in [6] is similar, but measures approximation quality also by the appearance of the mesh from different viewpoints. Applied to color images it yields the same method as [5].

The method presented in [7] exhausts the space of valid triangulations more thorough, finding better approximations at higher computational costs. It alternates between refinement and decimation phases. However, it can get caught in infinite loops, e.g., alternating between removal and insertion of the same vertex.

Another class of algorithms works only with *edge flips* keeping the number of vertices fixed. In [8] the error is measured by the L^2-norm. Starting with an arbitrary triangulation, iteratively a random edge is chosen and flipped, if that reduces the error. As every greedy strategy, this may lead to situations where the algorithm gets stuck in local minima. *Simulated annealing* is used in [9] to improve these results. By also allowing for edge flips that increase the error with a slowly decreasing probability, better approximations can be found. In [10] this approach is extended by operations changing the position of vertices. Specialized to images, [11] simplifies these operations and adds a greedy refinement strategy to improve the initial triangulation. Our method further improves the performance and results of these approaches.

The main focus of methods to create vector images from raster images is editability of the output and not compression. A general framework for this transformation is described in [12]. Edge detection followed by a constrained Delaunay triangulation is used to split the image into triangles of uniform color. Incident triangles of similar color are combined to polygons. It yields C^{-1} approximation of the image. In [13] also the color gradient of every triangles is estimated, and triangles with similar color gradients are combined, but only yielding a C^{-1} approximation at polygon borders.

Most image compression algorithms are associated with a file format. The GIF and PNG file formats [14] store an image using lossless compression. On natural images both formats usually reach compression ratios of only 0.5 or less. The JPEG format [14] uses a lossy compression. For encoding the image is partitioned into square blocks, transformed into the frequency domain using the discrete cosine transform (DCT), and small coefficients are dropped. The JPEG2000 format [15] does not partition the image and uses a wavelet transformation instead of the DCT. For both formats the user can select a threshold for small coefficients, trading off between file size and image quality. Because our method is also lossy, we use these two formats for comparison.

3 Notation

An image $I : \Omega \rightarrow \mathbb{R}^3$ of width w and height h maps every pixel of $\Omega = \mathbb{Z}_w \times \mathbb{Z}_h$ to a color. This color is represented by L, a and b of the CIEL*a*b* color model [16], where color differences, as they are perceived by the human eye, can be measured as the L^2 distance of two colors.

A triangulation $t = (V, E, F)$ consists of a set of vertices $V = \{v_1, \ldots, v_n\} \subset \Omega$, edges $E \subset V^2$, and triangles $F \subset V^3$. The convex hull of $M \subset \mathbb{R}^2$ is denoted by $CH(M)$. A triangulation is valid, if it covers $CH(\Omega)$. We denote by T the set of all valid triangulations t of Ω. The set of triangles incident to an edge e is $F(e) = \{f \in F | e \in f\}$, the set of triangles incident to a vertex v is $F(v) = \{f \in F | v \in f\}$, and $P(f) = \Omega \cap CH(f)$ is the set of pixels within triangle f. The approximation $A_t : \Omega \rightarrow \mathbb{R}^3$ induced by t is the piecewise linear interpolant of I at the vertices. For the approximation we measure the error per pixel $p = (x, y)$, triangle f, edge e, vertex v, and set of edges $S_e \subset E$ or vertices $S_v \subset V$

$$\Delta(p) = (I(p) - A_t(p))^2, \qquad \Delta(f) = \sum_{p \in P(f)} \Delta(p), \qquad \Delta(e) = \sum_{f \in F(e)} \Delta(f),$$

$$\Delta(v) = \sum_{f \in F(v)} \Delta(f), \qquad \Delta(S_e) = \sum_{e \in S_e} \Delta(e), \qquad \Delta(S_v) = \sum_{v \in S_v} \Delta(v).$$

The total error used to measure the global approximation quality is

$$\Delta_{\mathrm{G}} = \sqrt{\sum_{(x,y) \in \Omega} \Delta(x, y)/(h \cdot w)}.$$

4 The Principle of Simulated Annealing

Simulated annealing is a stochastic, iterative method to find the global extremum s^* of a given function $f(s) : D \to \mathbb{R}$ in a large search space D. It is based on the following approach: Starting with an arbitrary setting $s_0 \in D$, slightly modify s_0 to get $s'_0 \in D$. Using a probability distribution $p_{\text{accept}} : \mathbb{R}^2 \times \mathbb{N} \to [0,1]$, s'_0 is accepted with probability $p_{\text{accept}}(f(s_0), f(s'_0), 0)$, i.e., $s_1 = s'_0$, otherwise it is rejected, i.e., $s_1 = s_0$. Iterating the process yields a sequence of settings s_i which converges to the global extremum s^* under certain assumptions on p_{accept} [17],

$$p_{\text{accept}}(f(a), f(b), i) = \begin{cases} \exp\left((f(a) - f(b))/\tau_i\right) & \text{for } f(b) > f(a) \\ 1 & \text{for } b \leq a \end{cases}, \qquad (1)$$

where $\tau_i = \tau_0 \tau_{\text{base}}^i$ is a temperature initialized with τ_0, that is decreased to zero by a factor $\tau_{\text{base}} \in]0,1[$ in every iteration. Since settings that increase f might be accepted depending on (1), the sequence of s_i can escape local minima. The temperatures τ_0 and τ_{base} define the annealing schedule. If τ_i decreases too fast, the sequence can get stuck in a local minimum. If it decreases too slowly, the sequence converges to a better local minimum using more iterations. It can be shown that for the right annealing schedule the probability to find the global minimum converges to one, usually requiring a large number of iterations [17].

5 Simulated Annealing for Image Approximation

5.1 Basic Simulated Annealing (BSA)

The basic approach to find an approximation A_t for an image I is based on [10,11]. To find t minimize $\Delta_{\text{G}}(t)$ for all $t \in T$ using (1), where τ_0 and τ_{base} are user-defined parameters. There are three modifications to generate t'_i from t_i:

Edge Flips. change mesh connectivity: If the union of two triangles $\{v_a, v_b, v_c\}$, $\{v_b, v_c, v_d\}$ is convex, they are replaced by $\{v_a, v_b, v_d\}$, $\{v_a, v_d, v_c\}$, see Fig. 1 and [8].

Local Vertex Moves. change vertex positions locally: A vertex of t_i is assigned a new position in its vicinity while changing the local connectivity as little as possible without generating degenerate triangles, see Fig. 2 and [10].

Global Vertex Moves. change vertex distribution globally in two steps:

a) A vertex is removed from t_i and the resulting hole is triangulated.

b) A new vertex $v \notin t_i$ is inserted to t_i either by splitting the triangle that contains v, or by splitting the two adjacent triangles into two triangles, if v lies on an edge of t_i, and a locally optimal data-dependent triangulation for v using the method in [8] is applied.

Then, one iteration of BSA consists of five steps:

1. Select one modification at random, with probability p_f for an edge flip, p_l for a local vertex move, and $p_g = 1 - p_f - p_l$ for a global vertex move.

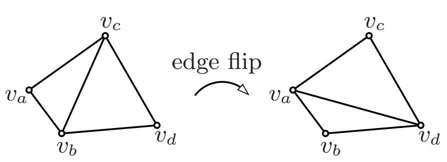


Fig. 1. An edge flip

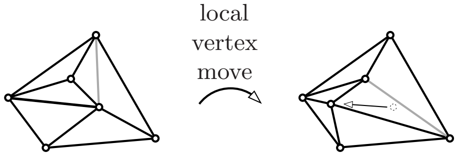


Fig. 2. A local vertex move. The gray edge is flipped to prevent a degenerate triangle.

2. Select the edge for the edge flip or the vertex for the local/global vertex at random, where every edge or vertex has the same uniform probability.
3. If the selected modification is a vertex move, select the new vertex position at random, where every new position has the same uniform probability.
4. Generate t'_i from t_i using the selected modification as determined in 1.-3.
5. The modified triangulation t'_i is accepted or rejected using (1), i.e.,

$$t_{i+1} = \begin{cases} t'_i & \text{with probability } p_{\text{accept}}(\Delta_{\text{G}}(t_i), \Delta_{\text{G}}(t'_i), i), \\ t_i & \text{if } t'_i \text{ is rejected.} \end{cases}$$

The initial triangulation t_0 is computed similar to that in [11]: Starting with the four corner vertices, we repeatedly insert a new vertex v into $f \in F$ with largest error $\Delta(f)$ at its error barycenter, but instead of using a Delaunay criterion as [11], we construct a locally optimal data-dependent triangulation for v using the method in [8].

5.2 Guided Simulated Annealing (GSA)

The BSA finds a high-quality data-dependent approximation to an image with a large number of iterations. Therefore, we adapt the probability distributions, so-called guides, to speed up BSA: In steps 2. and 3. we select edges, vertices and new positions in regions of large approximation error with a higher probability. We use edge/vertex guides for the selection of edges and vertices and position guides for the selection of new vertex positions.

Edge Guide. Denote by $E_{\text{flippable}}$ the set of edges incident to two triangles forming a convex quadrilateral. To prefer edges in regions of large approximation error, $e \in E_{\text{flippable}}$ is selected with probability

$$p_{\text{flip}}(e) = \Delta(e)/\Delta(E_{\text{flippable}}). \tag{2}$$

Local Vertex Guide. Denote by V_{movable} the set of all vertices except the four corners of an image. To prefer vertices in regions of large approximation error, $v \in V_{\text{movable}}$ is selected with probability

$$p_{\text{local}}(v) = \Delta(v)/\Delta(V_{\text{movable}}). \tag{3}$$

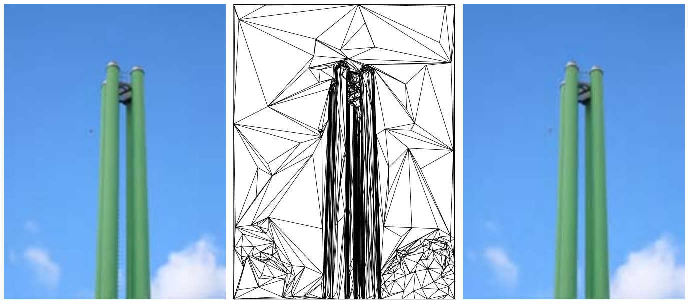

(a) Original image. (b) Triangulation. (c) Approximation.

Fig. 3. Original image (550 kB) (a), the triangulation (711 vertices) (b), the approximate image (~4 kB) (c)

Global Vertex Guide. For the global move we have to remove an "unimportant" vertex changing the approximation error only a little. For this we use the variance $\mathrm{Var}(v)$ of the gradient of A_t for the triangles incident to a vertex v. Within each triangle f of A_{t_i} the color gradient $g(f) = \nabla (A_{t_i})|_f$ is constant. So, the mean gradient and its variance for v are defined as

$$\bar{g}(v) = \frac{1}{|F(v)|} \sum_{f \in F(v)} g(f) \quad \text{and} \quad \mathrm{Var}(v) = \frac{1}{|F(v)|} \sum_{f \in F(v)} (g(f) - \bar{g}(v))^2.$$

If $\mathrm{Var}(v)$ is small, the color gradients of the triangles incident to v are similar. Removing v and triangulating the hole leads to new triangles with a similar color gradient, changing approximation quality only a little. So, we use $w(v) = (\mathrm{Var}(v)+\varepsilon)^{-1}, \varepsilon > 0$, to guide the removal and select $v \in V_{\mathrm{movable}}$ for a global move with probability

$$p_{\mathrm{global}}(v) = w(v) \Big/ \sum_{v \in V_{\mathrm{movable}}} w(v). \tag{4}$$

For the new vertex position we prefer a position with a large approximation error $\Delta(x,y)$. The position guides for the local and the global move only differ in the set F_{select} of possible triangles:

$$F_{\mathrm{select}} = \begin{cases} F & \text{for a global move,} \\ F(v) \cup \{f \in F | f \cap F(v) \in E\} & \text{for a local move.} \end{cases}$$

The new position is selected using the following guide:

Position Guide

a) To prefer triangles with a large approximation error, select a triangle $f_{\mathrm{select}} \in F_{\mathrm{select}}$ with probability

$$p_{\mathrm{triselect}}(f) = \Delta(f)/\Delta(F_{\mathrm{select}}). \tag{5}$$

b) To prefer positions with a large approximation error, select the new position $\omega_{\text{select}} \in f_{\text{select}}$ with probability

$$p_{\text{posselect}}(\omega) = \Delta(\omega)/\Delta(f_{\text{select}}). \tag{6}$$

5.3 Acceleration Trees

Using guides leads to less iterations to achieve highly similar approximation error, see Fig. 5. But computing the guides requires additional effort increasing the total computation time. Every guide involves computing weights for all vertices/edges/triangles from a set of candidates, see (2) – (5), and for the selection the sum of weights normalizes the probabilities. The complexity for computing these sums can be reduced by storing the weights in a binary tree, the so-called acceleration tree, and updating only those that are changed by a modification with $\mathcal{O}(\log n)$. All leaves have the same depth $\lceil \log(n) \rceil$ and contain the weights of the vertices/edges/triangles. The inner nodes contain the sum of their siblings, i.e., the sum of all leaves of the corresponding subtree. If the number of leaves is not a power of two, some inner nodes have no right sibling. The root contains the sum of all weights. Fig. 4 shows an example of an acceleration tree.

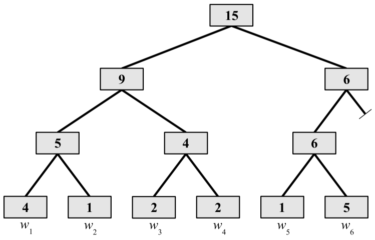


Fig. 4. An example of the acceleration tree for six vertices/edges/triangles

To select a vertex/edge/triangle according to its probability, we use a uniformly distributed random number $r \in [0, 1[$ and multiply it with the sum of weights in the root of the tree and trace it down as follows: Denote by w_{left} and w_{right} the weight of the left and right sibling of the current node. If $r < w_{\text{left}}$, we proceed with the left sibling; if $r \geq w_{\text{left}}$, we set $r = r - w_{\text{left}}$ and proceed with the right sibling. We repeat this until we reach a leaf, and select the corresponding vertex/edge/triangle. This also requires only $\mathcal{O}(\log n)$ operations.

A separate acceleration tree instance is used for each of the following guides:

Edge Guide. The leaves store $\Delta(e)$ for every $e \in E_{\text{flippable}}$.
Local Vertex Guide. The leaves store $\Delta(v)$ for every $v \in V_{\text{movable}}$.
Global Vertex Guide. The leaves store $w(v)$ for every $v \in V_{\text{movable}}$.
Position Guide. The leaves store $\Delta(f)$ for every $f \in F$ for global vertex moves.

We do not use acceleration trees for local vertex moves, because F_{select} contains only a small number of triangles and does not grow linearly with the number of triangles or vertices of the triangulation. An acceleration tree cannot be used to speed up the selection of a pixel within a triangle for the second step of the position guide. Fig. 6 shows the speed-up that is gained by acceleration trees.

5.4 Adaptive Number of Vertices

In general, the more triangles used, the better the approximation. But it depends very much on the complexity, size of details and noise of the image how many triangles are needed to achieve a user-specified error. Therefore, also the number of vertices is changed during the simulated annealing procedure.

The approximant A_t is piecewise linear and we observed that the approximation error is $O(n^{-1})$, where n is the number of vertices. Thus, there exists a constant α with $\Delta_{\mathrm{G}} \approx \frac{\alpha}{n}$ for which a given error Δ_{goal} can be achieved by an estimated number of vertices $n_{\mathrm{goal}} \approx n\,\Delta_{\mathrm{G}}/\Delta_{\mathrm{goal}}$. To reach n_{goal} vertices, in every iteration additional vertices can be inserted or removed with probability

$$p_{\mathrm{change}} = n/m \cdot |1 - \Delta_{\mathrm{G}}/\Delta_{\mathrm{goal}}|\,,$$

where m is the number of iterations to reach n_{goal} for constant p_{change}. A vertex is inserted or removed by a variant of the Global Vertex Move:

– If $\Delta_{\mathrm{G}} \geq \Delta_{\mathrm{goal}}$ a new vertex is inserted by **Global Vertex Move** b) with position guides followed by the local optimal data-dependent triangulation.
– If $\Delta_{\mathrm{G}} < \Delta_{\mathrm{goal}}$ a vertex is removed by **Global Vertex Move** a) with the global vertex guide.

Note that insertion/removal of vertices can happen in every iteration before step 1., independently of the rest of the simulated annealing. Its acceptance probability is one, since a vertex removal always increases Δ_{G}.

5.5 Termination Conditions and Image Compression

There are several termination conditions for simulated annealing focussing on run-time, memory consumption or approximation quality:

Simulation Time. The iteration stops after a predefined time. It can be used in combination with the next two conditions.

Triangulation Complexity. Changing the number of vertices adaptively, the iteration stops after a predefined number of vertices or triangles is reached.

Specified Error. The iteration stops after reaching the predefined error Δ_{goal}. If Δ_{G} differs from Δ_{goal} only by $|1 - \frac{\Delta_{\mathrm{G}}}{\Delta_{\mathrm{goal}}}| < 0.03$ for example, only a fixed number of additional iterations $m_{\mathrm{post}} = \max(1000, 0.1m)$ are done where m is the number of iterations done so far. If during these iterations Δ_{G} differs from Δ_{goal} by more than 3%, the simulation continues.

For every approximation A_t we stored coordinates, i.e. $\lceil \log_2(|\Omega|) \rceil$ bits per vertex, and colors, i.e. 24 bits per vertex, for every vertex and the connectivity of the mesh with [18], i.e. on average two bits per triangle. Since only an approximation is stored, the compression is lossy. Experiments revealed that further compression of the color and coordinate components is hardly possible, so these are stored uncompressed. Using the second termination condition enables the user to set a limit for the required memory. For the third termination condition GSA uses as many vertices and triangles as needed to achieve Δ_{goal}.

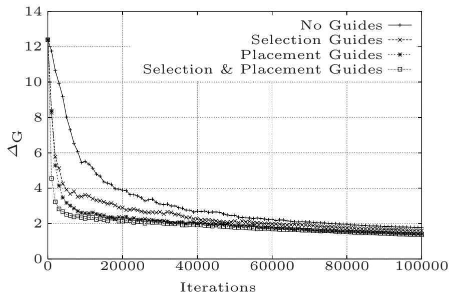


Fig. 5. Decrease of Δ_G using different combinations of guides

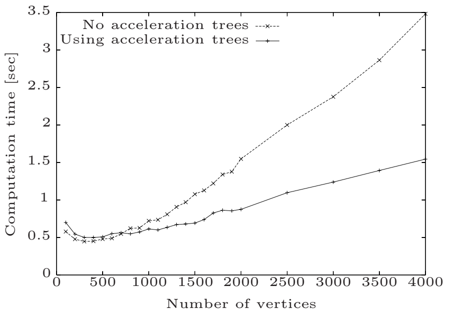


Fig. 6. Time for 500 iterations of simulated annealing for different numbers of vertices with/without acceleration trees

6 Results

All experiments were performed on a computer with an Intel Pentium 4PC, 2.66 GHz, 512 kB cache, 1 GB RAM.

Fig. 5 compares the approximation error as a function of number of iterations of the simulated annealing method for different combinations of guides. It shows that using guides does not corrupt the convergence of the simulated annealing method. The graphs show that the fastest decrease in approximation error is achieve by using all guides, and that placement guides have a higher impact than selection guides.

Fig. 6 compares the computation time of 500 iterations of GSA for Fig. 3(a) with and without acceleration trees for an increasing fixed number of vertices. It shows that for more than 700 vertices the overhead for the acceleration trees is compensated by the faster computation of the weights. For images with high complexity and many vertices the acceleration trees give a significant speed-up.

Table 1. Δ_G for Fig. 7 for different methods

	SA [11]	GSA		GT [5]	JPG	JPG 2000
Color model	RGB	RGB	Lab	Lab	Lab	Lab
Δ_G	17.26	16.45	6.00	6.16	10.38	6.81

For comparison Fig. 7 shows approximations of the Lena image (512 × 512 pixels, 786 440 bytes) using five different methods as listed in Table 1. The file size of the compressed images is ∼ 5 750 bytes. Figs. 7(b) and 7(c) were computed with 300 000 iterations, the triangulations of Figs. 7(b), 7(c), and 7(f) contain 1 000 vertices. For comparison with the results provided in [11], the GSA was also computed using the RGB color model (see Table 1). GSA produced the smallest Δ_G of the tested methods. Its approximation (Fig. 7(b)) misses some high-frequency details, but its overall impression is smooth, whereas

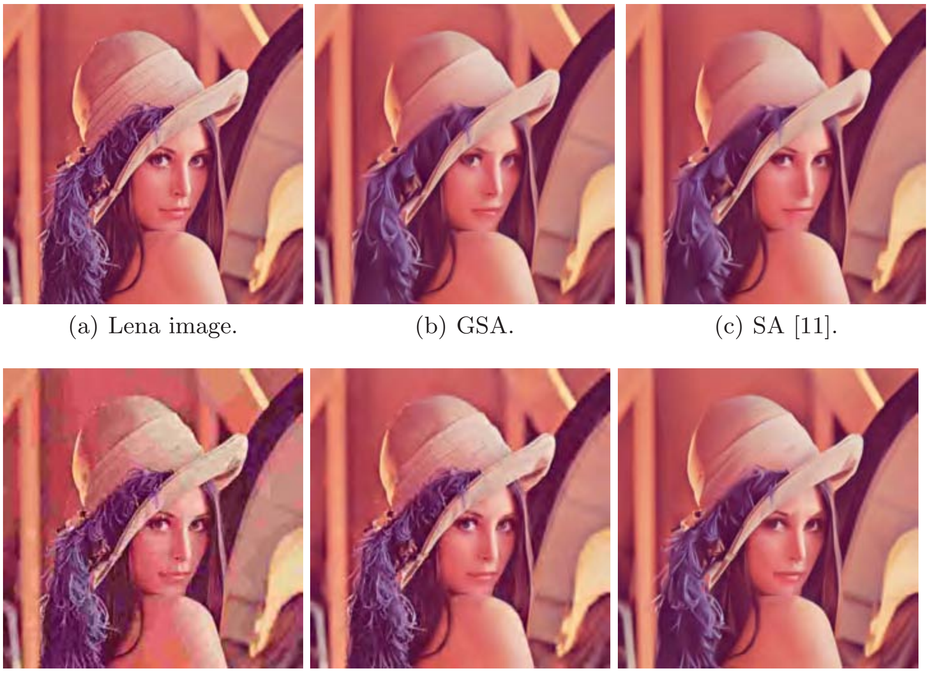

(a) Lena image. (b) GSA. (c) SA [11]. (d) JPEG. (e) JPEG2000. (f) GT [5].

Fig. 7. Original image (∼770 kB) (a), and approximations (∼5 750 bytes) using GSA (b), SA [11] (c), JPEG (d), JPEG2000 (e), and GT [5] (f), (courtesy USC SIPI)

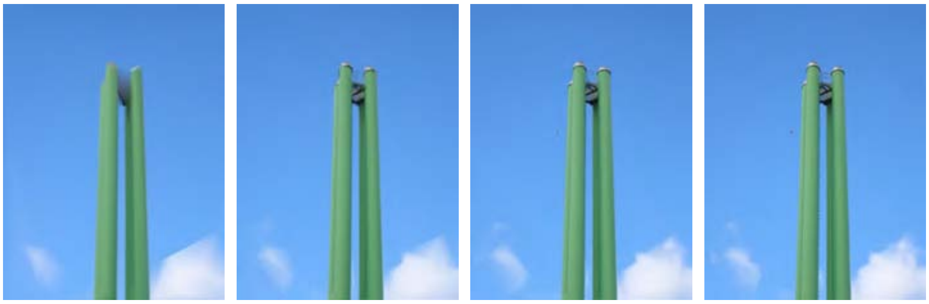

(a) $\Delta_G = 3.0$. (b) $\Delta_G = 2.0$. (c) $\Delta_G = 1.5$. (d) $\Delta_G = 1.0$.

Fig. 8. Approximations of the image shown in Fig. 3(a) with different errors Δ_G

JPEG and JPEG2000 provide some of the detail, but lead to a rough and discontinuous overall impression.

Figs. 3 and 8 show examples of the results of our compression algorithm summarized in Table 2. The original image shown in Fig. 3(a) has 384×512 pixels

Table 2. Comparison of approximations of Fig. 3(a)

Δ_G	Itera-tions	Time [sec]	Verti-ces	Size [bytes]	Bits per pixel	Compr. rate	Fig.
3.0	18 715	18.51	73	450	0.0018	1:1 130	8(a)
2.0	24 109	19.46	176	1 039	0.042	1: 568	8(b)
1.5	27 408	22.64	385	2 242	0.091	1: 263	8(c)
1.25	36 173	38.67	711	4 115	0.17	1: 143	3(c)
1.0	47 370	97.92	2 256	11 689	0.47	1: 50	8(d)

and is stored with 589 840 bytes. Fig. 3(b) shows the underlying triangulation of Fig. 3(c) for $\Delta_G = 1.25$. The number of iterations and the number of vertices increase as Δ_G decreases. Thus, best compression rates and lowest computation times are achieved for low image quality and vice versa, see Figs. 8(a) and 8(d).

7 Conclusions and Future Work

The method for the construction of a piecewise linear representation presented in this paper can be used for the construction of high-quality approximations of images. These approximations have high compression rates comparable or even superior to JPEG compression results. Considering the approach discussed in [10,11], our work extends their approach in the following ways:

1. By using guides with acceleration trees we achieve higher compression rates and approximation quality.
2. By adaptively changing the number of vertices the approximation quality can be controlled.
3. Different termination conditions allow for different optimization objectives.
4. Memory requirements for our image approximation are compared to those of JPEG and JPG2000 image compression.

There are some additional research issues we want to work on in the future. The guides can be further improved, for example, by using sharp features detected in the image as attractors for vertices. We plan to investigate to what extend the sliver triangles can cause aliasing effects when scaling the triangulation. Using spline interpolation instead of linear interpolation could improve the approximation quality. We also plan to extend the method to the compression of videos, enhancing compression rates using the coherence of successive frames.

Acknowledgments

This work was supported by the DFG IRTG 1131, "Visualization of Large and Unstructured Data Sets", University of Kaiserslautern, and NSF contract ACI 9624034 (CAREER Award) and a large ITR grant, University of Davis. We thank the members of the Visualization and Computer Graphics Research Group at

IDAV. In particular, we express our special thanks to Oliver Kreylos, for his support at the beginning of this project, and Klaus Denker for his help on the implementation of the software.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: SIGGRAPH 1993, pp. 19–26 (1993)
2. Gao, J., Zhou, M., Wang, H.: Mesh simplification with average planes for 3-d image. Systems, Man, and Cybernetics 2, 1412–1417 (2000)
3. Garland, M., Heckbert, P.: Fast polygonal approximation of terrains and height fields. Technical report, CS Department, Carnegie Mellon University (1995)
4. Schaetzl, R., Hagen, H., Barnes, J., Hamann, B., Joy, K.: Data-dependent triangulation in the plane with adaptive knot placement. In: Brunnett, G., Bieri, H., Farin, G. (eds.) Geometric Modelling, Comp. Suppl., vol. 14, pp. 199–218. Springer, Heidelberg (2001)
5. Hoppe, H.: Progressive meshes. In: SIGGRAPH 1996, pp. 99–108 (1996)
6. Lindstrom, P., Turk, G.: Image-driven simplification. ACM Trans. Graph. 19, 204–241 (2000)
7. Pedrini, H.: An improved refinement and decimation method for adaptive terrain surface approximation. In: Proceedings of WSCG, Czech Republic, pp. 103–109 (2001)
8. Dyn, N., Levin, D., Rippa, S.: Data dependent triangulations for piecewise linear interpolations. IMA J. of Numerical Analysis 10, 137–154 (1990)
9. Schumaker, L.L.: Computing optimal triangulations using simulated annealing. Computer Aided Geometric Design 10, 329–345 (1993)
10. Kreylos, O., Hamann, B.: On simulated annealing and the construction of linear spline approximations for scattered data. IEEE TVCG 7, 17–31 (2001)
11. Petrovic, V., Kuester, F.: Optimized construction of linear approximations to image data. In: Proc. 11th Pacific Conf. on Comp. Graphics and Appl., pp. 487–491 (2003)
12. Prasad, L., Skourikhine, A.: Vectorized image segmentation via trixel agglomeration. Pattern Recogn. 39, 501–514 (2006)
13. Lecot, G., Levy, B.: Ardeco: Automatic region detection and conversion. In: Eurographics Symposium on Rendering conf. proc. (2006)
14. Miano, J.: Compressed Image File Formats. JPEG, PNG, GIF, XBM, BMP. In: SIGGRAPH series, Addison-Wesley Longman, Amsterdam (1999)
15. Taubman, D.S., Marcellin, M.W.: JPEG2000: Image Compression Fundamentals, Standards and Practice. Springer, Heidelberg (2002)
16. Wyszecki, G., Stiles, W.: Color Science: Concepts and Methods, Quantitative Data and Formulae. Wiley-Interscience (1982)
17. Kirkpatrick, S., Vecchi, M.P., Jr. Gelatt, C.D.: Optimization by simulated annealing. Science Magazine, 671–680 (1983)
18. Rossignac, J.: Edgebreaker: Connectivity compression for triangle meshes. IEEE TVCG 5, 47–61 (1999)

Unsynchronized 4D Barcodes
(Coding and Decoding Time-Multiplexed 2D Colorcodes)

Tobias Langlotz and Oliver Bimber

Bauhaus-University Weimar
{Tobias.Langlotz,Oliver.Bimber}@medien.uni-weimar.de

Abstract. We present a novel technique for optical data transfer between public displays and mobile devices based on unsynchronized 4D barcodes. We assume that no direct (electromagnetic or other) connection between the devices can exist. Time-multiplexed, 2D color barcodes are displayed on screens and recorded with camera equipped mobile phones. This allows to transmit information optically between both devices. Our approach maximizes the data throughput and the robustness of the barcode recognition, while no immediate synchronization exists. Although the transfer rate is much smaller than it can be achieved with electromagnetic techniques (e.g., Bluetooth or WiFi), we envision to apply such a technique wherever no direct connection is available. 4D barcodes can, for instance, be integrated into public web-pages, movie sequences, advertisement presentations or information displays, and they encode and transmit more information than possible with single 2D or 3D barcodes.

1 Introduction and Motivation

Encoding and decoding digital information into printed two dimensional barcodes becomes more and more popular. They are used in advertisements, on business cards or e-tickets, or for referencing to web-pages as in Semapedia (www.semapedia.org). The amount of information that can be decoded robustly from a 2D barcode with ordinary mobile devices, such as mobile phones, is usually restricted to several characters only. Thus, usually IDs, URLs or simple addresses are encoded. Yet, professional industrial scanners are able to decode a much larger amount of characters (several thousands) with an acceptable reliability. In this paper we present a new kind of barcode that we refer to as *4D barcode*. It encodes data in four dimensions: width, height, color and time. Consequently, it cannot be printed on paper but is displayed on screens of mobile or spatial devices. Time-multiplexing colored 2D barcodes allows to transmit a larger amount of information robustly to off-the-shelf mobile phones without requiring an explicit synchronization (cf. figure 1(a)).

2 Related Work

A large number of applications for mobile phones exist that read and decode printed *QR-codes* [1] as a standardized black-and-white 2D barcode.

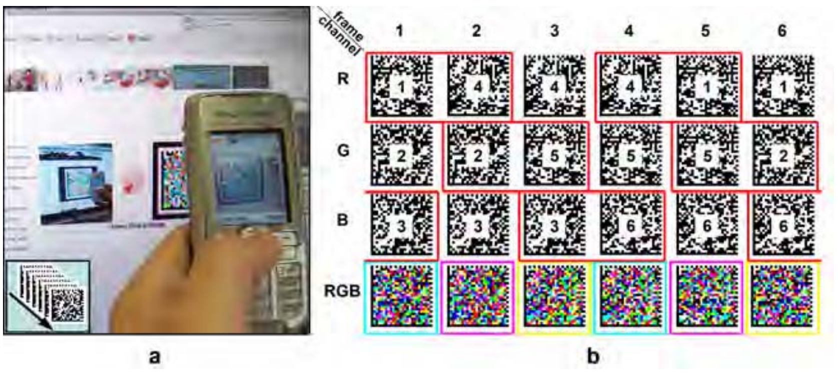


Fig. 1. 4D barcodes: (a) displaying and capturing, (b) encoding scheme for embedding 2D barcodes into a sequence of time-multiplexed 3D barcodes (barcode transitions are framed in red)

Datamatrix [2] is a similar example (yet not as common as QR-codes) and is used by applications like Semacode (www.semacode.com). As mentioned earlier, only a small amount of information can be decoded robustly with consumer camera phones - limiting QR-code or Datamatrix barcodes to encode a few characters only. Han et al. [3] propose *Colorcode*, a 3D barcode which -in addition to a 2D matrix layout- uses colored bits as third dimension. But due to its small resolution of 5x5 cells it encodes only IDs that are resolved through a central lookup service (www.colorzip.co.jp). Besides applications in advertisement, the Colorcode is used for context aware systems [4]. 2D barcodes have also been applied to realize interaction techniques with mobile phones. Rohs [5] describes a barcode named *Visual Code* that stores up to 83 bits of data. Displaying it on a screen, it is used for tracking the movement and rotation of the phone relative to the screen [6,7]. Another novel approach is presented by Scott et al. [8], who use *Spotcode* (a circular 2D barcode) for out-of-band device discovery and service selection - bypassing the standard Bluetooth in-band device discovery. This is applied by Madhavapeddy et al. [9] to also implement interaction techniques with Spotcodes that are displayed on a screen - using an online Bluetooth connection for data exchange. Similar techniques that apply displayed 2D barcodes for supporting mobile phone based interaction methods in combination with screens can be found in [10,11,12].

Besides barcodes, other possibilities for optical data transfer exist. Shen et al. [13], for example, explain how to read 7-segment digits from LCD/LED displays with camera equipped mobile phones. This system was mainly developed to support people with vision disorders by using their phones to recognize and read the digits on simple displays, such as on clocks. The system requires approximately two seconds for capturing and reading the digits on a Nokia 6630 - which is comparable to other OCR Software for mobile phones. Yet another interesting new approach for transmitting data optically is to use light sources instead of displays. In *visible light communication*, ordinary light sources are modulated with a digital signal. Approaches presented by Tanaka et al. [14] or Komine and Nakagawa [15] use white-light LEDs for illuminating a room and for transmitting time-multiplexed signals. The modulation frequency is high enough so that the transmitted signal remains invisible to the human eye. The embedded signal is received by photo diodes and is finally decoded. Using such a system, it is possible to transmit up to 100 Mbit/s and more.

3 Time-Multiplexed Colored 2D Barcodes

One possibility to enlarge the data volume that can be embedded into a 2D barcode is to increase the code matrix resolution. However, the optics used for consumer cameras set clear limitations. Consequently, this is not an option when off-the-shelf mobile phones are used. The main idea of 4D barcodes is to split

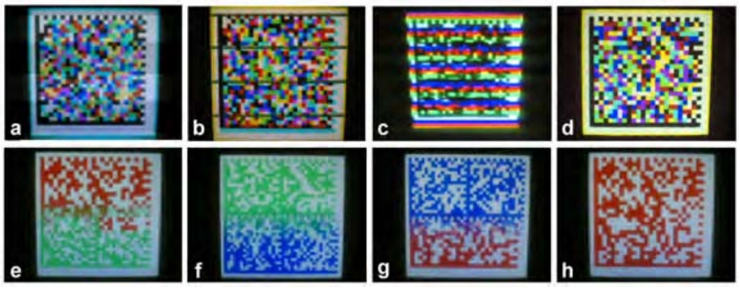


Fig. 2. Captured unsynchronized 3D barcodes from (a) 120Hz and (b) 60Hz CRT monitor, (c) DLP projector with white color wheel segment, and (d) LCD projector. Time-multiplexed R,G,B 2D barcode sequence captured form LCD monitor, (e-g) with and (h) without frame transitions.

the data into smaller chunks that are embedded into a series of decodable 3D barcodes. Thereby, the color dimension is used for increasing the robustness of the transmission. The animated 3D barcodes are displayed in an endless-loop on screens, and can be recorded by mobile camera phones. The looping duration and state is visually indicated on the display to give a feedback on how long the code sequence needs to be captured. After recording, individual barcodes are extracted, assembled and decoded on the phone to reconstruct the entire data content. Thereby, the challenge is the missing synchronization between displaying and recording. Our system is able to support LCD panels (or projectors) and Plasma screens. CRT monitors (or projectors) can only be applied if the decay rate of the utilized phosphor and the display's refresh rate ensure no full blank regions during the integration time of the camera chip. We found that fast 120Hz CRT monitors (cf. figure 2a) are sufficient, while most slow (e.g. slower than 85Hz) CRT monitors (cf. figure 2b) are not. Due to an image generation via time-multiplexing (color and gray levels), DLP-based displays (i.e., projectors or back-projected screens) are not supported (cf. figure 2c). We use Datamatrix barcodes in our prototype for encoding and decoding since decoders are freely available. Yet, it is extended to carry nested color bits. Animated GIFs are used to display the sequences of color codes. They can be easily embedded into web-pages. The following sections describe the encoding and decoding process in more detail.

3.1 Encoding

As mentioned above, the whole data set is split into smaller portions. They are encoded into a series of 2D Datamatrix barcodes having a size and resolution that can be decoded robustly by consumer phones. Binary data is preconverted into a sequence of 6-bit characters that is supported by Datamatrix. Therefore, we apply a similar technique as proposed by Josefsson [16]. After decoding, the reconstructed 6-bit character sequence is converted back to its original format. Furthermore, the data can be compressed before encoding and is uncompressed after decoding to achieve a possibly higher throughput. The sequence of 2D barcodes are then converted into an animated GIF for presentation and recording. Due to the missing synchronization between camera phone and display, however,

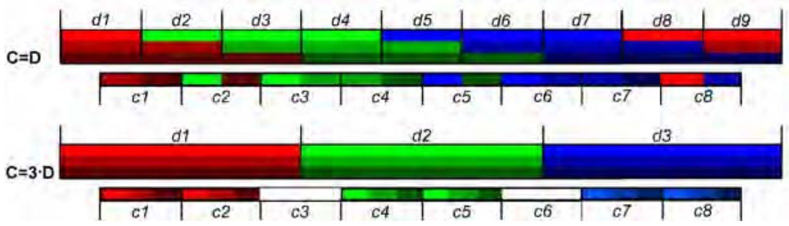


Fig. 3. Encoding: sifted (top) and non-shiftet (bottom) encoding scheme (c_i and d_i are captured and displayed frames respectively, the individual frame-embedded and captured/transmitted barcodes are color-coded)

such a simple approach would be very vulnerable to failures. The reason for this is that during the integration time of the camera, the screen content can change. This effect is illustrated in figures 2e-f for an LCD display. Here, full red, green, and blue 2D barcodes are displayed sequentially. Recording the sequence might show two different frame portions (and consequently two different barcode portions) in the same captured image. We solve this synchronization problem with a new encoding scheme. Instead of encoding one 2D barcode, we encode three different 2D barcodes simultaneously into each frame of the displayed sequence. Each of them will be embedded into the red, green and blue color channels - making it a 3D barcode. This, however, is not being done to triple the transfer throughput, but to increase the robustness of the system by adding redundancy. Every 2D barcode of the original sequence is embedded exactly three times - ones in each of three subsequent 3D barcodes, and it is always encoded into the same color channel. This is illustrated in figure 1(b). Only one 2D barcode is replaced between two subsequent frames. Combining the three color channels in each frame leads to the displayed colored 3D barcodes in lower row of figure 1(b). In addition, we surround each 3D barcode by a colored border. This is necessary for detecting if a captured frame was recorded while the barcode was replaced. Therefore the border color is alternating between yellow, magenta and cyan - colors that are complementary to the RGB code colors. Furthermore, the border color allows detecting which barcodes are encoded and which one will be replaced in the next frame. We use the complementary border color for indicating an upcoming barcode transition within a particular color channel. For example, if a barcode will be replaced in the next frame's blue channel, the border color for the current frame is chosen to be yellow.

Our encoding scheme (figure 3-top) applies equal capture (C) and display (D) rates and adds a two-fold redundancy. It shifts each barcode to three subsequent display frames and ensures that it can be captured completely in at least two frames. The same result (i.e., redundancy and transmission rate) could be achieved with an unshifted encoding scheme and with $C = 3 \cdot D$ (figure 3-bottom), for example. Shifting, however, increases the recognition probability during code resolution transitions and for non-constant capturing times (caused by online JPEG compression in our case). Note that both cases satisfy the Nyquist-Shannon theorem.

3.2 Decoding

After the sequence of 3D barcodes have been recorded on the mobile phone, each captured frame is analyzed for extracting the individual 2D barcodes and finally the encoded information. This task can be split into two preprocessing steps and one final decoding step, as illustrated in figure 4.

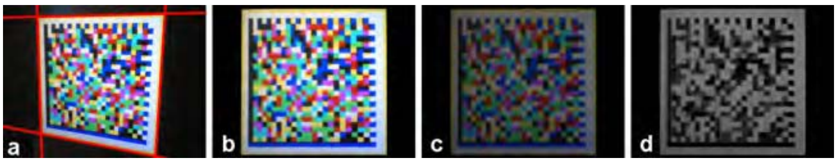


Fig. 4. Preprocessing steps for decoding: (a) captured frame and detected corners, (b) rectified image, (c) contrast and brightness adjusted image, (d) extracted 2D barcode (red channel) in gray scale

Preprocessing. During the preprocessing steps, the 3D barcode in each captured frame is rectified to compensate for perspective distortions and is then contrast and brightness enhanced to compensate for noise. For rectification, the edges of the colored borders are detected through a conventional scan line algorithm. Having found multiple points on each edge, the corresponding line equations can be determined by solving a linear equation system. The intersections of the four edge lines lead to the corner points of the border that can be used to estimate a homography for rectification [17]. In our current prototype, the rectification works well for small barcode resolutions, but our Datamatrix decoder (we applied the Symbian Semacode library) fails often for rectified high resolution codes. However, this has not been critical in our case, since we have to limit the barcode resolution for mobile phone decoding anyway.

$$col_{new} = a \cdot col_{old} + b, a = 255/(255 - 2 \cdot \Delta), b = a \cdot (l - \Delta), \Delta = 127 \cdot c/100 \quad (1)$$

Following this step, the contrast and the brightness of the rectified images are adjusted using equation 1 to reduce image noise. Experimentally we found that a constant brightness reduction of l=20% and a constant increase in contrast of c=50% was optimal in combination with the (unknown) build-in white-balancing function of our mobile phones. This pushes the recognition rate up by a maximum of 20% (compared to no adjustments).

Handling Code Transitions. After optimizing the captured frames, the embedded 2D barcodes can be extracted and decoded. The first step is to detect whether or not a barcode transition happened within a frame. This can be detected by analyzing the border color (which has already been found during rectification, as explained above). If the colors of the upper and the lower border edges are the same, the barcode recorded in the frame is consistent. In this case, all three 2D barcodes that are encoded into the RGB color channels are completely captured. They can be separated, converted into gray scales, and decoded by the Datamatrix decoder. If the colors of the upper and lower edge are unequal, an inconsistency is detected. However, due to our encoding scheme it is possible to guarantee that always two barcodes are consistent (and completely recorded) in one frame. The reason for this is that only one of the three 2D barcodes is replaced between two subsequent 3D barcodes images. By analyzing the color of the upper border edge, we can determine in which color channel a 2D barcode is replaced in the following 3D barcode image (and is consequently recorded inconsistently in the current frame), and which ones are completely recorded. In correspondence to the coding example from section 3.1, a yellow upper border indicates a code transition in the blue color channel. Thus, two different barcodes are captured in the upper and in the lower portions of the current frame's blue channel, while the barcodes in the red and green channels are consistent and complete, in this example. The same applies for the other two possible variations. The complete 2D barcodes can be decoded after converting them into gray scales (cf. figure 5). Note, that the intensity variations of the code bits are not critical for decoding.The inconsistent 2D barcode is discarded, but our encoding scheme guarantees that it will be complete in at least two of the three frames in which it was encoded (i.e., in the best case the same barcode is consistent in all three 3D barcodes; in the worst case it is only consistent in two 3D barcodes). After decoding the individual 2D barcodes, the encoded data packages from each one have to be rebuilt in the primal order of coding. Since it is possible that entire 2D barcodes cannot be decoded at all, and it is likely that recording the 3D barcode sequence does not start with the first frame (users

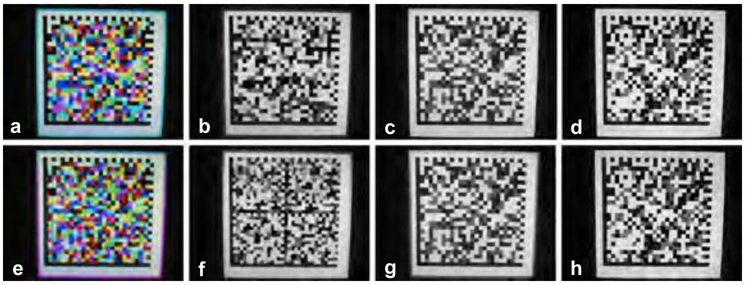


Fig. 5. Decoding: (a) captured 3D barcode with all three embedded 2D barcodes (b-d) completely recovered, (e) example with inconsistent 2D barcode in red channel (f) while green and blue channels can be recovered (g+h)

might start recording at an arbitrary point during the looped image sequence), a correct order of reconstructed data packers is not given by the order of decoding. To overcome this problem, we add a unique frame ID into a header section of each 2D barcode. This allows rebuilding the entire data set in the correct order. The data might have to be uncompressed and transformed back into the original representation, if necessary (in analogy to the encoding, as explained in section 3.1).

4 Results

We tested and optimized our system in two steps: First, it was evaluated with respect to adjustable parameters, such as animation speed, barcode size, capturing resolution, and an optional data compression to maximize its data throughput and robustness. Second, we have carried out a user study to find the final transfer and failure rates under realistic conditions, as well as to get feedback on the acceptance of our approach.

4.1 Optimizing Parameters

Several parameters can freely be chosen in our system: the size of the 2D barcodes (i.e., the number of encoded characters per barcode), the capturing resolution (the capturing speed depends on the adjusted resolution), the animation speed (i.e., the frame rate of the displayed 3D barcode sequence), and whether or not the data should be compressed and decompressed.

All of these parameters interplay with each other and influence the final result. While, for instance, choosing a small capturing resolution and a high animation speed might allow to recorded many 3D barcodes during a particular time period, recognition can fail often since capturing might become too unreliable. As another example, encoding many compressed characters into a single barcode might maximize the transmission of data per 3D barcode, but more time is required for decoding and uncompressing the data. If the recognition rate drops, as yet another example, barcodes might have to be decoded again - which also costs additional time and consequently reduces the overall transfer rate. To achieve the highest possible transfer rate and robustness, the optimal configuration of parameters have to be found. For this, we designed several experiments that recorded recognition and transfer rates under varying parameter settings. All experiments were carried out with a Nokia 6630 mobile phone, using version 1.5 of the Semacode Symbian C++ library for decoding Datamatrix barcodes. First, the recognition rate for 2D barcodes with respect to different capturing resolutions, barcode sizes, and an optionally applied compression was evaluated. As it

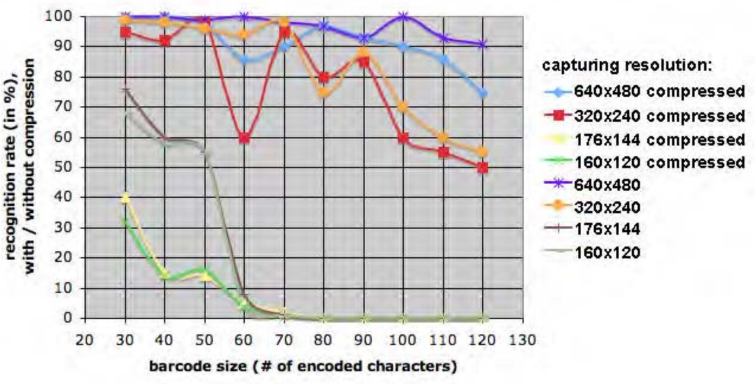


Fig. 6. Recognition rate: successful decoded 2D barcodes under varying capturing resolutions, barcode sizes, and an optionally applied compression

can be seen in figure 6, the highest recognition rates are achieved with capturing resolutions of 320x240 pixels (QVGA) and 640x480 pixels (VGA), while smaller resolutions are mainly not suitable for decoding barcodes sufficiently robust as their sizes increase. In general, barcodes containing uncompressed data decode slightly better because smaller barcodes are required in this case. The reason for this is that most 2D barcodes, such as Datamatrix, are normally used for encoding text and optimize their matrix sizes depending on the probability of character appearance in defined alphabets (e.g. capital letters are less likely and require more coding bits within the barcode matrix than lower-case letters). Compressed data (we applied a deflate compression [18]) is transformed to random characters that require larger matrix sizes in general. In a second experiment, we evaluated the resulting transfer rate against the same parameters as for the recognition rate. It can be seen in figure 7 that the transfer rate is maximal when encoding 70 characters in a single 2D barcode and capturing with a QVGA resolution. With respect to figure 6, a VGA capturing resolution is more accurate, but requires significantly more time, and consequently leads to a lower overall transfer rate. Compressed data performs worst than uncompressed data in this case for the same reason as explained above. Note, that the steep drop-off of recognition rate at around 60 characters is due to possible resolution transitions of the code

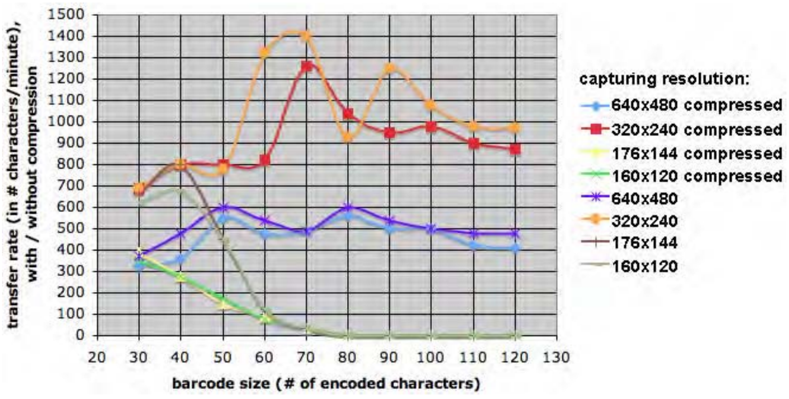


Fig. 7. Transfer rates: number of transmitted characters per minuted under varying capturing resolutions, barcode sizes, and an optionally applied compression

matrix that require the integration of helper lines (cf. figure 5(f)). If 2D barcodes with different resolutions (depending on the encoded content) are encoded into the same 3D barcode, the recognition rate drops significantly if helper lines are inserted.

Based on our experiments, we apply a QVGA capturing resolution, no compression, and encode 70 characters per 2D barcode. This leads to a maximum transfer rate of 1400 characters per minute (23 characters per second) and to a maximal recognition rate of 95% for an experienced user. The fastest capturing rate that was supported by our mobile phone at QVGA resolution (with view finding enabled and direct recording into the integrated flash memory) was 2.5 frames per second.

4.2 User Study

To verify our system using optimal settings under realistic conditions, we carried out a study with a total of 48 unexperienced users during a public event (university's open house). For an experiment we encoded 700 characters of text into a 13x13 cm large 4D barcode that was played with 2.5 fps in a looping animated GIF sequence embedded into a HTML page. The page was displayed in a web browser on a 17 inch LCD screen. Nokia 6630 mobile phones were used for capturing and decoding. The users were able to see the live recording of the camera on the LCD panel of the phone. We asked them to fill as much as possible of the recorded image with the displayed barcode. By pressing a button on the phone, they triggered the beginning and the end of the recording. After recording, the barcode was decoded on the phone. With this study, we were mainly interested in finding the realistic recognition behavior of our system and on getting concrete user feedback. Unexperienced subjects might not always use the system in an optimal way (e.g. they might sometimes not capture the entire barcode image due to arm jitter, or they might record the barcode from a too large distance). On average, we found a recognition rate of 73% for individually extracted 2D barcodes, but due to the encoded redundancy the overall recognition rate was 82% under realistic conditions. Consequently, in 18% of all cases, some parts of the 700 character text were missing while in all remaining cases the whole text was recovered. The averaged time for decoding was about 35 seconds. This is mainly due to the performance of the Semacode library and could not be influenced by our system. However, since decoding was carried out after recording, the users did not have to aim the phone at the screen for this duration. Only for recording (on average 5 seconds), this was necessary. Each subject was finally asked to fill out a questionnaire to provide feedback on the usability of the system - rating various questions between 7 (very good) and 1 (very poor). The averaged results from the questionnaires are shown in figure 8. In general, we can say that the user feedback was overall positive. The long

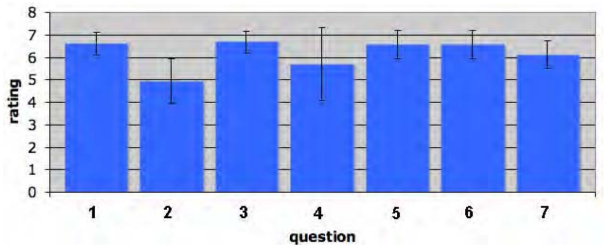


Fig. 8. Results of the user feedback: 1) How easy was it to aim at the barcode? 2) How do you rate the decoding time? 3) How easy was it for you to learn how to use the system? 4) How do you judge the recognition rate? 5) How easy was the handling of the software? 6) How good was the graphical user interface? 7) How much do you like the general concept?

decoding time was criticized most. As mentioned above, this was mainly due to long decoding requirements of the Semacode library which our system could not influence.

5 Summary and Future Work

In this paper we presented the concept and an implementation of unsynchronized 4D barcodes. With our technique, we are currently able to transmit 1400 characters per minute (23 characters per second) with a success rate of 82% (95% for experienced users) from LCD, Plasma, and fast CRT displays to unsynchronized mobile phones. A user study has shown that such a technique would be accepted, if the decoding speed can be improved. Our technique has a much smaller transmission rate than established electromagnetic techniques, such as Bluetooth or WiFi. But it can be used in cases where such connections are are not established per se. Furthermore, it transmits significantly more data than corresponding 2D or 3D barcodes. Besides transmitting data from location- and device-independed public web-pages, we envision applications for recorded and broadcasted video content, for advertisement with billboard displays (as being done already with 2D barcodes) or in movie theaters, for information displays (e.g., transmitting updated schedules at airports or in trains), or for electronically displayed e-tickets (2D barcodes are already accepted to be displayed on mobile phones instead on printed paper). In future, the decoding time has to

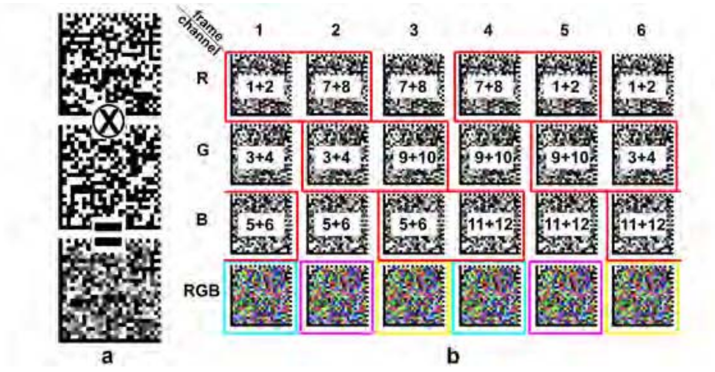


Fig. 9. 5D barcodes: (a) intensity coding of two 2D barcodes, (b) encoding scheme for embedding intensity coded 2D barcodes into a sequence of time-multiplexed 3D barcodes (barcode transitions are framed in red)

be decreased. The Kaywa-Reader (reader.kaywa.com) for example, offers a more robust decoding and a speed-up by a factor of two compared to Semacode. Porting our system to newer Symbian versions (e.g., Symbian Series 60 Version 9.x) allows benefitting from improved camera control functions. This may increase the quality of the captured frames and might open the door to embedding six or more 2D barcodes instead of only three by using discrete intensity variations in addition (cf. figure 9). We have implemented and tested these 5D (height, width, color, time and intensity) barcodes, but deferred them due the too low image quality provided by the utilized mobile phones.

References

1. ISO/IEC: International Organization for Standardization: QR Code. ISO/IEC 18004 (2000)
2. ISO/IEC: International Organization for Standardization: DataMatrix. ISO/IEC 16022 (2000)
3. Han, T.D., et al.: Machine readable code and method and device of encoding and decoding the same, japan patent 3336311 (2002)
4. Han, T.D., et al.: Implementation of personalized situation-aware service. In: ubiPCMM 2005. Proceedings of the First Internaltional Workshop on Personalized Context Modeling and Management for UbiComp Applications (2005)

5. Rohs, M.: Real-world interaction with camera-phones. In: Murakami, H., Nakashima, H., Tokuda, H., Yasumura, M. (eds.) UCS 2004. LNCS, vol. 3598, pp. 74–89. Springer, Heidelberg (2005)
6. Ballagas, R., Rohs, M., Sheridan, J.G.: Sweep and point and shoot: phonecam-based interactions for large public displays. In: CHI 2005. CHI 2005 extended abstracts on Human factors in computing systems, Portland, OR, USA, pp. 1200–1203. ACM Press, New York, NY, USA (2005)
7. Rohs, M.: Visual code widgets for marker-based interaction. In: IWSAWC 2005. Proceedings of the 25th IEEE International Conference on Distributed Computing Systems – Workshops (ICDCS 2005 Workshops), Columbus, Ohio, USA, IEEE Computer Society Press, Los Alamitos (2005)
8. Scott, D., Sharp, R., Madhavapeddy, A, Upton, E.: Using visual tags to bypass bluetooth device discovery. ACM Mobile Computer Communications Review 9, 41–53 (2005)
9. Madhavapeddy, A., Scott, D., Sharp, R., Upton, E.: Using camera-phones to enhance human-computer interaction. In: Adjuct Proc. of Ubicomp 2004, Springer, Heidelberg (2004)
10. Toye, E., Sharp, R., Madhavapeddy, A., Scott, D., Upton, E., Blackwell, A.: Interacting with mobile services: an evaluation of camera-phones and visual tags. Personal Ubiquitous Comput. 11, 97–106 (2007)
11. Vartiainen, P., Chande, S., Rämö, K.: Mobile visual interaction: enhancing local communication and collaboration with visual interactions. In: MUM 2006. Proceedings of the 5th international conference on Mobile and ubiquitous multimedia, p. 4. ACM Press, New York (2006)
12. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G.: The smart phone: A ubiquitous input device. IEEE Pervasive Computing 5, 70 (2006)
13. Shen, H., Coughlan, J.: Reading lcd/led displays with a camera cell phone. In: CVPRW 2006. Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, p. 119. IEEE Computer Society Press, Washington, DC, USA (2006)
14. Tanaka, Y., et al.: Indoor visible light transmission system utilizing white led lights. In: IEICE Transactions on Communications, vol. E86-B(8), pp. 2440–2454 (2003)
15. Komine, T., Nakagawa, M.: Performance evaluation of visible-light wireless communication system using white led lightings. In: ISCC 2004. Proceedings of the Ninth International Symposium on Computers and Communications, vol. 2, pp. 258–263. IEEE Computer Society Press, Washington, DC, USA (2004)
16. Josefsson, S.: The Base16, Base32, and Base64 Data Encodings, RFC 3548, United States (2003)
17. Heckbert, P.S.: Fundamentals of texture mapping and image warping. Technical report, University of California at Berkeley, Berkeley, CA, USA (1989)
18. Deutsch, P.: Deflate Compressed Data Format Specification version 1.3, RFC1951, United States (1996)

A Control Architecture for Long-Term Autonomy of Robotic Assistants[⋆]

Christopher King, Xavier Palathingal, Monica Nicolescu, and Mircea Nicolescu

Department of Computer Science and Engineering
University of Nevada, Reno NV 89557, USA
{cjking,xavier,monica,mircea}@cse.unr.edu

Abstract. A major challenge in deploying service robots into the real world is to design a framework that provides effective, long-term interactions with people. This includes interacting with people in a natural way, dealing with multiple users, and being continually aware of the surroundings. This paper proposes a robot control architecture that addresses these issues. First, it enables the representation of complex, sequential, and hierarchical robot tasks, in a behavior-based framework. Second, it provides a robot with the flexibility to deal with multiple requests and interruptions, over extended periods. Third, it uses a visual awareness mechanism to recognize users and to identify their need for robot interaction. We demonstrate our approach on a Pioneer 3DX mobile robot, performing service tasks in a real-world environment.

1 Introduction

A major challenge in designing robots for service or assistive applications is to enable a natural interaction between robots and non-technical users, while ensuring long-term, robust performance [1]. Robots have traditionally been developed to operate in controlled environments and are programmed to perform tasks in a highly structured and sequential manner. These robots are usually “blind” to other agents in the environment and adapt poorly to changing conditions. The limitations that generally prevent these robots from operating in more realistic domains are their lack of awareness, flexibility, and long-term autonomy.

We propose a control architecture that introduces a level of flexibility and perceptual ability that allows robots to overcome traditional limitations and operate in more dynamic settings. Our architecture equips robots with the *visual-awareness* necessary for them to monitor their surroundings and detect when other social agents have the need for their interaction. Our architecture also provides the means for *long-term autonomy* by enabling robots to manage a large repertoire of tasks over extended periods. Additionally, our system is designed for realistic assistive applications, where multiple people are simultaneously competing for the robot’s assistance.

The contribution of this paper is a framework that addresses three key issues for human-robot interaction in the context of service applications: 1) complexity

[⋆] This work was supported in part by NSF CAREER Award IIS-0546876 to Monica Nicolescu and by the ONR Award N00014-06-1-0611.

and robustness of task representations, 2) long term interactions with multiple users, and 3) awareness of the environment and other agents.

The remainder of the paper is structured as follows: Section 2 presents our interactive framework, Section 3 describes our control architecture, Section 4 discusses our vision-based perceptual approach and Section 5 describes the experimental setup and results. We present our conclusions in Section 6.

2 Interactive Framework

This work is aimed at creating a framework that provides robots with the ability to operate in typical service or assistive application environment. This requires robots to be directed easily by non-technical operators, and function in the presence of multiple users.

Vision-based perception can provide a wealth of information regarding the robot's environment and of other agents within the environment. In the case of human-human interaction, one person can frequently identify the needs of another simply by observing them. Ideally, a service robot should make similar deductions. To this end, a posture-based control paradigm is used. As will be described in Section 4, robots are trained to recognize various postures, which are associated with different tasks. These associations can have a logical relationship (e.g. if a person is observed with an object, the robot should approach the human and accept the object), or may be more symbolic (e.g. if a person is observed with raised hands, a predefined series of actions are performed. In either case, a non-technical user should be able to easily learn how to interact with and receive services from the robot.

A service robot will likely have to perform in the presence of multiple users, where one user may solicit a service while the robot is engaged in another task. To respond accordingly, the robot should interrupt its current activity, detect the new request, and determine appropriate action. Our framework enables this functionality using linked *awareness* and *control* modules. The *awareness module* identifies known users and postures. This information is relayed to the *control module*, which determines the robot's action. Currently, each posture is associated with a task (robot service) that can have *low*, *regular* or *high* priority. When a posture is detected, the robot will perform the associated task, only if the priority of the new task exceeds that of any current activity. The task with the lower priority will be suspended and stored to a priority-based queue. Lower-priority tasks will be resumed when higher-priority tasks are completed. Our architecture provides the flexibility of using different priority queue strategies.

Prioritized task switching resembles human decision-making behavior and is a logical addition to the service robot domain. While people perform this activity switching with ease, robots are presented with the difficulty of maintaining the status of current tasks during interruption, such that the task can be resumed from the same point later. The control architecture proposed in this paper use a set of representations that allow the robot to naturally recover from interrupted tasks without the need to explicitly store any additional state information.

3 Control Architecture

The architecture proposed in this paper is motivated by the Behavior-Based Control (BBC) paradigm, which is popular in robot control. We propose to use this paradigm to **1)** enable the use of both *command arbitration and fusion* within a single representation and **2)** allow the encoding and robust execution of sequential and hierarchical tasks. Historically, the two main action selection mechanisms of *arbitration* and *fusion* have been employed separately in robot control [2], which limits the range of executable tasks. By recognizing the ability of *arbitration* to encode temporal sequences and of *fusion* to combine concurrently running behaviors, we merge the strengths and features of both within a unique task representation. For behavior representation we use a schema-based approach, similar to the work in [3].

3.1 Fusion Primitives

Our controllers are built from two components: *behavior primitives* (BPs) and a *fusion primitive* (FP), which through the combination processes described below result in controllers in the form of *behavior networks* [4].

The BPs express basic capabilities that allow a robot to react to immediate environmental conditions. If input received from the robot sensors meets the preconditions to make a particular BP *active*, then the BP sends an appropriate action to the actuators. For example, an *obstacle-avoidance* BP becomes *active* when sensors detect an object that obstructs the robot.

The *active/not active* status of all BPs is encoded in a n-dimensional vector, where n is the number of BPs. This vector, which we call a *behavior applicability condition* (BAC), contains a 1 (active) or a 0 (not active) for each BP. It is theoretically possible for n BPs to produce 2^n BACs, though many combinations are never seen, and this number is usually much smaller.

The FP linearly combines the vectors produced by all active BPs to produce a control vector that moves the robot toward the direction of highest urgency. BPs are combined using a weighting scheme that modulates the influence each BPs has on the final vector. The set of weights used in the BP summation is determined by the BAC table, as shown in Figure 1. Each BAC entry represents a different set of weights and can be indexed using the n-bit BAC value.

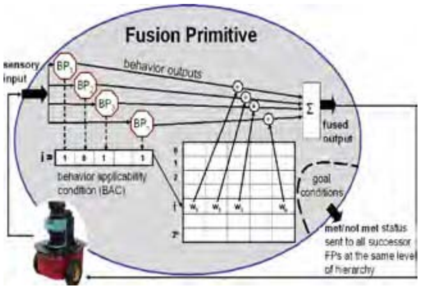

Fusion Primitive
sensory input
BP
behavior outputs
Σ
fused output
behavior applicability condition (BAC)
goal conditions
met/not met status sent to all successor FPs at the same level of hierarchy
w
2ⁿ

Fig. 1. Fusion primitive

At each timestep t, each BP B_i provides a response output vector v_i^t, which represents a desired heading for the robot. The FP's output is a linear combination of the vectors $[v_1^t \cdots v_n^t]$, according to the BAC superposition weights $W^t = [w_1^t \cdots w_n^t]$:

$$V_r^t = \sum_{i=1}^{n} w_i^t v_i^t \tag{1}$$

The multiple BACs represent different environmental situations, since different behaviors are "applicable" in each case. The weights of behaviors within each BAC encode the mode of performing the current task given the situation. For example, in a *target reaching* task, the robot may be influenced by *corridor-follow, target-follow* and *avoid-obstacle* behaviors in the presence of an obstacle, while only *target-follow* would be active in an open space. Inferring the fusion weights is a challenging task requiring time-consuming fine-tuning. We used a method of learning weights through human-provided demonstration [5].

3.2 Hierarchical Task Representations

With fusion primitives alone, a controller can only encode *flat representations* of tasks using sequencing of fusion primitives. This does not have the modularity needed to allow more complex tasks to be created from existing ones. We enable this higher-level of representation by grouping fusion primitives into *behavior networks*, which can be nested to allow hierarchical representations of tasks. In these networks, links between components represent task-specific precondition-postcondition dependencies, which provide a simple way to represent complex activities (Figure 2).

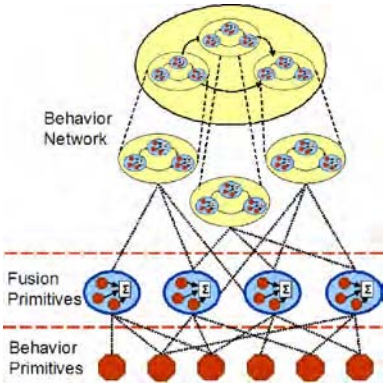


Fig. 2. Generic hierarchical task

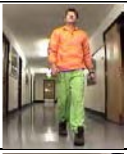
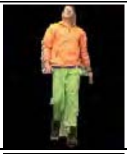
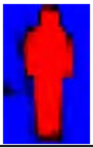
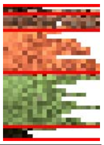
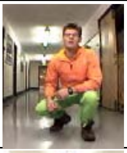
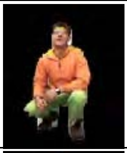
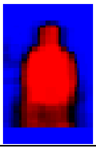
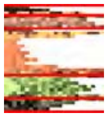
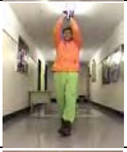
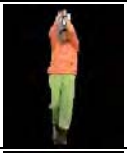
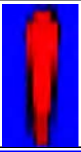
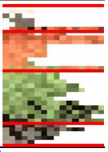
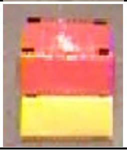
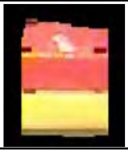
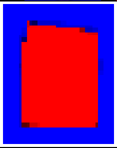
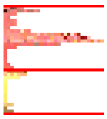
The term *metabehavior* is used to describe both fusion primitives and nodes of a behavior network, as both have similar functions in the network. Each metabehavior encapsulates information about the behavior's preconditions and goals (postconditions). These conditions are continuously monitored to ensure proper task execution. The only difference between a behavior network node and a fusion primitive is that the network node activates underlying metabehaviors, while a fusion primitive activates only its component primitive behaviors. When a behavior network node becomes active, its underlying components are enabled, and the subnetwork becomes the current "network" to be executed. Upon completion, the behavior network node updates its goal status accordingly. Successor behaviors will detect the achievement of the goal and a new network node will execute. To perform complex tasks, the robot activates the metabehavior at the task's topmost level and the activity propagates through the task's steps.

An advantage of this architecture is that it is adaptive to both favorable (inadvertently satisfying a goal) and unfavorable (undoing a goal) changes. Since the behavior's pre and post-conditions are continuously monitored, the active behavior is driven by environmental state, thus providing a sense of "awareness" about task progress. The representation also allows interruption to occur without additional modifications. When a task is interrupted, the behaviors preserve the current status of execution until the robot can return to the task. The behaviors' continuous grounding in sensory information allows the task to be performed correctly, even when environmental conditions changed during suspension.

4 Vision-Based Human Posture Recognition

The role of the *visual awareness* module is to provide the robot with the capability of detecting the presence of humans that might be interested in interacting with the robot. Toward this end, we developed visual capabilities that can recognize human postures that are likely to be relevant to the robot-human interaction. Postures description and examples are displayed in Table 1 (first and second column).

Table 1. Set of Postures. Column 1: Posture description. Column 2: Sample video frame. Column 3: Segmented image. Column 4: Shape model. Column 5: Color model.

Posture Type/Description	Image	Foreground	Shape	Color
The Standing Posture A good posture to recognize. It is displayed frequently and may indicate that the human is on the move or engaging in a task.				
Kneeling Posture Given the robot's size, humans must crouch or kneel to pass objects to and from the robot. A robot should therefore recognize a crouching human.				
Arms-Up Posture Humans learn at a young age that they can attract another's attention by raising their hand and a robot should respond accordingly.				
Object Posture Held-objects were trained independently from the human. This increases model robustness and allows the robot to orient itself toward the object.				

4.1 Related Work in Visual Identification/Tracking

The identification and tracking of objects in a video feed is reasonably easy when a relatively static background can be maintained. The background features can be modeled using intensities, Gaussian distributions, non-parametric distributions [6], etc., which all allow objects that do not match the stored models to be segmented as foreground. These techniques can be robust to gradual changes in the background [6] or even smooth and linear camera movements [7], but are still unsuitable for use on a mobile robot. Robot camera movements are usually too complex to be stabilized by motion-modeling algorithms and the limitless variability of the shape, color, and texture of background features precludes the use of standard feature-based background models. Consequently, foreground-modeling techniques are generally the norm for robotics. The most common of these approaches models objects as a colored blob. While most of the existing techniques can be used to detect multiple colored blobs, they usually do not enforce the relative positioning or relative size of the blobs. For example a person with a red hat and blue shirt would likely appear to be the same as a person with a blue hat and red shoes. Also, these methods rarely incorporate shape due to efficiency constraints. The method we developed improves the convenience of previous techniques, enforces relative position and size of modeled objects, and incorporates shape information without sacrificing speed [8].

4.2 Training

For our demonstration, the robot was trained to recognize three different postures from two different people, as well as a colored box (Table 1 second column). The training process required a person to present each pose to a static robot for about fifteen-seconds. During this period, an adaptive background modeling technique [9] was used to segment the foreground object (Table 1 third column). Segmented objects were normalized in terms of position and height and were used to form models of the object's shape and color.

Although human silhouette can be highly variable, there is enough regularity to warrant a shape-based model. This was done by dividing the segmented and normalized foreground object into a matrix of square blocks. A map is then generated that contains the likelihood that each block is part of the foreground or background (Table. 1 forth column (Red corresponds to high foreground probability and blue to low probability)).

Given N training frames, the probability at each block i is:

$$p_{shape}(i) = \frac{1}{N} \sum_{k=1}^{N} fg_k(i) \tag{2}$$

$fg_k(i)$ equals 1 if block i belongs to foreground in frame k, and 0 otherwise.

Color models were developed to exploit the fact that human figures usually contain coloration that is less variable in the horizontal direction than in the vertical direction. Variability usually occurs at the transitions between face and shirt, shirt and pants, etc. Also, the relative size and location of these regions remain reasonably consistent even as a human moves. We recorded this natural grouping by dividing the object into a vertical stack of horizontal color bands, where the size and position of each band was determined by the object's colors. Bands frequently corresponded with the face, shirt, pants, and shoes as seen in Table 1 (fifth column). The color-values corresponding to each band were modeled as a mixture of Gaussians in three-dimensional RGB color-space. In addition to color composition, the model contained information about vertical location, and size of the regions.

4.3 Detection and Tracking

Since humans tend to assume an upright posture, they will usually occupy a larger proportion of an image in the vertical direction than they will in the horizontal direction. This property simplifies an object search because it allows promising x-axis locations to be identified before considering y-axis locations.

For both x-axis and y-axis searches, pixels in the image were assigned a probability that represented the likelihood that the pixel's color was present in the foreground object space. Pixels with color values matching the most prominent colors in the target model are assigned high probabilities, while colors not found in the model are assigned a probability of zero. For a given model, the probability that pixel i with color (x_r,x_g,x_b) belongs to the model is determined using all Gaussians in all bands of the color model:

$$p_{color}(i) = \frac{1}{N_{bands}} \sum_{bands} \frac{e^{-\left(\frac{(x_r-\mu_r)^2}{2\sigma_r^2}+\frac{(x_g-\mu_g)^2}{2\sigma_g^2}+\frac{(x_b-\mu_b)^2}{2\sigma_b^2}\right)}}{\left(\sqrt{2\pi}\sigma_r\right)\left(\sqrt{2\pi}\sigma_g\right)\left(\sqrt{2\pi}\sigma_b\right)} \tag{3}$$

For the horizontal search, a summation was made for the resulting probability values in each column of pixels. Local maxima were then recorded as likely positions along the x-axis (Figure. 3 (a)).

A vertical search was conducted on the region surrounding every probable x-axis location using a similar technique. This would yield locations along the y-axis that had a high probability of matching the vertical position and coloration of the associated model (Figure. 3 (b)).

The object-shape probability map is used for a final measure of similarity. Certain blocks of the shape-map will have a high probability of falling on the figure while other areas will have a low probability. The shape-based probabilities are used to weight the color-based probabilities for each region, in order to produce a final similarity score. Regions corresponding to high scores are considered foreground (Figure. 4).

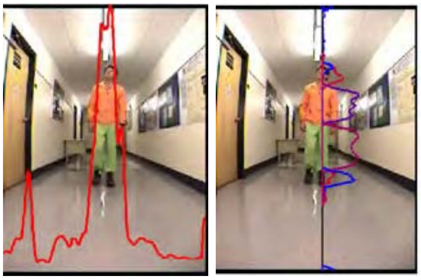

a) X-axis **b)** Region

Fig. 3. X-axis/Region probabilities

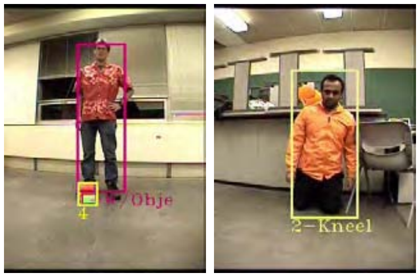


Fig. 4. Detection

4.4 Efficiency

It should be noted that although every color region is represented by many Gaussians and although the image is searched for each posture of each object, our implementation allows this to be done quite efficiently. We use a one-time preprocessing step that compiles the Gaussians into a 3D array indexed by (R,G,B). With a single reference to this array, a probability measure can be obtained to determine the likelihood that that a particular pixel is part of an object. This optimization allows tracking to be performed in real-time (20 frames/sec) on a modest 1 GHz computer, even when tracking over a dozen models.

5 Experimental Setup and Results

We validated our approach with a Pioneer 3DX mobile robot equipped with a SICK LMS-200 laser rangefinder, two rings of sonars, and a pan-tilt-zoom (PTZ) camera. For robot control we use the Player robot device interface [10]. We performed the robot control experiments in a classroom environment (Figure. 5). In these experiments, two different users interacted with the robot using three different postures: *standing*, *arms-up*, *kneeling (with object)*, and *kneeling (without object)*.

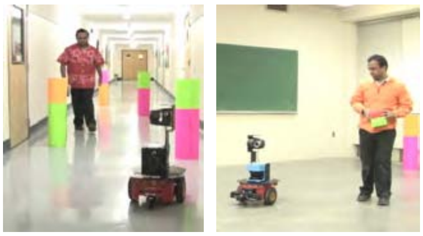


Fig. 5. Experimental environment

The robot's behavior primitives consist of: laser obstacle avoidance, attraction to a goal object, attraction to unoccupied space, attraction to walls, rear sonar obstacle avoidance, tangent wall follow, circular avoid, and pick up and drop objects. The behaviors produce a motor command output in the form of a vector in the robot's coordinate system. Using these behaviors we created a set of fusion primitives and task controllers, which constitute our robot's repertoire of services. The robot's tasks involve a series of visit-target and object-transport tasks, representative for a service robot's potential delivery scenarios. Each of these tasks has a given priority and is associated with one of the users' posture, as shown in Table 2. The *visit-target* component of each task is a metabehavior, whose goals are achieved when the robot is a particular distance with respect to the target.

Table 2. Task requests. *User row*: The user # and posture type (Au=Arms Up, Kn=Kneel, Ob=Object). *Request row*: The requested task. *Action row*: The task pushed to or popped from the queue. *Queue row*: The queue contents. *Current row*: The task currently being executed ('*' indicates that the task was executed to completion).

User Posture	–	1 Ob	2 AU	1 Kn	2 Kn	2 Ob	1 AU	–	–	–	–	–	–
Request Priority	T0 low	T3 med	T5 med	T1 med	T4 high	T6 high	T2 high						
Action		push T0	push T5	push T1	push T3	push T6	push T2	pop T6	pop T2	pop T3	pop T5	pop T1	pop T0
Queue		T0	T5, T0	T5, T1, T0	T3,T5, T1, T0	T6,T3, T5,T1, T0	T6,T2, T3,T5, T1,T0	T2,T3, T5,T1, T0	T3,T5, T1, T0	T5, T1, T0	T1, T0	T0	
Current	T0	T3	T3	T3	T4	T4	T4*	T6*	T2*	T3*	T5*	T1*	T0

In addition to the above tasks, the robot is equipped with a wander task (T0), which has a *low* priority and is executed whenever the robot has no requests to service. The standing posture is not associated with any task, but serves as a trigger from the *visual awareness* module that a user is in vicinity.

In our experiments the two users requested services from the robot over an extended period, in order to demonstrate the main features of our approach: 1) awareness to the presence of multiple people during task execution, 2) ability to handle multiple requests, 3) ability to handle task interruptions and 4) long-term robot autonomy.

Upon starting, the robot begins wandering, while waiting for requests. If the robot detects a user (through the standing posture), the robot interrupts its task and reduces its speed. If within several seconds no new postures are detected (i.e., no requests from the user), the robot resumes its task, ignoring that user for some predefined period of time, unless the user later displays a non-standing posture. This later step is needed to avoid infinite loops of attending to a passer-by user. When the user displays a non-standing posture for a sufficient duration, the robot queues the request and provides audible confirmation. The robot ignores a person for a short period after they issue a request, and ignores requests for tasks that are in the queue or are currently being executed.

We performed experiments for two different task request scenarios, with each scenario repeated four times. We use the same sequence of requests for each scenario to establish a baseline for evaluation, both from the perspective of task execution (the *control module*) and posture recognition (the *visual awareness module*). We used different priority schemes for each scenario.

Results. In both scenarios, the robot correctly identified the postures (and thus the requests), made the correct decisions regarding priorities, and correctly executed the tasks. All runs took approximately 20 minutes. In scenario 1, the only error occurred in the fourth run, where the robot detected a request for task 4 instead of task 1. In the third run of scenario 2, the user made the mistake of requesting task 6 before task 4. However, this being a change in scenario, the robot correctly identified and serviced the requests. In both scenarios, the robot processed tasks with highest priority first. For scenario 1, tasks with equal priority were processed using LIFO (last-in-first-out), and for scenario 2, FIFO (first-in-first-out) was used. Graphical results are shown for scenario 2 in Figure. 6. The Red squares mark the time when requests are received and green squares represent task completion. Task progress is shown by incremented numbers. When an interrupted task is resumed, these numbers show that the robot continues the task from where it left off.

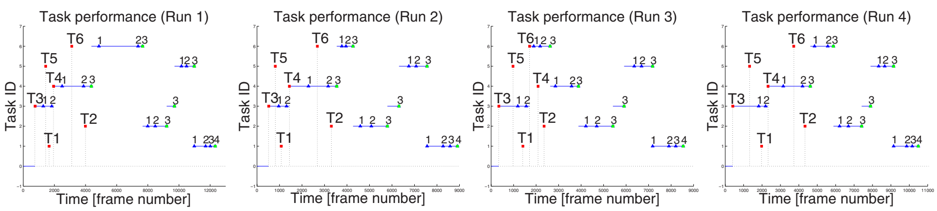


Fig. 6. Robot task execution and request identification. Red squares: New requests. Green squares: Task completion. Blue triangles: Subtask completion with ID.

6 Conclusion

In this paper we propose a framework for developing robot assistants that addresses two key issues of human-robot interaction: *awareness* of the environment and other agents, and *long-term interaction* with multiple users. Our awareness mechanism is built on visual capabilities that allow the robot to identify multiple users, with multiple postures, in real-time, in dynamic environments where both the robot and human users are moving. Long-term human-robot interaction is supported by a novel control architecture that allows a robot to accommodate multiple user requests and task interruptions and it enables the representation of complex, sequential and hierarchical robot tasks. The architecture provides the robot with flexibility in dealing with multiple users, such as to accommodate multiple user requests and task interruptions, over extended periods. We validated our approach on a Pioneer 3DX mobile robot, performing service tasks in a real-world environment.

References

1. Fong, T., Nourbakhsh, I., Dautenhahn, K.: A survey of socially interactive robots. Robotics and Autonomous Systems 42, 143–166 (2003)
2. Paolo Pirjanian. Behavior coordination mechanisms - state-of-the-art. Tech Report IRIS-99-375, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, California (1999)
3. Arkin, R.C.: Motor schema based navigation for a mobile robot: An approach to programming by behavior. In: IEEE Conference on Robotics and Automation, pp. 264–271. IEEE Computer Society Press, Los Alamitos (1987)
4. Nicolescu, M.N., Matarić, M.J.: A hierarchical architecture for behavior-based robots. In: Proc., First Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, July 2002, pp. 227–233 (2002)
5. Nicolescu, M., Jenkins, C., Olenderski, A.: Learning behavior fusion estimation from demonstration. In: IEEE Intl. Symp. on Robot and Human Interactive Communication (RO-MAN 2006), Hatfield, United Kingdom, pp. 340–345. IEEE Computer Society Press, Los Alamitos (2006)
6. Elgammal, A., Duraiswami, R., Harwood, D., Davis, L.: Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. In: Proceedings of the IEEE, vol. 90, pp. 1151–1163 (2002)
7. Kang, J., Cohen, I., Medioni, G.: Continuous tracking within and across camera streams. In: Computer Vision and Pattern Recognition, pp. 267–272 (2003)
8. King, C., Palathingal, X., Nicolescu, M., Nicolescu, M.: A vision-based architecture for long-term human-robot interaction. In: Proc., the IASTED Intl. Conf. on Human Computer Interaction (2007)
9. Nicolescu, M., Medioni, G., Lee, M.-S.: Segmentation, tracking and interpretation using panoramic video. In: IEEE Workshop on Omnidirectional Vision, pp. 169–174. IEEE Computer Society Press, Los Alamitos (2000)
10. Gerkey, B., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proc. the 11th International Conference on Advanced Robotics, pp. 317–323 (2003)

Classification of Structural Cartographic Objects Using Edge-Based Features

Güray Erus and Nicolas Loménie

Université de Paris 5, Laboratoire SIP-CRIP5
45 rue des Saints Pères; 75006; Paris; France

Abstract. The aim of this study is to classify structural cartographic objects in high-resolution satellite images. The target classes have an important intra-class variability because the class definitions belong to high-level concepts. Structural attributes seem to be the most plausible cues for the classification task. We propose an Adaboost learning method using edge-based features as weak learners. Multi-scale sub-pixel edges are converted to geometrical primitives as potential evidences of the target object. A feature vector is calculated from the primitives and their perceptual groupings, by the accumulation of combinations of their geometrical and spatial attributes. A classifier is constructed using the feature vector. The main contribution of this paper is the usage of structural shape attributes in a statistical learning method framework.

We tested our method on CNES[1] dataset prepared for the ROBIN Competition[2] and we obtained promising results.

1 Introduction

With the spread of very high resolution satellite images, more sophisticated image processing systems are required for the automatic extraction of information. CNES prepared a database of cartographic object images to develop tools and algorithms to exploit images acquired by the new generation satellites. In the frame of the ROBIN Competition, a classification task entitled *discrimination of compact structures* is defined on a subset of this database. The database consists of centered images of objects from the 4 target classes, namely, *"Roundabouts"*, *"Crossroads"*, *"Insulated buildings"* and *"Bridges"*. The task is to determine the class of the object on the given test images. Figure 1 presents sample images of each class.

Low-level pixel based classification methods are not well adapted for the task. In low-resolution, the objects, that disappear and become part of the texture, are in general indistinguishable. In high resolution, the discriminative structural

[1] French National Space Agency.

[2] Object Detection on Digital Image Bases. A research project funded by the french ministry of defense and the french ministry of research, with the aim of producing datasets, ground truths, competitions and metrics for the evaluation of object recognition algorithms. http://robin.inrialpes.fr/

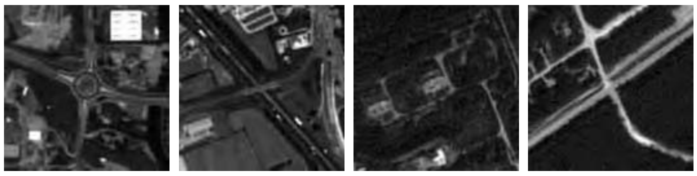


Fig. 1. Sample images. a. Roundabout, b. Bridge, c. Insulated building, d. Crossroad.

features of the objects appear. Man-made geographical objects have well-defined, mostly geometrical structures. However, the main difficulty is the high variance in scale, rotation, illumination, and even shape between the objects belonging to the same class. The class definitions belong to high-level semantic concepts and there is no direct one-to-one correspondence between low-level object parts or features. For example, the "Roundabout" class can be literally defined as "a central circular shape connected to three or more linear shapes towards the center".

Another difficulty is that the borders between the target object and the background are quite fuzzy on the images. Consequently, it is practically not possible to isolate the object with an automatic segmentation method.

Taking into consideration the formentioned difficulties of the problem, we look for a method that detects the potential structural elements of the target object and combines them to reveal the inherent class definition from the given examples.

[1] proposed perceptual grouping of edge features for classification. The main idea is to count evidences of the target class in the image. The evidences are detected starting with the line segments and applying grouping rules to obtain, hierarchically, co-terminations, L and U junctions, parallel lines and groups, and closed polygons, From these, three scalar features are calculated and the images are labeled as Structure, Non-structure or Intermediate using a nearest neighbor classifier. The size of the final feature vector is only adequate for a very coarse classification. We propose two major modifications: To use more primitives (particularly circular ones) and to take into consideration primitives' geometrical and spatial attributes while counting them. In that way a large feature vector with the evidence accumulation on various attribute value intervals is calculated.

The Viola-Jones [2] face detector is based on the Adaboost algorithm, where weak classifiers, obtained each from a single feature, are combined to construct the final strong classifier. In each step of the algorithm the feature with the smallest error is added to the strong classifier with an appropriate weight. They used simple Haar-like rectangular regions as features. [3] proposed an improvement by calculating feature values by sampling individual pixels, determined using a genetic algorithm. We propose to use the Adaboost learning framework with the edge-based feature vector as the weak learners. The algorithm will select the most pertinent features to represent the structure of the class.

The following section explains the main steps of our method. Section 3 presents the experimental results. Finally the conclusions and the perspectives are given in section 4.

2 Adaboost Classification Using Edge- Based Features

The classification method consists of three main parts: The extraction of the edge-based primitives, calculation of the feature values from the primitives and the classification using an Adaboost learner.

2.1 Extraction of Primitives

The edges of an object give sufficient information about its shape. It's possible, in general, to recognize an object only from its edges, even when an important part of the edges are missing. The aim of this first step is to represent the edges of the image as a set of primitive shapes. The primitives are *the straight lines*, *the circle arcs* and finally *the blobs*, closed edgel chains . The extraction is done on a Gaussian image pyramid with 4 levels in order to detect the primitives on different scales. The following steps of the extraction are applied in each scale and the results are grouped together:

- Sub-pixel edge detection using a modified Canny [4] edge detector. Sub-pixel precision is necessary in order to guarantee a robust approximation by line segments or arcs.
- Grouping of edge points in edge-chains. A new chain is started in each junction.
- Detection of blobs.
- Polygonal approximation using Douglas-Peucker algorithm [5].
- Detection of arcs. The segments are recursively joint with their neighbors if they can be fit by a circular arc. The pseudo-code of the algorithm is given below:

```
for each consecutive line segment pair $(s_1, s_2)$ do
  $p_1 = pixels(s_1),\ p_2 = pixels(s_2)$
  $e_1 = approximation\_error(s_1), e_2 = approximation\_error(s_2)$
  $p_3 = join(p_1, p_2)$
  $s_3 = fit\_circle(p_3)$
  $e_3 = approximation\_error(s_3)$
  if $e_3 \leq max(e_1, e_2)$ then
    $label(s_3, \text{“}Circle\text{”})$
    $replace(s_1\ and\ s_2, s_3)$
  end if
end for
```

- Elimination of small lines and arcs.
- Detection of L and T-junctions.

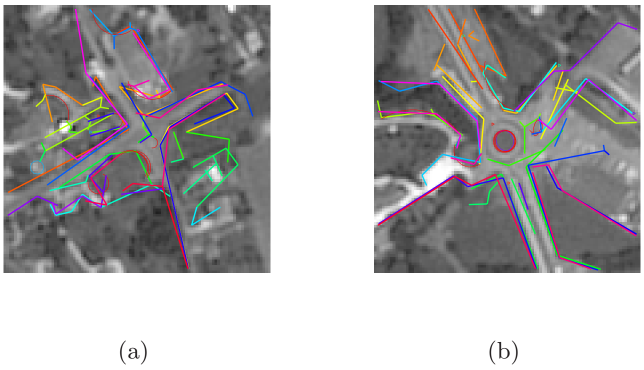


Fig. 2. Extracted primitives. a. Bridge, b. Roundabout.

Figure 2 shows the extracted primitives on a roundabout and a bridge images. It's necessary to not to miss the primitives belonging to the object as much as possible for a good performance in the following steps, at the expense of having spurious primitives (that is also inevitable because the object is not isolated from the background). The random accumulation of the evidences from these spurious primitives should not have a systematic bad effect on the final learners. The edges remaining after a simple filtering by hand-made rules are shown in figure 3 as a visual justification.

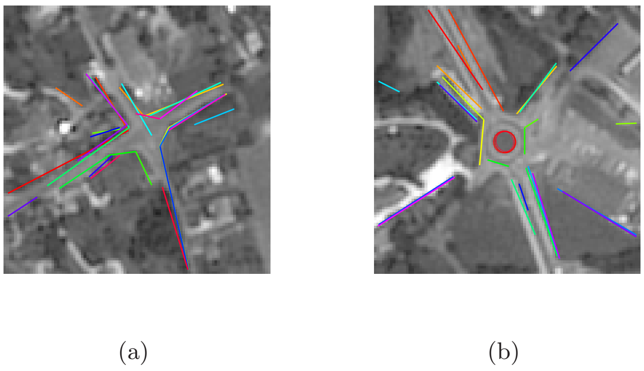


Fig. 3. Extracted primitives after filtering by handmade rules. a. Bridge, b. Roundabout.

2.2 Feature Calculation

The feature vector F is composed of features F_i corresponding to the number of primitives in different bins. The bins are defined by value intervals of combination of spatial, geometrical or relational attributes. For each primitive

$S \in \{line, arc, L\ junction, T\ junction\}$, given that C is the image center point, a set of attributes A_p is defined as:

- Arc attributes $A_{arc} = \{dist.\ to\ C,\ radius,\ arc\ angle\}$
- Blob attributes $_{Ablob} = \{dist.\ to\ C,\ major\ axis\ length, compacity\}$
- L junction attributes $A_L = \{angle\ btw.\ lines,\ dist.\ to\ C,\ len.\ 1,\ len.\ 2\}$
- T junction attributes $A_T = \{angle\ btw.\ lines,\ dist.\ to\ C,\ len.\ 1,\ len.\ 2\}$

Briefly, geometrical attributes are related to shape size, the relational attributes are related to the junctions, and as the objects are centered on the images, the spatial attributes are the position and the orientation with respect to the image center.

For each attribute A_i, a set I_i of interval value tuples (lower bound and upper bound) is defined. The size of I_i can be set according to the required detail level on A_i.

Given that $n(P)$ is the number of primitives, $n(A_i)$ is he number of attributes for the primitive P_i , and $n(I_{ij})$ is the number of intervals for the attribute A_{ij}; the total number of bins is calculated as

$$\sum_{i=1}^{n(P)} \prod_{j=1}^{n(A_i)} n(I_{ij})$$

As an example, a sample bin is:

The **Arcs** with a **distance to image center in [0, 20] , radius in [0, 20]** and **arc angle in [pi, 2*pi]**

For our task we calculated a feature vector of size 209 by defining the interval values manually. The size of feature vector may be used for tuning performance versus rapidity.

2.3 Classification

The classical Adaboost algorithm is designed for a binary classification problem. For our case, where there are 4 classes, we implemented two methods that combine binary classifiers.

[6] proposed a voting scheme using the 1 against rest approach: given K classes, a classifier between 1 class and the $K - 1$ other classes is trained. To classify an object, a confidence value is calculated for each classifier and the object is assigned to the class with the maximal confidence value.

The weak learners used in Adaboost classify the data according to the optimal threshold value on the selected dimension. We used the absolute value of the normalized distance to this threshold value, as the confidence value of a sample. We trained 4 classifiers, one for each class, after relabeling the data as $\{object, non\text{-}object\}$ according to the corresponding class.

As a second approach, a hierarchical classification with a predefined order is proposed. We grouped together classes which are more similar in the first level.

The first classification is done between the sets $\{Bridges, \ Crossroads\}$ and $\{Roundabouts, \ Insulated\ buildings\}$. In the second level each set is classified to the two corresponding classes. In training, we learned three classifiers, C_1 for the first level and C_{21}, C_{22} for the second level. The final confidence values are calculated by the multiplication of the confidence values in each level.

3 Experimental Results

We tested our methods on an image set with 961 training images and 961 test images. Both training and test images are nearly equally partitioned in 4 classes.

We evaluated the results according to the ROBIN competition evaluation principles and metrics [7]. The class decision is the result of a thresholding operation on the confidence value. The threshold is initially set to the minimal confidence value. As the threshold increases, the objects with a confidence value lower than the threshold are labeled as "Ambiguous". Discrimination is the rate of correctly labeled objects when the ambiguous objects are discarded. Uncertainty is the rate of ambiguous objects. Three specific operating points are measured:

– Discrimination at minimal uncertainty rate: D
– Uncertainty at maximal discrimination rate: U
– Equal discrimination and uncertainty rate: EDU

We obtained a discrimination at zero uncertainty (D) of 0.7242 and equal discrimination and uncertainty rate (EDU) of 0.7884 using the hierarchical classification (figure 4). This corresponds to a total error of 27.5% when all samples are classified in one of the 4 classes. For the classification by vote, these values are respectively 0.7055 and 0.7665. The scores of the voting method are quite

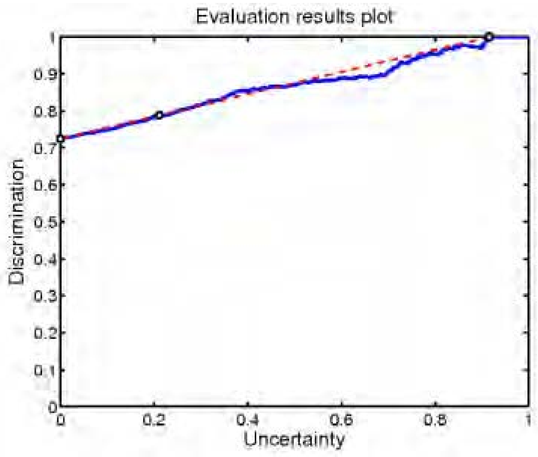


Fig. 4. Classification results. D, EDU and U values are respectively marked by circles from left to right on the plot.

close to the scores of the hierarchical method that required manual ordering of classes. With these scores, according to the unofficial results of the first phase of the competition, we had the first place between the two teams participated to this classification task.

The confusion matrix for the hierarchical classification is given in table 1.

Table 1. Confusion matrix M. $M(i, j)$ is the rate of objects in class i labeled as j.

	RA	CR	IB	B
RA	**0.8294**	0.0471	0.1118	0.0118
CR	0.0969	**0.5271**	0.0891	0.2868
IB	0.1086	0.0300	**0.7903**	0.0712
B	0.0075	0.1617	0.0489	**0.7820**

The confusion between the Crossroads and Bridges classes is very high. Several crossroad objects are labeled as bridges. This is mainly due to the difficulty of discriminating the two classes. They have quite similar structural components, that differ only in the central intersection of lines. When the image resolution is poor, the discrimination becomes very difficult. Figure 5 shows some examples of misclassification

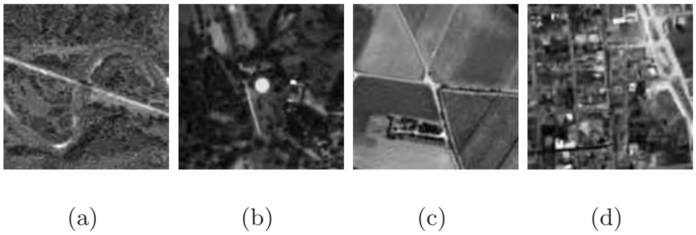


Fig. 5. Some classification errors. a. Bridge labeled as crossroad, b. Insulated building labeled as roundabout, c. Crossroad labeled as bridge, d. Crossroad labeled as roundabout.

The components of the strong learners explains what is learned as class definitions. A very brief analysis of the 4 strong learners obtained by the voting method gives the following observations:

For the roundabout class the features with highest weights are central arcs with different combinations of radius and angle. Surprisingly no feature depending on line attributes is selected. This shows that a roundabout is classified according to evidences of central circle arcs with various spatial and geometrical attributes. We believe that the roads connected to the central circle are not selected by Adaboost, because they also exist on objects of other classes and no different negative example is presented to the system. For the insulated buildings the blobs are the most discriminant features. The crossroads and bridges are discriminated by linear features and junctions. For the bridges long lines passing from the center are selected. Interestingly, for the crossroads, lack of circular arcs in the central region is also considered as a discriminant factor. A more systematic analysis of the classifiers is an important perspective and is necessary in order to improve the method.

4 Conclusions and Perspectives

We proposed a classification method, in which edge-based structural shape attributes are used in a statistical learning framework. In this way, we aimed to bridge the gap between the low-level image features and the high-level object definitions without using a sophisticated high-level object representation. The first results and their interpretations are promising. We obtained acceptable classification results and there is room for improvement mainly in two directions: fusion of binary classifiers and definition of more discriminant features.

References

1. Iqbal, Q., Aggarwal, J.: Retrieval by classification of images containing large man-made objects using perceptual grouping. Pattern Recognition Journal 35, 1463–1479 (2002)
2. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vision 57, 137–154 (2004)
3. Abramson, Y., Steux, B., Ghorayeb, H.: Yet even faster (yef) real-time object detection. International Journal of Intelligent Systems Technologies and Applications 2, 102–112 (2007)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8, 679–698 (1986)
5. Douglas, D.H., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Canadian Cartographer 10, 112–122 (1973)
6. Tax, D., Duin, R.: Using two-class classifiers for multiclass classification. In: Kasturi, R., Laurendeau, D., Suen, C. (eds.) 16th International Conference on Pattern Recognition (Quebec City, Canada, Aug.11-15), vol. 2, pp. 124–127. IEEE Computer Society Press, Los Alamitos (2002)
7. Dangelo, E., Herbin, S., Rativille, M.: Robin Challenge, Evaluation Principles and Metrics (2006), `http://robin.inrialpes.fr`

Determining Atmospheric Dust Concentrations During Strong Flow Perturbations Using a Digital-Optical Technique

J. McAlpine, D. Koračin, K. Veropoulos, D. Boyle, E. McDonald, and G. Lamorey

Desert Research Institute, Reno, Nevada
jd.mcalpine@dri.edu

Abstract. Dust emissions due to low-level flight of a helicopter are studied as part of the Integrated Desert Terrain Forecasting for Military Operations project. Atmospheric concentrations of PM_{10} were measured at different heights downwind of the helicopter flight path. Digital video images captured the entrainment and dispersion of the dust plume formed by the wake of the helicopter during each pass down the flight course. The video data are analyzed to relate the dust plume strength to degradation of local visibility. A strong relationship between color changes/standard deviations and plume strength is found. This relationship is used to develop an algorithm that can determine local visibility degradation due to local PM_{10} concentrations around a helicopter. This algorithm can be combined with concentration output data from an atmospheric dispersion model to simulate visibility in a helicopter simulator.

1 Introduction

The Integrated Desert Terrain Forecasting for Military Operations Project is a multi-departmental research project being conducted by the Desert Research Institute (DRI) to assess the impact of military operations on air quality and visibility in an arid environment. A major aspect of this project so far has been the characterization of dust emissions from the desert surface due to operations such as artillery back-blasts [1] and vehicle movement [2]. Current research efforts are focused on the characterization of dust emission due to the low flight of helicopters near the desert floor.

One of the goals of this part of the project is to understand the impact of helicopter induced dust emission on local visibility. Local visibility degradation can be an operational hazard for military operations by affecting the helicopter pilots sense of his/her environment and by obscuring the view of the battlefield. The findings from this work are intended to assist with the development of a simulation of helicopter operation in the desert environment, in which visibility degradation is a major part. The simulation is being developed by the DRI Center for Advanced Visualization, Computation and Modeling (CAVCaM) using the CAVE Automatic Virtual Environment (CAVE).

As part of the helicopter dust emission study, an experiment was conducted at the U.S. Army Yuma Proving Grounds in May, 2007 to measure the dust entrainment and dispersion from a military helicopter. A UH-1 "Huey" helicopter was flown down a desert terrain course under a variety of conditions and dust concentrations were measured downwind by a variety of instrumentation. Images of each pass of the helicopter and its resultant plume were recorded with digital video tape.

The goal of this study was to be able to relate changes in the visual frame to the local concentration of dust in the air entrained by a strong flow perturbation such as a helicopter wake. This was achieved by comparing the recorded images against the "strength" of the dust plume created by the helicopter. The results of this study have been used to develop a method to simulate local visibility degradation based on particle concentration in the immediate environment.

2 Overview of the Experiment

The helicopter flight course was aligned so that the prevailing wind crossed perpendicularly to the course. The dust concentration sensor array was situated 140 meters downwind of the flight course, aligned parallel to the course. The dust concentration data used in the visibility study were collected by DustTrak Laser photometers. These devices operate by pumping ambient air into an internal chamber and measuring 90° laser light scattering from a laser directed through the ambient air. Scattering is proportional to the concentration of particles in the air. The 15 DustTraks were positioned on three towers at five different heights, and set to measure particles with a diameter less than 10 μm (PM_{10}). An illustration of the experiment course is included in Figure 1.

The helicopter conducted 26 passes along the course centerline at various speeds ranging from 15 km/hr to 60 km/hr and maintained a skid height of 10 feet above the surface for all passes. The amount of dust entrained into the air from the helicopter was greatly dependent on helicopter speed, with large

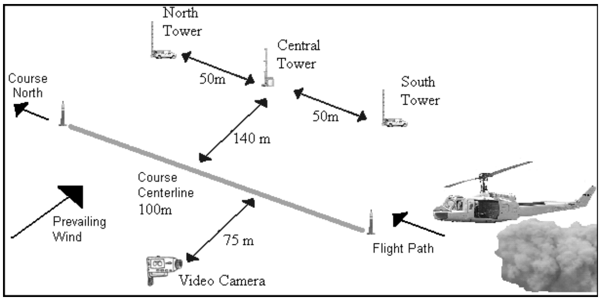


Fig. 1. Illustration of the Helicopter Dust Experiment course. The DustTrak towers are located about 140 meters downwind of the course centerline. The course centerline indicates the marked path of the helicopter flights.

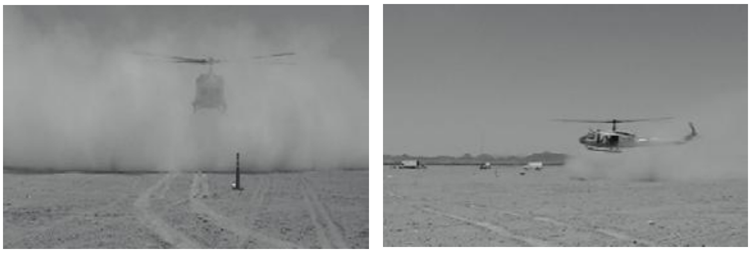


Fig. 2. Photographs of the dust entrainment during a 15 km/hr helicopter pass looking down the course centerline (left) and across the course centerline (right)

amounts entrained during the slow speeds. Figure 2 includes photographs of a 15 km/hr pass of the helicopter and the entrained dust resulting from the wake influence at the surface.

Dust entrainment was also heavily dependent on the surface wind speed and direction. Winds for the selected passes were generally perpendicular to the course centerline but varied slightly, with an average of 9° deviation from normal. Passes during winds that deviated more than 45° from normal to the course centerline were eliminated from the visualization study in an attempt to limit error due to excessive lateral transport of the plume. 20 of the 26 passes were selected for the study.

The video images were collected using a Canon ZR200 Digital Video Camcorder. The camera was upwind of the course, directed normal to the course centerline so that the three measurement towers were centered in the video. Video data were collected at 30 frames per second and output into 480 × 720 pixel frames.

3 Visualization Analysis Method

A time integration of concentration during the entire passage of the dust plume past the measurement towers is a dependable measure of the plume dimension and magnitude. For each pass, dust concentration was integrated for the period of the plume passage for each instrument and averaged for each tower. The integrated concentration, referred to as "plume strength", is determined from the following equation for each tower:

$$P\left(\frac{mg}{m^2}\right) = V\overline{\int_{t_{ps}}^{t_{pe}} C\left(\frac{mg}{m^3}\right) dt} \qquad (1)$$

where P is the plume strength integrated from the start of the concentration pulse to the end of the pulse (where concentration values return to background values). C is concentration, and V is the background average wind speed at 10 meters. Wind speed was recorded at several different heights on the central tower and also with a sonic anemometer at the center of the flight course, and averaged about 4 m/s for the period. It was noted that dust measurements varied little in magnitude with height, indicating that the plume was rather uniformly dispersed before reaching the measurement towers. This adds to the credibility of averaging the five sensor data on each tower and the prospect of using the video frames to assess visibility changes.

Video frame data used for the analysis were selected based on the time of impact of the plume at the measurement towers. The second of initial contact of the plume with the tower (often indicated by a sharp spike of dust concentration) was selected for the frame of analysis. The 30 frames from the impact second were averaged to minimize the effect of camera noise. The visual data used were in the standard RGB [1] format, with 256 settings for each color.

Three video frame boxes were selected with each frame centered on one of the three different measurement towers. At the moment selected for each pass analysis the plume is entirely in between the camera and the measurement tower of observation. Therefore, all visual degradation is due to dust scattering light from the tower and background. This method does result in a certain degree of uncertainty due to variations in wind direction and the angle of view. The first set of analysis frames are centered on the instrument tower bases, and referred to as the "base frames." The base and mountainous background provide a high degree of differing colors and sharp lines for comparison and are 40 $\times$ 70 pixels in dimension. The second set of frames is centered on the top of the north and south instrument towers, containing the top 3 measurement instruments for each tower. This second set provides a nearly uniform blue-sky background, and each frame is 35 $\times$ 35 pixels in dimension. The entire experiment was conducted in early afternoon on a clear day in late May at about 33° latitude.

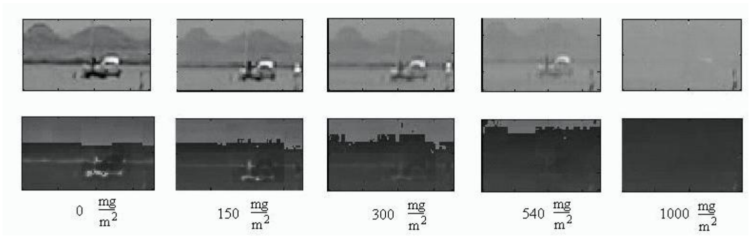


Fig. 3. Selected north tower frames in RGB (upper) and HSV (lower) for a range of plume impact strengths. The $0\frac{mg}{m^2}$ frame represents the clean-air background case before each pass.

For the analysis, color differences between the plume impact frame and a clean-air frame at a moment directly before the helicopter pass (again, a 30 frame average) are taken. The color differences for the entire frame are averaged. Red, green, and blue channels are all analyzed separately. Hue was also analyzed for the uniform blue-sky background frames, but no significant trend

[1] Red/Green/Blue.

was identified due to the compression noise (blockiness) present in the image. Standard deviation of color is also analyzed for the base frames to ascertain the change in sharpness of the image with different plume strengths. Figure 3 includes both RGB and HSV [2] frame examples of the north measurement tower for a range of plume strengths measured during the experiment.

4 Base Frame Color Results

For each tower RGB colors were averaged for the 40×70 frames centered on the tower both during the plume impact and immediately before passage of the helicopter. Average color differences between the impact-frame and the before-frame were calculated for each pass. The background color averages were nearly identical for all three towers for all passes. Therefore, color differences could be used directly instead of a normalized color difference. It is expected that the relationship between color difference and plume strength is non-linear since we would expect an approach towards a “brown-out” condition where the entire background is obscured. Therefore second-order polynomial regression was used for each set of results. When brownout is reached, color difference is assumed to reach a steady state where with higher concentrations, the polynomial regression is no longer valid.

The results for the red color difference for all of the passes are included in Figure 4a. The relationship is fairly good for color vs. plume strength with an R value of 0.88 and R^2 value of 0.78. Figure 4b includes the results for the green color differences. Again, we have a generally good relationship with an R value of 0.90 and an R^2 value of 0.81. Figure 4c includes the results for the blue color differences. The relationship is also good for this color, fairly strong with an R value of 0.87 and R^2 value of 0.76. Figure 4d includes the results of color difference magnitude versus plume strength, where the color difference magnitude is the sum of the red, green, and blue color difference for each pass. The relationship is similar to the separate colors with an R value of 0.90 and an R^2 value of 0.81.

5 Base Frame Sharpness Results

Another important aspect of visibility is sharpness. The dust cloud will tend to blur the background so that object edges are less discernable. A simple method to compare the sharpness of two images of an identical location is to take the average standard deviation of color over the frame. The difference in average standard deviation between the two frames is related directly to the difference in sharpness. The differences in average standard deviation were calculated for the base background frames and the base plume impact frames and the results analyzed again using a second-order polynomial regression.

[2] Hue/Saturation/Value (or Intensity).

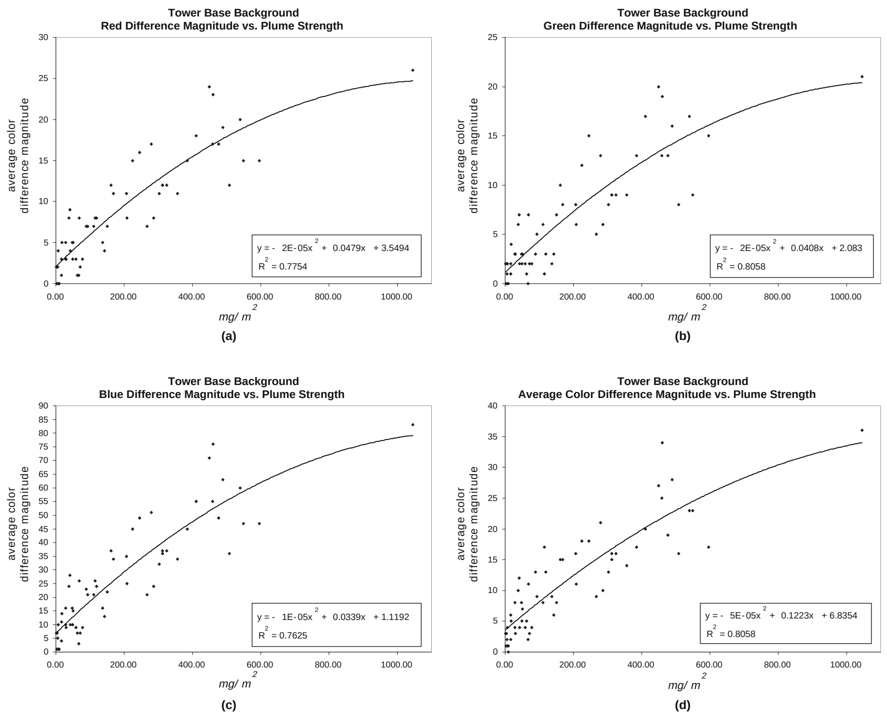


Fig. 4. Tower base frame color difference for red (a), green (b), blue (c) and average (d) *vs*. plume strength. 60 points total for each.

Unlike color, average standard deviation varied greatly depending on the objects in the image. Therefore, the standard deviation required normalization: differences were normalized by the background average standard deviation. The calculation was performed for each color and averaged over the three colors. The results are presented in Figure 5. The relationship is a bit weaker than those of the color analysis: the R value was 0.84 and the R^2 value was 0.70.

6 Uniform Frame Color Results

It was hypothesized that color differences may be better characterized against a uniform blue background. Therefore, extra frames were selected centered on the tops of the north and south towers, where the solid sky background behind the tower dominated the frame. The central tower was lower in height than the north and south tower, so that its peak was only slightly above the mountain-background in the frames. Therefore, only the north and south tower datasets were used in this analysis.

Again, average color values for red, green, and blue were calculated for the frames. The difference was taken between the plume impact frame and back-

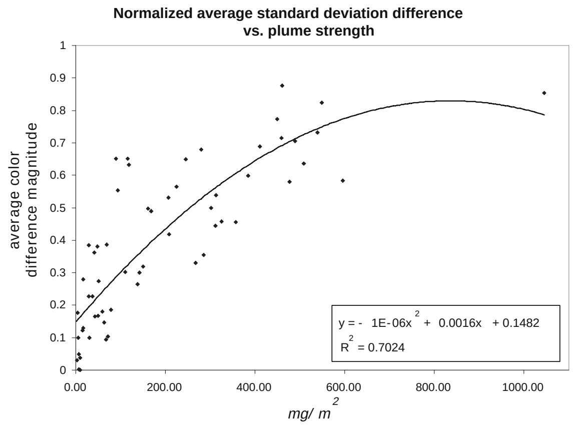


Fig. 5. Tower base frame average standard deviation differences vs. plume strength. 60 points total.

ground frame. It was noted that the green color averages were very similar in all frames. Red and blue however were quite different with a higher red value with more dust, and a lower blue value with more dust, as we would expect with blue sky being obscured by the reddish dust.

The results of color difference magnitude (blue and red only) are included in Figure 6. Data from Pass 1 were left out due to extreme differences in color intensity, likely due to the lack of background dust in the air before the experiment: the data would need to be normalized for the comparison. The Pass 1 data represents the top 5% of outliers considering it is 2 points from the 40 available.

Again, a second order polynomial regression was conducted, resulting in an R value of 0.91 and an R^2 value of 0.83. The red color difference regression resulted in about the same R and R^2 value, 0.90 and 0.80 respectively. The blue color difference regression is not as good, with an R value of 0.86 and an R^2 value of 0.74.

7 Visualization Algorithm

The linear regressions of the various analyses indicate a good relationship between observed video characteristics and the plume strength. It is therefore evident that visibility degradation can be characterized based on the concentration of dust in the air. Using the regressions, an algorithm can be established that converts a clean-air image to an obscured image based on PM_{10} concentration.

Using the base frame color difference magnitude algorithm as a guide, it is evident that peak color differences are reached (brown-out conditions) when the plume strength is about 1200 mg/m^2. The brownout condition, based on our color data, is best resembled by a constant background of red at 165, green at 140, and blue at 135. These color values would be different depending on the

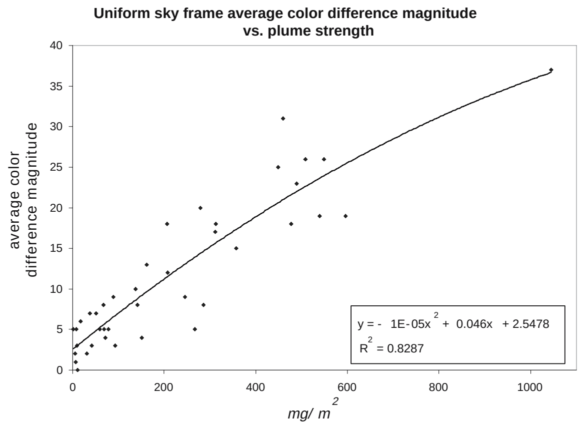


Fig. 6. Uniform sky frame average color difference magnitude (both blue and red) vs. Plume Strength. 38 points total.

dust material and lighting conditions, but the general rate of loss of visibility will be independent of lighting in most conditions.

The visibility alteration algorithm developed alters the frame pixel color based on concentration. The higher the concentration, the more the pixels are altered towards the brownout pixel conditions specified above. The gradient of change of color of each pixel is based on the regression equation found for the base frame color difference magnitude (Figure 4d), normalized so that the peak color difference=1 and no color difference=0.

For each pixel in a specified frame, the difference for each color from the brownout color is determined. Based on the concentration value and regression equation, the difference is multiplied by a scale and added to the pixel. Higher concentrations will lead to the pixel color approaching the brown-out color condition. The algorithm is as follows:

$$K_{new} = K + (K_{brown} - K)(c_1 C^2 + c_2 C) \tag{2}$$

Where K_{new} is the new color of the pixel, K is the original pixel color, Kbrown is the specified brown-out color mentioned above, C is the plume strength in mg/m^2 , and c_1 and c_2 are the regression components: $c_1 = -6.3 \times 10^{-7}$ and $c_2 = 0.00159$.

This equation was applied to the background base north tower frame with different plume strengths. These results are compared qualitatively to actual video frames of the same base north tower frames with similar plume strengths. The results are included in Figure 7. The error between the computed frames and observed frames is low from both a frame average and a pixel-to-pixel frame average. The error results are included in Table 1. Error was calculated for the frame average standard deviation by considering the difference from the background frame standard deviations for both the observed and calculated standard deviations. Color increment differences are the average (between the three colors)

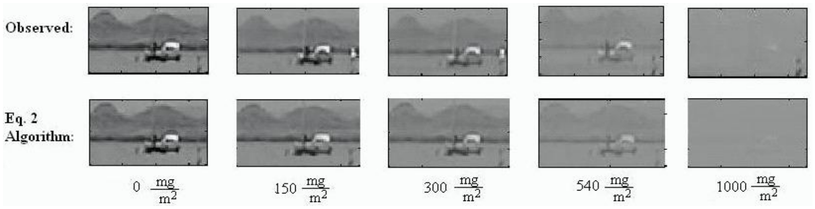


Fig. 7. Comparison between video data (upper) and Eq. 2 visibility algorithm (lower) for a range of plume strengths

difference of calculated vs. observed average frames. Pixel-to-pixel calculations were conducted by comparing each pixel from the observed frame to those of the calculated frame. The mean and standard deviations for the pixel-to-pixel average calculations are included in Table 1.

Table 1. Errors in increment of color averaged between the three color bands and associated error percentages between the observed frames at different plume strengths and the Eq. 2 algorithm

Plume Strength (mg/m^2)	Frame avg. stdv. error	Frame avg. color incr. diff.	Pixel avg. color incr. error	Pixel avg. color error	Pixel avg. color incr. stdv.
150	4.0%	0.0	4.0	3.6%	4.1
300	4.3%	1.3	4.3	3.4%	4.5
500	1.9%	2.0	5.0	3.7%	5.2
1000	1.1%	4.0	4.3	3.3%	4.9

8 Conclusion

The results of the DRI Helicopter dust experiment were used to analyze visibility degradation based on measured plume strength. A good relationship between dust plume strength and pixel color alteration was derived. This relationship was used to develop an algorithm to simulate visibility degradation based on knowledge of the line of sight plume strength.

This algorithm can be used in a visual computing environment to resolve the background visibility based on dust concentration data output from a dispersion model. A coupled Computational Fluid Dynamic (CFD) and Lagrangian Particle Model (LPM) modeling approach is being developed at DRI to simulate helicopter flight and associated dust entrainment and dispersion near the desert surface.

The output concentration field can be used to compute the visual plume strength based on the position of the observer. The plume strength of dust for each pixel can be calculated based on the line of sight through that pixel. Concentration integrated over distance results in a plume strength (mg/m^2):

$$P\left(\frac{mg}{m^2}\right) = \int_{x_o}^{x_{object}} C\left(\frac{mg}{m^3}\right) dx \tag{3}$$

where P is plume strength, and C is the concentration. The visibility algorithm (Eq. 2) can then be used to simulate the local visibility degradation due to dust entrainment. It should be noted that this method should only be used for local visibility degradation due to desert dust entrainment. Larger scale urban air quality is much more dependent on particle composition, pollutant gases, and $PM_{2.5}$ particle concentration. Further work is being conducted on this project to test the universality of method.

Acknowledgements

This material is based upon work supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract number DAAD 19-03-1-0159. This work is part of the DRI Integrated Desert Terrain Forecasting for Military Operations Project.

The authors also express special appreciation to Dr. Jack Gillies (DRI) for assistance with analysis of the dust concentration data. We would also like to acknowledge the contributions from the Strategic Environmental Research and Development Program (SERDP), Sustainable Infrastructure Project SI-1399, of logistical support in the field to J.D. McAlpine and the dust concentration data used in our analysis. The team would also like to express their gratitude to the Natural Environments Test Office, Yuma Proving Ground, Yuma AZ for financial and logistical support of the helicopter flights.

References

1. Gillies, J., Etyemezian, V., Kuhns, H., Engelbrecht, J., Uppapalli, S., Nickolich, G.: Dust emissions caused by backblast from department of defense artillery testing. J. Air & Waste Manage. Assoc. (accepted 2007)
2. Moosmüller, H., Varma, R., Arnott, W., Kuhns, H.D., Etyemezian, V., Gillies, J.: Scattering cross section emission factors for visibility and radiative transfer applications: Military vehicles traveling on unpaved roads. J. Air & Waste Manage. Assoc. 55, 1743–1750 (2005)

Visual Print Quality Evaluation Using Computational Features

Tuomas Eerola, Joni-Kristian Kamarainen, Lasse Lensu, and Heikki Kälviäinen

Machine Vision and Pattern Recognition Research Group
Department of Information Technology
Lappeenranta University of Technology, Finland
`firstname.lastname@lut.fi`
`http://www.it.lut.fi/ip/research/mvpr/`

Abstract. The ultimate print quality evaluation is always based on end-users' "quality experience", and therefore, the main challenge in automatic evaluation is to model the visual path and cognition process from physical properties to the experience. The present efforts to automate print quality evaluation have been concentrated on automating the current manually-performed assesments, which reduces the laborious work, but does not provide any novel information. In this work, a new approach for automating the evaluation is proposed and the approach is utilised by defining new computational level features which are able to explain visual quality evaluations performed by human experts.

1 Introduction

Measuring visual quality of printed products is a challenging problem since the ultimate quality evaluation can only be given by a human, end-user, who evaluates the print quality qualitatively, subjectively and dependent on the context (product type, cultural factors, etc.). Understandably, the evaluation has been traditionally conducted by a human observer, but recent development in computer and machine vision has made it intriguing to apply their methods to print quality evaluation. Machine vision operates on visual information reliably and may replace humans in certain laborious off-line evaluations. In addition, computational methods provide novel possibilities for on-line measurements for paper manufacturing and the printing industry.

To develop methods for automated quality assessments, robust models of the human visual system are needed. However, efficient and accurate computational models for the human visual perception and cognition are not currently available at the required level. Therefore, the current efforts to automate print quality evaluation have been based rather on what can be measured than what should be measured. Especially, the machine vision researchers have been interested to automate the current manually-performed assessments which can be described as low-level visual cues of the total quality. Such visual cues are, for example, unevenness of solid printed areas (mottling) and missing print dots in raster patterns (Heliotest). Detecting, localising and measuring the missing dots can be performed by the current machine vision methods (e.g., [1, 2, 3]), but the human visual system must be at least at the low level modelled in the evaluation of the unevenness of solid printed areas (e.g., [4, 5, 6]). The before-mentioned and similar manual evaluations in regular use can be replaced by machine vision hardware and algorithms. This, however, would only reduce the laborious work, and it would not help in the quest for evaluating visual print quality as a whole. Machine vision should be able to do much more than just ease the manual tasks, however.

In this study, the authors provide a novel approach for the evaluation of visual print quality by using machine vision. First, the currently recognised factors affecting the total print quality are divided into two different levels: physical level and visual level. The physical level consists of accurately and reliably measurable physical quantities whereas the visual level measures always involve a human observer. Second, the authors introduce a computational level between the two levels mentioned above. At the computational level, the upper level visual quality measures are computationally estimated using measurements performed at the physical level, e.g., by a camera. Third, this new conceptual approach is implemented by introducing computational level features which are easy to measure. As a novel result, we claim that these computational level features represent visual quality more comprehensively than the low-level visual cues in current use. To validate our claim, we estimate two low-level visual cues, namely mottling and Heliotest, using the proposed features. In the experimental part of the work, it is demonstrated how different combinations of the features can represent two distinctly different visual cues. The main contributions of this report are as follows: a new approach to divide the print quality evaluation into different levels, and a set of computer vision based features for the computational level. The results can be used to automate the manually measured low-level cues, or completely replace them by the proposed more descriptive features.

2 Print Quality

Print quality should describe the final result after printing on some media, e.g., the quality of a printed image. Good print quality cannot be defined unambiguously as it is always a subjective opinion of the observer. In addition to the paper properties, many other factors, such as the printing process, may have a significant influence to the final result [7]. Unfortunately, effects of these technically motivated quality factors to the actual perceived print quality are unknown. Therefore, we abandon this "technical quality factorisation" in this study and cover the quality by abstract levels which represent the information flow from the physically measurable quantities to the human visual experience. In our model, higher level evaluations define what is important and relevant information to be measured at lower levels.

2.1 Levels of Print Quality Evaluation

In this study, we propose three levels to be used in the development of automatic print quality assessments utilising machine vision: physical, computational and visual levels. The physical level consists of physically measurable quantities, and the visual level of human measurable visual quantities. In between the two levels is the computational one, which should be able to computationally connect the two.

Physical level properties, such as thickness, surface strengths, roughness and gloss, are well known in the paper physics and printing science [7, 8]. Measurements of the physical properties can be implemented and they have been shown to contribute to print quality. Visual properties are less strictly defined, but they can be measured quantitatively with certain limitations. Numerical values can be defined, for example, for "average observer" and "99% of the observers" to measure visual level properties, such as visual thresholds (e.g., smallest visible density, colour, or detail difference), and difference to the optimal visual quality (best possible appearance of an image) [9]. Laboratory test for visual measures (e.g., mottling and Heliotest) are usually compromises between the physical level measures and a full-scale visual quality evaluation.

Computational level measures and properties can be established computationally by directly measuring certain physical properties, or indirectly by measuring visual effects, e.g., by machine vision and image processing methods. In principle, the computational level is able to measure everything what a human observer can measure, but the difficulty is in modelling physiology and the actual visual cognition. One of our contributions presented in this study is to reveal some dependencies (correlations) between the computational level and (low) visual level.

Our novel hierarchy is illustrated in Fig. 1, where the vertical axis represents the levels, and the horizontal axis the complexity of realising a measure. In the case of physical measures, the complexity can be seen as the complexity of physical scale measurements (e.g., from nanoscale details to microscale details), or as the dimensionality of measurement itself. In the case of visual measures, the complexity refers to the objectivity or exactness of the evaluation task. For example, counting missing dots should be a less complex cognitive process than evaluating the ultimate quality of a versatile visual stimulus. The quality experience is the most essential part of quality, and it can be seen as the highest level. In practice, the most accurate way to measure the quality experience would be to use a jury of human evaluators to compare and judge different samples. Also the jury procedure has its problems. For example, different individuals experience quality differently, and thus, the result may depend on, for example, their background. In addition, the highest level visual quality depends on the end use. For example, a glossy paper may look visually impressive, but if text is printed on it, the text can be difficult to read. All these factors are included to the quality experience.

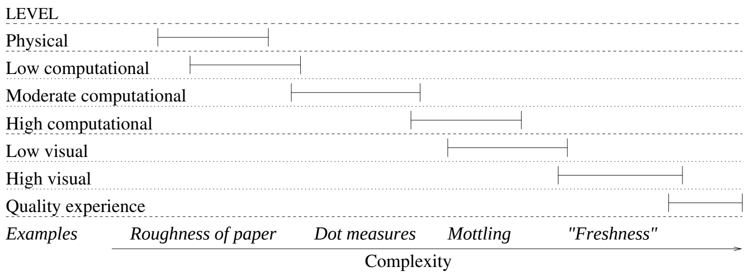


Fig. 1. Levels of print quality evaluation

2.2 Current Evaluation Methods

By referring to the concepts in the diagram in Fig. 1, the industrial print quality measures in current use are within the low visual level. This kind of measures are, e.g., very widely used Heliotest and mottling. These measures are typically conducted by a human evaluator, but are only partly subjective by their nature (e.g., "count the number of missing dots in a raster pattern"). As will be demonstrated in the following sections, such measures can be quite accurately estimated from the moderate and high computational level properties.

If the computational level measures can explain the low level visual measures, which are assumed to explain some parts of the ultimate print quality, then we can build a feed-forward chain to approximate even the higher level visual quality experiences using physical measurements and computational methods. The idea of the feed-forward chain is depicted in Fig. 2 where the continuous lines represent strong or even exact relations, and the dashed lines denote possible paths to estimate quality properties at different levels. In our chain, ϵ_1 and ϵ_1 refer to still unknown attributes affecting the print quality.

3 Computational Level Features

In this section we describe a set of computational level features, referred to as dot measures, which can be measured using standard machine vision hardware. After introducing the features, we explain the necessary image processing methods to compute them.

3.1 Dot Measures

The following computational level measures are motivated by the fact that most printed products are generated by printing dot patterns (halftoning), and therefore, we claim that properties of the individual dots can explain the visual level quality properties (Fig. 2). If a human evaluator cannot see the unevenness of plain paper, and effects of lighting and image content are not taken into account, then everything the observer experiences from the print are consequences of the

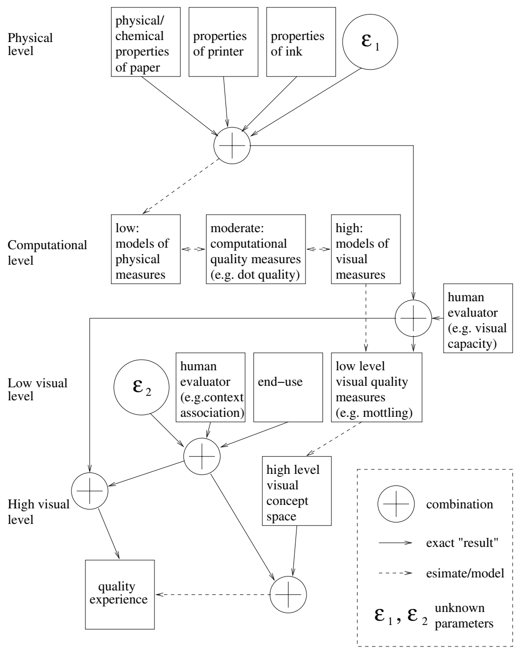

Physical level
physical/ chemical properties of paper
properties of printer
properties of ink
ε1
Computational level
low: models of physical measures
moderate: computational quality measures (e.g. dot quality)
high: models of visual measures
human evaluator (e.g. visual capacity)
Low visual level
ε2
human evaluator (e.g.context association)
end−use
low level visual quality measures (e.g. mottling)
High visual level
high level visual concept space
quality experience
combination
exact "result"
esimate/model
ε1, ε2 unknown parameters

Fig. 2. Relationships between the physical, computational and visual level quality

printing dots. Thus, in theory, it is possible to estimate the human experience from properties of individual dots. Furthermore, if the printing process is considered as sufficiently stable and even, then dots in a small region can sufficiently represent average behaviour over the whole print.

The proposed dot measures are based on geometrical, edge and size properties of the print dots. The basis of the measures is an ideal dot, which is defined as a sharp and perfectly round dot without any extraneous spots (satellites) outside its area [10]. Certain similar dot measures have been discussed in [7,11], and the measures proposed in this study are as follows:

$$\begin{aligned}
\text{stretch} &= \frac{abs(a-b)}{\min(a,b)} * 100\% \\
\text{consistency} &= (1 - \frac{y}{x+y})100\% \\
\text{regularity 1} &= \frac{s_{ref}}{s} * 100\% \\
\text{regularity 2} &= \frac{A_{ref}}{x+y} * 100\% \\
\text{coarseness} &= \frac{s^2}{4\pi(x+y)} * 100\%
\end{aligned} \quad , \tag{1}$$

where a is the width of the dot in printing direction, b is the height, x is the area, y is the area of the holes inside the dot, and s is the perimeter of the dot. $s_{ref} = 2\pi\left(\frac{a+b}{4}\right)$ and $A_{ref} = \pi\left(\frac{a+b}{4}\right)^2$ are the estimated perimeter and area, which are based on the assumption that the ideal dot is perfectly round. The radius of the dot is estimated using the width and height of the dot (average of height and width divided by 2) and the estimated area and perimeter are computed using the formulas for the area and circumference of a circle ($A = \pi r^2$, $p = 2\pi r$). The measures in (1) are selected among many other area descriptors due to their correspondence to real defects appearing in the industry.

The edge sharpness of a dot is defined as the width of the edge area. The edge area can be found by analyzing the edge profile of the dot. The edge area is the region where the gradient magnitude of the edge profile is higher than a given threshold [11]. The variation of the dot size in a solid printed area can also be used as a quality measure (in an ideal print, the size should not vary) [12].

3.2 Image Processing Methods

To compute the proposed dot measures in Sec. 3.1 we need to solve two image processing problems: 1) detection of regular structure, i.e., spatial spacing of print dots and 2) segmentation and contour description of the dots. The detection of the regular structure as prior to the segmentation provides much more robust results than the direct segmentation and is a computationally efficient approach. The main method to detect the regular structure is presented by the authors in [3], but in the following we describe how the method is applied to accurately localise every dot and describe a sub-pixel method for dot segmentation and contour description.

In the method for detecting the regular structure, the Fourier spectra magnitude is used to to find frequency peaks responsible for the regular dot pattern structure in an input image [3]. A mask is applied to remove the non-peak frequency information (responsible for distortions in shape of dots, missing dots and image noise). After the inverse Fourier transform of the masked image, we can obtain "the ideal regular image part" (see Fig. 3). The approach is very robust against to any distortions or noise and the dots and their exact centroids can be found by thresholding the ideal regular part and computing centroids of connected (binary) components (Fig. 3(c)).

Segmentation of the dots from the original (input) images is a difficult task even after the dot centroids have been found, since the input image contains all distortions and noise. In our method, the dots are segmented one by one, and thus, they are locally "cropped" from the rest of the image before the segmentation. A dot is accurately segmented by first generating a 3-d surface of the dot (gray level on z-axis) and then using the surface to estimate the contour. The contour is an intersection of the 3-d surface and a threshold plane ($z = t$) where t is a pre-set threshold (see Fig. 4). This method is not sensitive to inaccuracies caused by the resolution limits and noise as compared to the standard thresholding and the threshold can be fixed if a proper equalisation procedure is used [13].

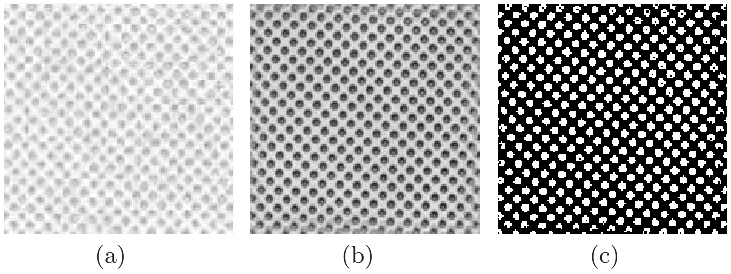


Fig. 3. Dot localization: (a) Input image; (b) Regular image part; (c) Thresholded image

After the contours are computed, width, height, area and perimeters are trivial to compute, and then, the computational level features (dot measures) defined in the previous section can be established. The dot measures are studied statistically using large amount of dots, and thus, inaccuracies caused by segmentation of single dot are insignificant.

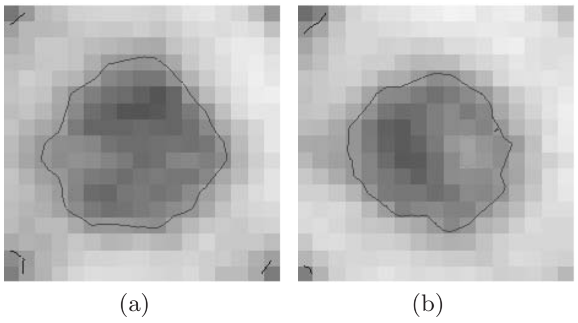


Fig. 4. The segmentation of printing dots

4 Experiments

Two separate experiments were performed to evaluate the ability of computational level features to explain visual level quality. In the first one, we estimate the number of missing dots (Heliotest assessment) using the dot measures, and in the second, human perceived solid area uneveness is estimated.

4.1 Counting Missing Dots (Heliotest)

Our test set consisted of 101 images of Heliotest strips and their expert's manually annotated ground truth (the 20th missing dot). The computational measures were computed from a region of approx. 600 dots from the lower left corner of each strip (Fig. 5). Average and variance values of the dot measures were recorded.

To perform the experiment similarly to the current industrial practice a correlation between the dot measures and location of the 20th missing dot were studied, thus resulting to a linear model. The 20th missing dot itself is a statistical measure as a single missing dot may appear randomly in a Heliotest strip, but the location of the 20th dot depends on the incidence of all missing dots, and respectively, on the paper average quality. The obtained linear model that

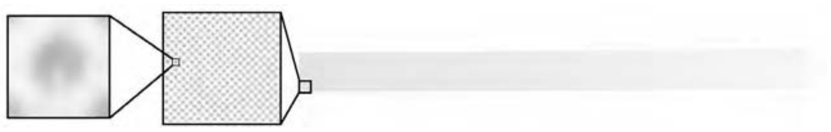


Fig. 5. Heliotest strip

predicts the location of the 20th missing dot is shown in (2).

$$\hat{y} = \begin{pmatrix} 3505 \\ 16662 \\ -3196 \\ 3911 \\ -9351 \\ 930 \\ 225 \\ -401 \\ -4733 \\ -2339 \\ -3437 \\ 3129 \\ -951 \\ -603 \end{pmatrix}^{\top} \begin{pmatrix} 1 \\ \text{avg(coarseness)} \\ \text{avg(consistency)} \\ \text{avg(regularity 1)} \\ \text{avg(regularity 2)} \\ \text{avg(sharpness)} \\ \text{avg(stretch)} \\ \text{var(area)} \\ \text{var(coarseness)} \\ \text{var(consistency)} \\ \text{var(regularity 1)} \\ \text{var(regularity 2)} \\ \text{var(sharpness)} \\ \text{var(stretch)} \end{pmatrix} \tag{2}$$

A linear model that predicts the location of the 20th missing dot ($\hat{y}$) based on the dot quality measures was obtained by defining optimal weights for every dot quality measures (averages and variances). Using the linear model, the average error between an estimated location of the 20th missing dot and the correct one was 6.78mm (9.0% of the average distance to the 20th missing dot) and the correlation coefficient value of the model was 0.74. The linear model seems to predict the missing dot phenomenon quite well (Fig. 6) and the interpretation of the factors in (2) is intuitively correct as, for example, variances of the coarseness and regularity 1, which describe unrepeatability in the dot reproduction, have strong negative factors and the average consistency a strong positive factor.

It should be noted that if the printing method changes, the model needs to be trained again, since emergence of missing dots vary between different methods.

4.2 Evaluating Unevenness

The test set consisted of 40 mottling samples. Half of the samples were printed with 70% halftone percentage (B70) and the rest with 40% halftone percentage (B40). The mottling indices (measure for unevenness of print) for B70 samples were defined by a group of human experts [5]. Only the B40 samples could be used to compute the dot measures since in B70 samples the printing dots were too large to be separated from each other. However, in the following we explain an alternative approach to compute the dot measures for dense halftone images.

For the B70 samples, dots were replaced by the holes between the dots, and for the holes it was possible to apply the methods. The approach is motivated, since if sizes, shapes and positions of dots do not change, then they should not

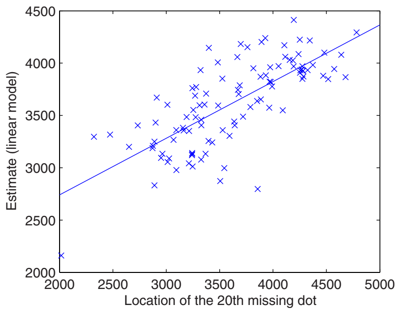


Fig. 6. Correlation diagram for the estimated and ground truth Heliotest results

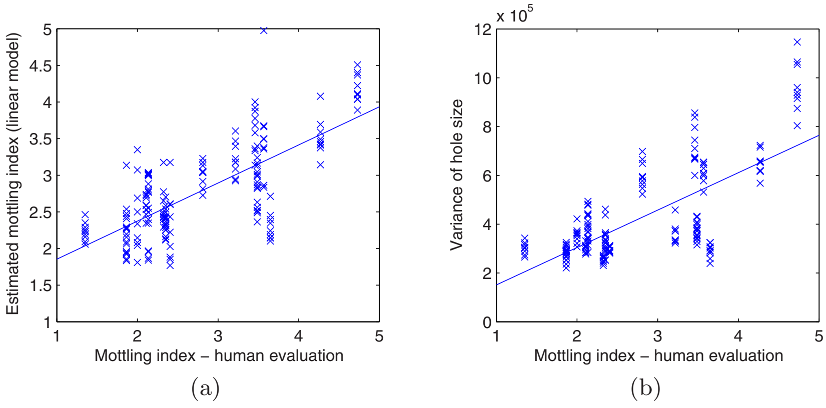


Fig. 7. Correlation diagrams for estimated and ground truth mottling: (a) B40; (b) B70

change for the holes either. The variance of hole size were used as the alternative computational level quality measures for the B70 samples.

The computational level quality measures were computed from 9 5mm x 5mm regions in each sample. One region consisted of about 750 printing dots (holes). Each region was considered as a single sample. There were thus 180 samples from both B40 and B70 sets.

Again, a linear model that predicts the mottling was obtained. The correlation coefficient between the human evaluated and estimated mottling indices ($\hat{y}$) was as high as 0.72 (see also Fig. 7(a)) being better than for many dedicatedly designed mottling estimation methods. The correlation coefficient between the human evaluated mottling indices and the variance of hole area computed from B70 samples was 0.72 (see Fig. 7(b)).

5 Conclusions

We proposed a new approach to printed media (paper) print quality evaluation. We introduced visual, computational and physical levels of quality measures, where the two extrema are the high visual level, which denotes the ultimate quality evaluation, human quality experience, and the low physical level, which denotes accurately measurable physical quantities. For establishing a connection between the levels, we proposed the computational level, for which computationally intensive measures can be realised.

As another important result we demonstrated power of the computational level by introducing a set of computational level measures, dot measures, and necessary processing steps to compute them. In the experimental part of the study, we demonstrated how the proposed measures were able to successfully predict two standard visual quality measures of halftone prints. In future, to transfer the methods from research to best practises in industry, they must be defined in the form where results do not depend on parameter selection. Also more powerful non-linear models will be studied.

As the main conclusion, this study proved that low level visual quality perceived by human evaluators can be estimated with machine vision based computational measures. By finding a similar connection between high and low visual level measures and between computational and physical level measures, the quality inspection can be shifted to a new era where computational on-line and off-line measures replace the current practises.

Acknowledgements

The work was partially supported from the following projects: PAPVISION (`http://www.it.lut.fi/project/papvision`) by European Union (Finnish Funding Agency for Technology and Innovation Project No. 70049/03, 70056/04 and 40483/05), DigiQ (`http://www.media.hut.fi/digiq`) (Finnish Funding Agency for Technology and Innovation Project No. 40176/06), and Academy of Finland Project No. 204708.

References

1. Heeschen, W., Smith, D.: Robust digital image analysis method for counting missing dots in gravure printing. In: Proc. Int. Printing & Graphic Arts Conference, Atlanta, GA, USA, pp. 29–35 (2000)
2. Langinmaa, A.: An image analysis based method to evaluate gravure paper quality. In: Proc. 11th IAPR Int. Conf. on Computer Vision and Applications, pp. 777–780 (1992)
3. Vartiainen, J., Sadovnikov, A., Kamarainen, J.K., Lensu, L., Kalviainen, H.: Detection of irregularities in regular patterns. Machine Vision and Applications (Accepted for publication)
4. Fahlcrantz, C.M., Johansson, P.A.: A comparison of different print mottle evaluation models. In: 56th Annual Technical Conference of the Technical Association of the Graphic Arts, San Antonio, USA, pp. 511–525 (2004)
5. Sadovnikov, A., Lensu, L., Kamarainen, J., Kalviainen, H.: Quantifed and perceived unevenness of solid printed areas. In: Xth Ibero-American Congress on Pattern Recognition, pp. 710–719 (2005)
6. Wolin, D.: Enhanced mottle measurement. In: PICS 2002. IS&T's PICS conference, pp. 148–151 (2002)
7. Levlin, J.E., Söderbjelm, L.: Pulp and Paper Testing. In: Papermaking Science and Technology. Fapet Oy (1999)
8. Niskanen, K.: Paper physics. In: Papermaking Science and Technology, Fapet Oy (1998) ISBN 952-5216-16-0
9. Oittinen, P., Saarelma, H.: Printing. In: Papermaking Science and Technology, Fapet Oy (1999)
10. Doyle, M.: Measuring the imperfect dot. In: IS&T's NIP16: Int. Conf. on Digital Printing Technologies, pp. 640–642 (2000)
11. Oittinen, P., Saarelma, H.: Kuvatekninen Laatu. Otatieto (1992)
12. Kipman, Y.: Image quality metrics for printers and media. In: Proc. of the PICS Image Processing, Image Quality, Image Capture, Systems Conf., pp. 183–187 (1998)
13. Vartiainen, J., Paalanen, P., Kamarainen, J.K., Lensu, L., Kälviäinen, H.: Minimum error contrast enhancement. Research report 102, Department of Information Technology, Lappeenranta University of Technology (2006)

Follow the Beat? Understanding Conducting Gestures from Video

Andrea Salgian[1], Micheal Pfirrmann[1], and Teresa M. Nakra[2]

[1] Department of Computer Science
[2] Department of Music
The College of New Jersey
Ewing, NJ 08628
salgian@tcnj.edu, micheal.pfirrmann@gmail.com, nakra@tcnj.edu

Abstract. In this paper we present a vision system that analyzes the gestures of a noted conductor conducting a real orchestra, a different approach from previous work that allowed users to conduct virtual orchestras with prerecorded scores. We use a low-resolution video sequence of a live performance of the Boston Symphony Orchestra, and we track the conductor's right hand. The tracker output is lined up with the output of an audio beat tracker run on the same sequence. The resulting analysis has numerous implications for the understanding of musical expression and gesture.

1 Introduction

In recent years, numerous artistic and expressive applications for computer vision have been explored and published. Many of these have been for dance, whereby moving dancers trigger various visual and audio effects to accompany their movements [1,2]. However, there is a small but growing area in which purely musical applications are being researched. In this area, musical conductors are frequently featured, perhaps because conductors are the only musicians who freely move their hands to create sound and whose gestures are not constrained by a rigid instrument.

Several computer-based conducting recognition systems have also relied on on tracking batons equipped with sensors and/or emitters. Most notably, the *Digital Baton* system implemented by Marrin and Paradiso [3], had an input device that contained pressure and acceleration sensors, and the tip of the baton held an infrared LED which was tracked by a camera with a position-sensitive photodiode.

Examples of prior "pure vision" applications featuring musical conducting include the work by Wilson and Bobick [4]. Their system allowed the general public to "conduct" by waving their hands in the air and controlling the playback speed of a MIDI-based orchestral score.

In another project, Bobick and Ivanov [5] took that concept further by identifying a typical musical scenario in which an orchestra musician would need to visually interpret the gestures of a conductor and respond appropriately.

More recently, Murphy *et al.* [6] developed a computer vision system for conducting audio files. They created computer vision techniques to track a conductor's baton, and analyzed the relationship between the gestures and sound. They also processed the audio file to track the beats over time, and adjusted the playback speed so that all the gesture beat-points aligned with the audio beat points.

Until recently, these vision-based techniques aimed at systems that would allow real conductors (and sometimes the general public) to conduct virtual orchestras, by adjusting effects of a prerecorded score. In contrast, the *Conductor's Jacket* created by Nakra [7] was designed to capture and analyze the gestures of a real conductor conducting a real orchestra. However, the input of this system was not visual. Instead, it contained a variety of physiological sensors including muscle tension, respiration and heart rate monitors.

In this paper we take the first steps towards devising a computer vision system that analyzes a conductor conducting a real orchestra. This is an important distinction from earlier work, because a professional conductor reacts differently when conducting a real (versus a virtual) orchestra. His/her motivation to perform the authentic gestures in front of the human orchestra can be assumed to be high, since the human orchestra will be continuously evaluating his/her skill and authenticity. There is scientific value in understanding how good conductors convey emotion and meaning through pure gesture; analysis of real conducting data can reveal truth about how humans convey information non-verbally.

We analyze the low-resolution video footage available from an experiment with an updated version of the *Conductor's Jacket*. We track the right hand of the conductor and plot its height as the music progresses. The vertical component of the conductor's hand movements, together with the beat extracted from the score, enables us to make several interesting observations about musical issues related to conducting technique.

The rest of the paper is organized as follows. In Section 2 we describe the background of our work. In Section 3 we present the methodology for tracking the conductor's hand. In Section 4 we discuss our results. Finally, we conclude and describe future work and possible applications in Section 5.

2 Background

Motivated by prior work, we undertook a joint research project to investigate the gestures of a noted conductor. Our goal was to use computer vision techniques to extract the position of the conductor's hands. Our source video footage featured the Boston Symphony Orchestra and conductor Keith Lockhart. This footage was obtained during a 2006 collaborative research project involving the Boston Symphony Orchestra, McGill University, Immersion Music, and The College of New Jersey. The Boston Symphony Orchestra and McGill University have given us the rights to use their video and audio for research purposes.

Figure 1 shows conductor Keith Lockhart wearing the measuring instruments for this experiment.

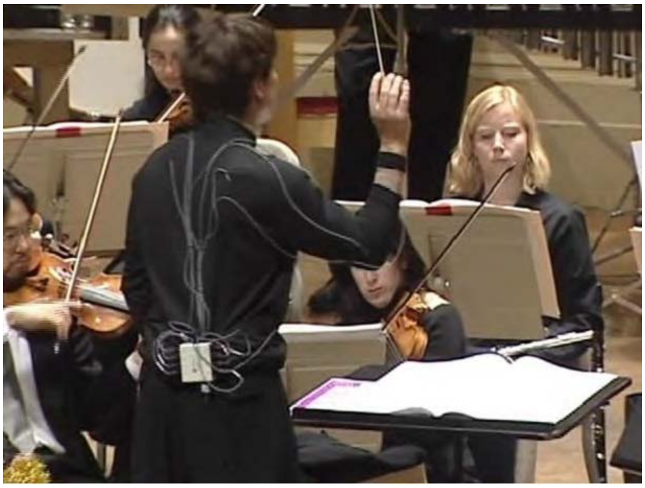


Fig. 1. Conductor Keith Lockhart wearing the measuring instruments (Photo credit: KSL Salt Lake City Television News, April 21, 2006)

The video sequence contains a live performance of the Boston Symphony Orchestra, recorded on April 7, 2006. The piece is the Overture to "The Marriage of Figaro" by W.A. Mozart, and our video has been edited to begin at the second statement of the opening theme. (The reason for the edit is that the beginning of the video features a zoom-in by the camera operator, and the first several seconds of the footage were therefore unusable. This segment begins at the moment when the zoom stopped and the image stabilized.) Figure 2 shows a frame from the video sequence that we used.

Given that image processing was not planned at the time of the data collection, the footage documenting the experiment is the only available video sequence. Hence, we were forced to work with a very low resolution image of the conductor that we cropped from the original sequence (see Figure 3).

Given the quality of the video, the only information that could be extracted was the position of the right hand. It is known that the tempo gestures are always performed by either the conductor's right hand or both hands, and therefore right hand following is sufficient to extract time-beating information at all times [8]. What makes tracking difficult is the occasional contribution of the right hand to expressive conducting gestures, which in our case lead to occlusion.

Our next task was to look at the height of the conductor's right hand - the one that conducts the beats - with the final goal of determining whether it correlated with musical expression markings and structural elements in the score. We have found that indeed, it does. The height of Keith Lockhart's right hand increases and decreases with the ebb and flow of the music.

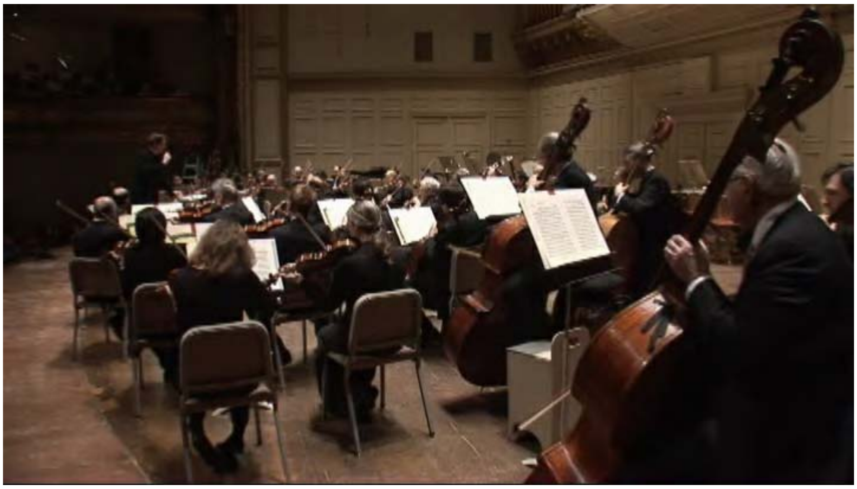


Fig. 2. A frame from the input video sequence

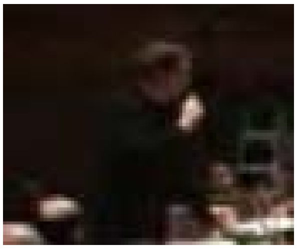


Fig. 3. The frame cropped around the conductor

3 Methodology

As described in the previous section, the frames of the original video were cropped to contain only the conductor. The crop coordinates were chosen manually in the first frame and used throughout the sequence. The frames are then converted to grayscale images. Their size is 71x86 pixels.

The average background of the video sequence is computed by averaging and thresholding (with a manually chosen threshold) the frames of the entire sequence. This image (see Figure 4) contains the silhouettes of light (skin colored) objects that are stationary throughout the sequence, such as heads of members of the orchestra and pieces of furniture.

For each grayscale image, the dark background is estimated through morphological opening using a circle of a radius 5 pixels as the structural element. This background is then subtracted from the frame, and the contrast is adjusted through linear mapping. Finally, a threshold is computed and the image is

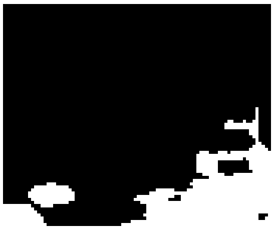


Fig. 4. The average video background

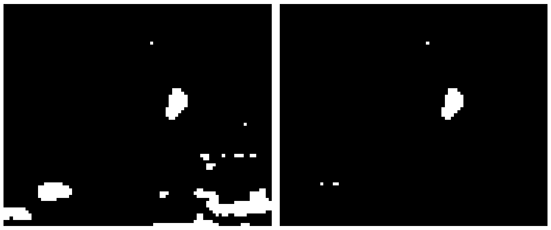


Fig. 5. Thresholded frame on the left, same frame with the average video background removed on the right

binarized. The left side of Figure 5 shows a thresholded frame. This image contains a relatively high number of blobs corresponding to all the lightly colored objects in the scene. Then the average video background is subtracted, and the number of blobs is considerably reduced. We are left with only the moving objects. An example can be seen on the right hand side of figure 5.

While in some cases background subtraction alone is enough to isolate the conductor's right hand, in other cases, other blobs coming from the conductor's left hand or members of the orchestra can confuse the result. Figure 6 shows such an example.

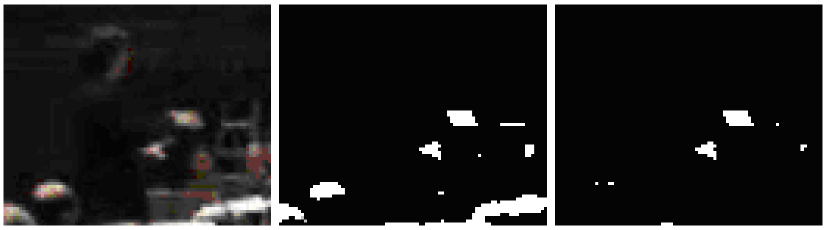


Fig. 6. Another frame, after thresholding, and background subtraction

In the first frame, the correct object is picked by the user. In subsequent frames the algorithm tracks the hand using the position detected in the previous frame. More specifically, the coordinates of the object that is closest to the previous position of the hand are reported as the new position. If no object is found within a specified radius, it is assumed that the hand is occluded and the algorithm returns the previous position of the hand. Figure 7 shows a frame with the position of the hand marked.

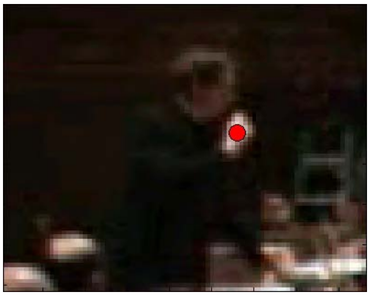


Fig. 7. Tracked hand

We then plot the vertical component of the position of the conductor's hand. Based on the conductors' gestures, the local minima and maxima should correspond the tempo of the music being played. To verify this, we extracted the beats from the score using an algorithm developed by Dan Ellis and Graham Poliner [9] that uses dynamic programming. We marked the beat positions on the same plot and generated an output video containing the cropped frames and the a portion of the tracking plot showing two seconds before and after the current frame. Figure 8 shows a frame from the output video sequence.

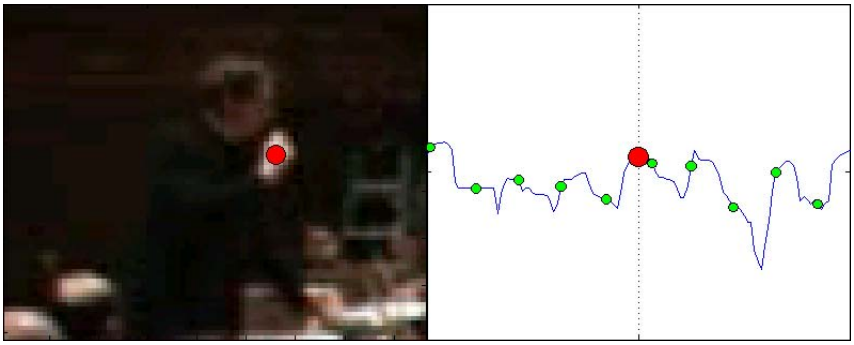


Fig. 8. A frame from the output video sequence

The left side of the image contains the cropped frame and the detected position of the hand. The right side consists of three overlaid items:

1. a vertical dotted line with a red dot, indicating the intersection of the current moment with the vertical height of Keith Lockhart's right hand.

2. a continuous dark line indicating the output of the hand tracker, giving the vertical component of Keith Lockhart's right hand
3. a series of green dots, indicating the places in time when the audio beat tracker determined that a beat had occurred

4 Results

To analyze the performance of our tracker, we manually tracked the conductor's right hand in the first 500 frames of the video and compared the vertical component with the one extracted by the algorithm. 421 of 500 frames (over 84%) had a detection error of less than 2 pixels. In 24 out of remaining 79 frames the tracking could not be performed manually due to occlusion.

Figure 9 shows the ground truth and the detected y coordinate in the first 500 frames. Ground truth coordinates that are lower than 10 pixels correspond to frames where the hand could not be detected manually. Horizontal segments in the detected coordinates correspond to frames where no hand was detected. In these situations the tracker returns the position from the previous frame.

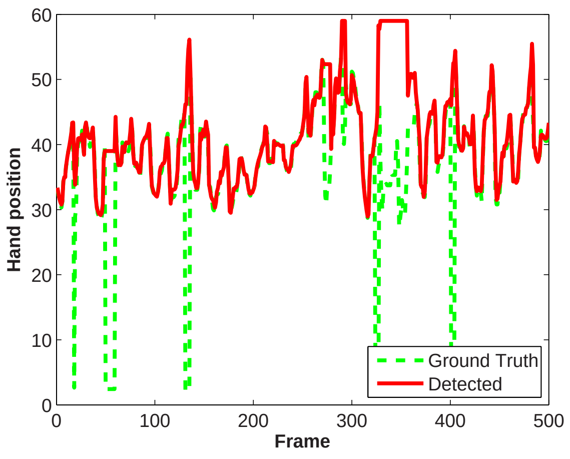


Fig. 9. Tracking performance on the first 500 frames

In the relatively few situations where the tracker loses the hand, it has no difficulty reacquiring it automatically.

Figure 10 shows the vertical component of the right hand position in blue, and the beats detected in the audio score in red. It may seem surprising that there is a delay between the local extrema of the conductor's hand and the audio beats. This is mostly due to the fact that there is a short delay between the time the conductor conducts a beat and the orchestra plays the notes in that beat. (This well-known latency between conductor and orchestra has been quantified in [10] to be 152 +/- 17 milliseconds, corresponding to one quarter of a beat at 100

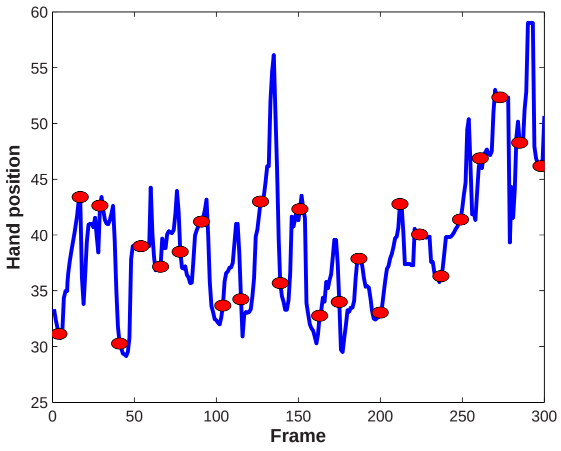


Fig. 10. Hand position and beat in the first 300 frames

beats per minute. This study is based upon conductors listening and reacting to a recording, which may have biased the data.) It should also be noted that in the current study, there are places where the conductor's beats are not in phase with the orchestra. It may be assumed that in such places, the conductor is not needed for routine "traffic cop"-type time beating, but rather motivating the orchestra to increase (or decrease) its rate of tempo change.

Using all the visual information provided by the various streams in the video, a musician can make several interesting observations about musical issues related to conducting technique. While these observations strictly refer to the technique of Keith Lockhart, nonetheless it can be assumed that some of these features may also be used by other conductors, perhaps in different ways. Some of the conducting features revealed by our method are as follows:

1. Tiered gesture "platforms" - Lockhart seems to use local "platforms" (horizontal planes) of different heights at different times in the music. The choice of what height to use seems to be related to the orchestration and volume indicated in the music.
2. Height "delta" - at certain times, the height difference between upper and lower inflection points changes. This seems also to be related to expressive elements in the score - particularly volume and density.
3. Smooth versus jagged beat-shapes - sometimes the beats appear almost sinusoidal in their regularity, whereas other times the shape of the peak becomes very jagged and abrupt with no rounding as the hand changes direction. This feature also appears to be controlled by the conductor, depending upon elements in the music.
4. Rate of pattern change - sometimes a particular feature stays uniform over a passage of music, sometimes it gradually changes, and sometimes there are abrupt changes. The quality of the change over time seems also to be related to signaling the musicians about the nature of upcoming events.

5 Conclusions and Future Work

We presented a system that analyzes the gestures of a conductor conducting a real orchestra. Although the quality of the footage was poor, with very low resolution and frequent self-occlusions, we were able to track the conductor's right hand and extract its vertical motion. The tracker was able to reacquire the hand after losing it, and we obtained a recognition rate of 84% on the first 500 frames of the sequence. We annotated these results with the beats extracted from the audio score of the sequence. The data we obtained proved to be very useful from a musical point of view and we were able to make several interesting observations about issues related to conducting technique.

There is much more work to be done in this area. Very little is known about professional conductors' gestures, and it is hoped that with more research some interesting findings will be made with regard to musical expression and emotion. Our next task will be to compare our results with those of the other (physiological) measurements taken during the experiment.

Additional data collections with higher quality video sequences will allow us to devise better algorithms that could track the conductor's hand(s) more accurately and extract a wider range of gestures.

Results of future work in this area are targeted both for academic purposes and beyond. For example, conductor-following systems can be built to interpret conducting gestures in real-time and cause the conductor to control various media streams in synchrony with a live orchestral performance. (Lockhart himself has agreed that it would be fun to be able to control the fireworks or cannons on the 4th of July celebrations in Boston while conducting the Boston Pops Orchestra.) Human computer interfaces could also benefit from understanding the ways in which expert conductors use gestures to convey information.

Acknowledgments

The authors would like to thank the Boston Symphony Orchestra and conductor Keith Lockhart for generously donating their audio and video recording for this research. In particular, we would like to thank Myran Parker-Brass, the Director of Education and Community Programs at the BSO, for assisting us with the logistics necessary to obtain the image and sound. We would also like to acknowledge the support of our research collaborators at McGill University: Dr. Daniel Levitin (Associate Professor and Director of the Laboratory for Music Perception, Cognition, and Expertise), and Dr. Stephen McAdams (Professor, Department of Music Theory, Schulich School of Music).

References

1. Paradiso, J., Sparacino, F.: Optical tracking for music and dance performance. In: Fourth Conference on Optical 3-D Measurement Techniques, Zurich, Switzerland (1997)
2. Sparacino, F. (Some) computer vision based interfaces for interactive art and entertainment installations. INTER-FACE Body Boundaries 55 (2001)

3. Marrin, T., Paradiso, J.: The digital baton: A versatile performance instrument. In: International Computer Music Conference, Thessaloniki, Greece, pp. 313–316 (1997)
4. Wilson, A., Bobick, A.: Realtime online adaptive gesture recognition. In: International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, Corfu, Greece (1999)
5. Bobick, A., Ivanov, Y.: Action recognition using probabilistic parsing. In: Computer Vision and Pattern Recognition, Santa Barbara, CA, pp. 196–202 (1998)
6. Murphy, D., Andersen, T.H., Jensen, K.: Conducting audio files via computer vision. In: Camurri, A., Volpe, G. (eds.) GW 2003. LNCS (LNAI), vol. 2915, pp. 529–540. Springer, Heidelberg (2004)
7. Nakra, T.M.: Inside the Conductor's Jacket: Analysis, Interpretation and Musical Synthesis of Expressive Gesture. PhD thesis, Media Laboratory, MIT (2000)
8. Kolesnik, P.: Conducting gesture recognition, analysis and performance system. Master's thesis, McGill University, Montreal, Canada (2004)
9. Ellis, D., Poliner, G.: Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In: Proc. Int. Conf. on Acous., Speech, and Sig. Proc. ICASSP 2007, Hawaii, pp. 1429–1432 (2007)
10. Lee, E., Wolf, M., Borchers, J.: Improving orchestral conducting systems in public spaces: examining the temporal characteristics and conceptual models of conducting gestures. In: Proceedings of the CHI 2005 conference on Human factors in computing systems, Portland, Oregon, pp. 731–740 (2005)

A Quantitative Comparison of Two New Motion Estimation Algorithms

B. Zhan[1], P. Remagnino[1], S.A. Velastin[1], N. Monekosso[1], and L.-Q. Xu[2]

[1] Digital Imaging Research Centre, Kingston University, UK
[2] British Telecom Research,Ipswich,UK
{B.Zhan,P.Remagnino}@kingston.ac.uk

Abstract. This paper proposes a comparison of two motion estimation algorithms for crowd scene analysis in video sequences. The first method uses the local gradient supported by neighbouring topology constraints. The second method makes use of descriptors extracted from points lying at the maximum curvature along Canny edges. Performance is evaluated using real-world video sequences, providing the reader with a quantitative comparison of the two methods.

1 Introduction and Previous Work

A pedestrian scene normally introduces different levels of difficulty, e.g. for visual surveilance the number of pedestrians in the field of view could varies from one to several hundred. When the scene is very complex, occlusions make it virtually impossible not only to track individuals but also to estimate stochas-tic background model. Robust and reliable descriptors must be employed to describe atomic features of underlying dynamics. The descriptors must be able to work under different levels of difficulty to extract the motion information, so as to enchance the higher level analysis of the scene dynamics. Extensive literature exists on local descriptors that have been used in in deformable object tracking [1], image retrieval applications, e.g. SIFT[2], Harris Corner[3]. Recently years it has also been applied to analysis of image sequences in sport and monitoring applications [4] [5]. Two novel motion estimation methods extended the usage of such local descriptors to esimate crowd motion are validated and compared in this paper. In both of the algorithms, constraints are applied to improve the robustness of the matching between individual descriptors. The first algorithm checks locally spatial temporal consistency of color gradient supported by local topology constraints and the second one uses the points of local extreme curvature along Canny edges and applies contour constraints. Overall dynamics of the scene can then be leant from the short term motion estimated from these algorithms. Section 2 describes the two algorithms , Section 3 gives the experimental results of both algorithms, and Section 4 draws some conclusions.

2 The Proposed Algorithms

Video sequences involving crowds are very complex to analyze, mainly because conventional background subtraction algorithms and motion estimation methods might not deliver expected results. Given the problem of interpreting crowd motion, we choose to use refined matching of local image descriptors to derive the dynamics features. Our aim is to recognise points of interests which can be tracked for some periods of time, and then combine their tracks into meaningful crowd trajectories. The following two subsections summarize the two algorithms, published in [6] [7].

2.1 Algorithm 1

The first approach, proposes a topological matching of interest points extracted in the evolving scene. Points of interest are extracted using a color variant of the Harris detector as described in [3]. The matching between frames is carried out in two steps: (i) searching of the candidate matching points by similarity and then (ii) applying the topological constrains. Briefly, for each selected interest point in the reference image, corresponding candidate matching point is searched in the matching image inside a given search window using the gradient similarity measure given by $sim(a, b) = \frac{min(R_a, R_b)}{max(R_a, R_b)}$ (given a, b as two interest points). This basic matching does not provide a robust solution, due to the instability of interest points in a highly complex scene. Frequent occlusions reduce the probability of identifying correct matches and, without local support, similar gradient localities might be found as acceptable matches generating false positives. To tackle this problem, we introduced an effective topological constraint into the search process. The necessary local support is derived from local window centred at the interest point and we make use of the relative location of the interest points in such window. Support is estimated for the matched interest point pair inside the support windows. All interest points found in the support window are then matched. Support is then quantified in terms of the error by measuring the standard deviation of the ensemble of found correspondences (see [6] for more details on the algorithm).

2.2 Algorithm 2

The results of $Algorithm 1$ for crowd dynamics measurement turns out to be good, however there is still a room for further improvement, as certain false positives still exist. The reason could be that the relationships between the interest points provided by topological constraints are still not very reliable, so we proposed another method [7] using local descriptors which provide additional information. We choose Canny edge information and extracted the curvature along each edge to retrieve descriptor points (those with local maximum curvature) as salient features along an edge. Besides the points, edge information is maintained by "edgelet constraint" to refine the estimate. Thus, we combine the advantage of using point features that are flexible to track with the advantage of using edge features that maintain structural information. Each Canny edge is a chain of points S_p, and all the edges are stored in an edge list L_p. It can be observed that even in a scene which depicts a crowd of moderate density, edge chains can occlude one another, increasing the descriptor matching complexity. Given two consecutive frames, we estimate the motion of the extracted local descriptors (matching within a window of interest).The best n matches are then selected as candidate points and considered for the next step. For an image frame, we divide every chain S to a uniform length edgelets represented by sub sequences E. There are two reasons for doing this: to avoid a very long edge that could be generated by several different objects, and to enhance the matching of the edge fragments that are generated by occlusions. For each local descriptor in E we have as result from the first step, the n candidate matching points. Each candidate point belongs to a sub sequence S. To find the best match of E, we use three parameters: energy cost, variation of displacements and the match length for each candidate and combine them into a single matching score. Here we assume that the length of E is small enough so that it would not split again to two or more matches. This is so that their candidate points correspond to the same candidate sequence. Details on the second algorithm can be found in [7].

3 Comparison of the Two Methods

The two motion estimation algorithms are tested using three sequences taken from crowded public space and quantitative results are generated. As we are concerned about extraction of short term motion instead of tracking, measures are defined based on two consecutive images instead of whole sequence. In the following we first give a brief description of the test dataset used in our experiments, then go through the details of the testing methods adopted and explain the results generated from the tests; some visual results are also included at the end of the section.

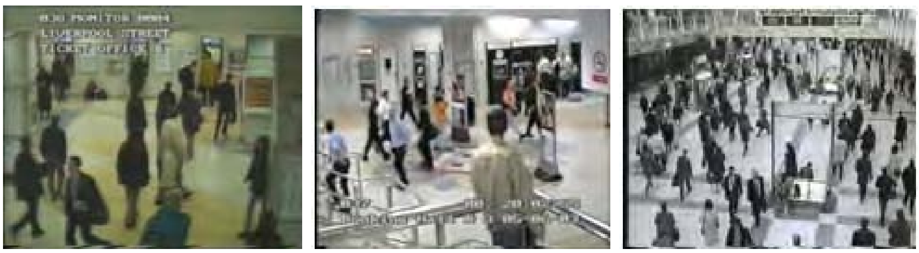


Fig. 1. Sample frames from 3 testing sequences

3.1 Testing Data

Sample frames from the three sequences are shown in Figure 1: sequence 1 (left) is a mid field scene with people scattered across the field of view; sequence 2 (middle) is a mid field scene with major motions taking place in certain areas; sequence 3 (right) is a far field scene with pedestrians present in all parts of the field of view, with some predominant trajectories.

3.2 Testing Based on Local Descriptors

In this testing only the quality of matching of individual local descriptors is considered. For each pair of consecutive frames, local descriptors in the initial frame are compared with their corresponding local descriptors, found by the two presented algorithms, in the target/second frame, respectively. Two measures, Mean Similarity (MS) and Mean Absolute Error (MAE), are used here. MS is designed to assess the relative similarity of the matched local descriptors.

$$MS = \frac{1}{n}\sum_{i=0}^{n}\frac{min(X_i^{t_0}, X_i^t)}{max(X_i^{t_0}, X_i^t)} \tag{1}$$

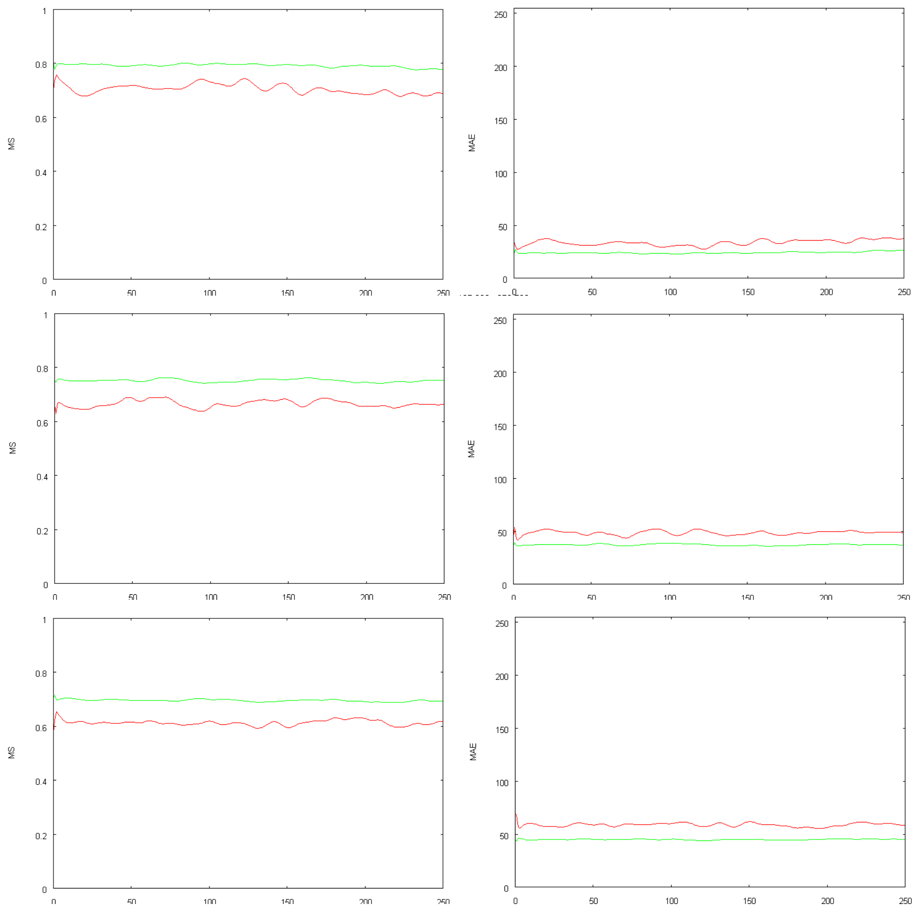


Fig. 2. MS (left column) and MAE (right column) along time for the 3 testing sequence (From top to the bottom: sequence 1, sequence 2 and sequence 3), red lines for Algorithm 1; green lines for Algorithm 2. Algorithm 2 keeps higher in MS and lower in MAE.

Where n is the total number of the local descriptors in the initial frame. MS is defined as the average of the similarity, and the similarity is calculated by the minimum of the two matched local descriptors' pixel value divided by the maximum. The result is a value which falls in the $(0, 1)$ range. Another measure, MAE is commonly used for the testing of motion estimation algorithms [8] as it returns an error measure. MAE is defined as follows

$$MAE = \frac{1}{n} \sum_{i=0}^{n} \| X_i^{t_0} - X_i^t \| \tag{2}$$

Where $X_i^{t_0}$ is the pixel value at the i^{th} corner in the first frame, and X_i^t is the cor-respond local descriptor in the next frame.

The images in Figure 2 represent the plots of MS and MAE for the two algorithms tested against the three sequences. MS and MAE are calculated every frame along the sequence. In each plot the x axis represents time (the number of the frame) and the y axis represents the values of MS and MAE, respectively. Hence, for the two algorithms the MS and MAE for the three testing sequences are both good, though in most of the cases the second algorithm has a higher MS and a lower MAE. Also, along the time scale the MS and the MAE produced from the first algorithm fluctuate a lot while the second one produces more stable results. It can be concluded that the second algorithm has a more desirable performances than the first one.

3.3 Testing Based on Motion Connect Component

The testing here makes use of connected components algorithm based on motion vectors (so called MCC – Motion Connected Component). The algorithm groups together motion vectors that are in close proximity and have common motion properties. The result of the MCC algorithm segments the motion field into clusters of uniform motion group (e.g. a (part of) pedestrian or a group of pedestrians), and the testing is based on each MCC to assess the two algorithms. In order to assess the two algorithms with MCC, we adopted two measures which are used in evaluating search strategies: Recall and Precision. Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved [9]. In the proposed implementation we take the bounding box of each MCC, and calculate the average motion of MCC, thus map the bounding box to the next frame. The number of "relevant records in the data base" should be the number of local descriptors of the MCC in the initial frame (N_{t_0}) while the number of "retrieved records" should be the number of local descriptors in the mapped bounding box in the second frame (N_t). The definitions of the two measures are given by

$$Recall = \frac{N_{t_0} \cap N_t}{N_{t_0}} \tag{3}$$

$$Precision = \frac{N_{t_0} \cap N_t}{N_t} \tag{4}$$

Both of the values are in the range $[0, 1]$. For every frame an average Recall value and an average Precision value are calculated. Figure 3 gives the plots of Recall and plots of Precision; the layouts of these plots remain similar to the previous ones, though the y axis represents Recall and Precision, respectively. From the plots again we can claim the results of Recall and Precision of both of the algorithms, especially the results of Recall, are satisfied. It can be observed that the results of Precision for sequence 3 is lower than the other three, one possible reason could be that as sequence 3 is a far field view for a crowded scene, when mapping the bounding box of the MCC to the second frame local descriptors of other MCC could be included and noise could be introduced. When comparing the results of Recall, it can be seen values for Algorithm 2 are always higher, though for sequence 2 and sequence 3 Precision values for Algorithm 1 are

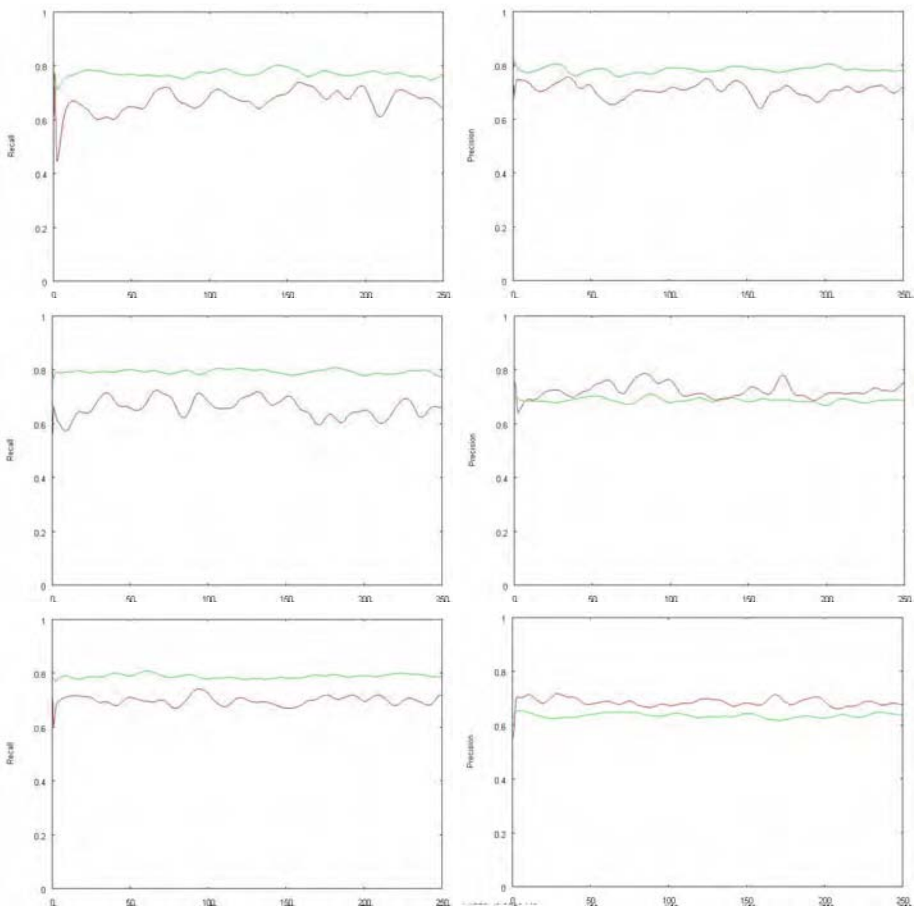


Fig. 3. Recall (left column) and Precision (right column) along time for the 3 testing sequence (From top to the bottom: sequence 1, sequence 2 and sequence 3), red lines for Algorithm 1; green lines for Algorithm 2. Algorithm 2 has higher values of Recall.

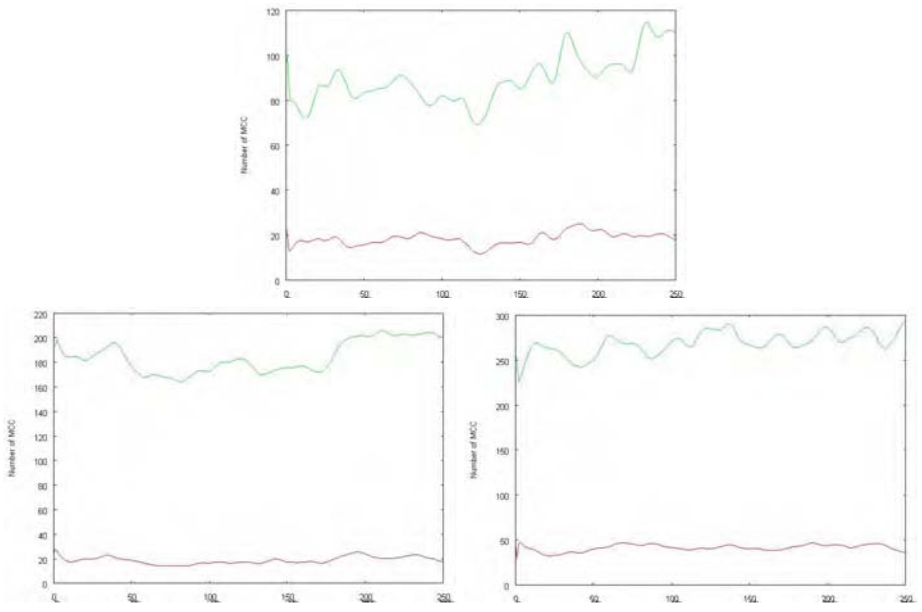


Fig. 4. Number of MCCs along time for the 3 testing sequence, red lines for Algorithm 1; green lines for Algorithm 2 (From top to bottom: sequence 1, sequence 2 and sequence 3). Algorithm 3 detects much more MCCs for all of the three video sequences.

slightly higher. Here another measure should be taken into consideration, which is the number of the MCC detected by each algorithm. According to the plots in Figure 4, in sequence 1 the average number of MCC detected by Algorithm 1 is around 20, while by Algorithm 2 the number is around 100; in sequence 2, the numbers are around 20 and 200, respectively; in sequence 3 the numbers are around 40 and 280, respectively. Algorithm 2 detects much more MCC, especially for sequence 2 and 3. Due to the above fact and the fact Algorithm 2 produces higher Recall, it can be deduced that the slight drawback of the Precision only indicates more noise has been introduced to our assessment.

4 Conclusion

Two novel algorithms to estimate the motion of a crowd in complex scenes are presented, evaluated and compared in this paper. The two algorithms are compared using three surveillance video sequences and quantitative results are generated based on individual local descriptor and MCC (Motion Connected Component). MS and MAE are used as criteria for local descriptor based assessment. The values of MS generated by the two algorithms are all above 0.6 and for Algorithm 2 the values are all above 0.7. For the values of MAE, those generated by Algorithm 2 are always below those generated by Algorithm 1. In the MCC based assessment, for ratio of Recall almost all of the values generated by

Algorithm 1 are above 0.6 while those generated by Algorithm 2 are close to 0.8; and for ratio of Precision, the values generated by both two are above 0.6. We can conclude that the experimental results show the Algorithm 2 works better with most of the experimental sequences while both outcomes are acceptable. The crowd dynamics estimation provides a suitable precursor to processes for learning modes of complex dynamics, describing behaviour and supporting for work in high-level vision and socio-dynamics modelling. Based on the two algorithms presented, our future work will be focused on developing novel method of building mature and reliable crowd dynamics model through computer vision.

Acknowledgement

The research described in this paper is partially supported by the British Telecom Group PLC.

References

1. Cohen, I., Ayache, N., Sulger, P.: Tracking points on deformable objects using curvature information. In: Proceedings of the Second European Conference on Computer Vision (1992)
2. Lowe, D.G.: Object Recognition from Local Scale-Invariant Features(ICCV 1999). In: Seventh International Conference on Computer Vision, vol. 2 (1999)
3. Gouet, V., Boujemaa, N.: About optimal use of color points of interest for content-based image retrieval. Technical Report, RP–4439 (2002)
4. Gabriel, P., Hayet, J., Piater, J., Verly, J.: Object tracking using color interest points. In: Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, pp. 159–164 (2005)
5. Mathes, T., Piater, J.: Robust non-rigid object tracking using point distribution models. In: Proc. of British Machine Vision Conference (BMVC) vol. 2 (2005)
6. Zhan, B., Remagnino, P., Velastin, S., Bremond, F., Thonnat, M.: Matching gradient descriptors with topological constraints to characterise the crowd dynamics. In: VIE 2006. Visual Information Engineering, IET International Conference, pp. 441–446 (2006) ISSN: 0537-9989, ISBN: 978-0-86341-671-2
7. Zhan, B., Remagnino, P., Velastin, S.A., Monekosso, N., Xu, L.Q.: Motion estimation with edge continuity constraint for crowd scene analysis. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Nefian, A., Meenakshisundaram, G., Pascucci, V., Zara, J., Molineros, J., Theisel, H., Malzbender, T. (eds.) ISVC 2006. LNCS, vol. 4292, pp. 861–869. Springer, Heidelberg (2006)
8. Subramanya, S.R., Patel, H., Ersoy, I.: Performance evaluation of block-based motion estimation algorithms and distortion measures. In: ITCC 2004. Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004), Washington, DC, USA, vol. 2, IEEE Computer Society, Los Alamitos (2004)
9. `http://www.hsl.creighton.edu/hsl/Searching/Recall-Precision.html`

A Binary Decision Tree Based Real-Time Emotion Detection System

Adam Livingston, Ming-Jung Seow, and Vijayan K. Asari

Computational Intelligence and Machine Vision Laboratory
Department of Electrical and Computer Engineering
Old Dominion University, Norfolk, VA 23529
{alivings,mseow,vasari}@odu.edu

Abstract. This paper presents a real-time emotion detection system capable of identifying seven affective states: agreeing, concentrating, disagreeing, interested, thinking, unsure, and angry from a near infrared video stream. A Viola Jones face detector is trained to locate the face within the frame. The Active Appearance Model is then used to place 23 landmark points around key areas of the eyes, brows, and mouth. A prioritized binary decision tree then detects, based on the actions of these key points, if on of the seven emotional states occurs as frames pass. The completed system runs accurately and seamlessly on an Intel Pentium IV, 2.8 GHz processor with 512 MB of memory, achieving a real-time frame rate of 36 frames per second.

1 Introduction

Charles Darwin was one of the first scientists to recognize that facial expression is one of the most powerful and immediate means for human beings to communicate their emotions, intentions, and opinions to each other. Computer has been integrated to the daily part of our live. Yet, when it comes to the world of computers, there are situations where the man–machine interaction could be improved by having machines capable of adapting to their users. The term “human computer interaction” suggests a two-way exchange, with each participant aware of the other and responding appropriately. The fact remains that current computers are almost completely unaware of the actual state of the human user.

Human-Computer Interaction (HCI) is a research area aiming at making the interaction with computer systems more effective, easier, safer and more seamless for the users. The goal of this paper is to contribute to the development of a human computer interaction (HCI) environment in which the computer detects and tracks the user's affective states, and initiates communications based on this knowledge, rather than simply responding to user commands.

The remainder of this paper is organized as follows. Section three gives background and related work to the devolvement of HCI interfaces. Next, section three discusses the design and function of the system developed. The results and performance of system are presented in section four. The final section presents a summary and future work.

2 Background

In the scientific community, a shared belief is that the next step in the advancement of computing devices and user interfaces is not to simply make faster applications but also to add more interactivity, responsiveness and transparency. In the last decade much more effort has been directed towards building multi-modal, multi-media, multi-sensor user interfaces that emulate human-human communication with the overall long-term goal to transfer to computer interfaces natural means and expressive models of communication [1].

Cross-disciplinary approaches have begun developing user-oriented interfaces that support non-GUI interaction by synergistically combining several simultaneous input and/or output modalities, thus referred to as multimodal user interfaces. In particular, multimodal Perceptual User Interfaces (PUI) have emerged as potential candidate for being the next interaction paradigm. PUIs are highly interactive perceptual interfaces actively sense and perceive the world making use of a wide range of modalities, either sequentially or in parallel [2].

Facial expressions are the facial changes in response to a person's internal emotional states, intentions, or social communications. Facial expression analysis has been an active research topic for behavioral scientists since the work of Darwin in 1872 [3]-[6]. Suwa et al. [7] presented an early attempt to automatically analyze facial expressions by tracking the motion of 20 identified spots on an image sequence in 1978. After that, much progress has been made to build computer systems to help us understand and use this natural form of human communication [8].

The process of reading the mind in the face is inherently uncertain. People can express the same mental state using different facial expressions. Moreover, the recognition of affective cues is in itself a noisy process. To account for this uncertainty, Kaliouby et al. use a multi-level representation of the video input, combined in a Bayesian inference framework, specifically the dynamic Bayesian networks, for developing their mind reading machine [9]. The statistical classifiers in their inference system are based on the Mind Reading DVD [10], a computer-based guide to emotions, developed by a team of psychologists led by Simon Baron-Cohen at the Autism Research Centre in the University of Cambridge.

Kaliouby's system [11] abstracts raw video input into three levels. Actions are explicitly coded based on the facial feature point return from the FaceTracker. Displays are then recognized by Hidden Markov Models (HMMs). Mental states are assigned probabilities by dynamic Bayesian networks. The six affective states identified due to their relevance to HCI are: agreeing, concentrating, disagreeing, interested, thinking and unsure. This project will focus on identifying same six states with the addition of the basic emotional state of angry, utilizing an efficient binary decision tree.

3 System Design

Facial expression analysis includes both measurement of facial motion and recognition of expression. The general approach to automatic facial expression analysis consists of three steps (Fig. 1): face acquisition, facial feature extraction, and facial expression recognition. Face acquisition is a processing stage to automatically find

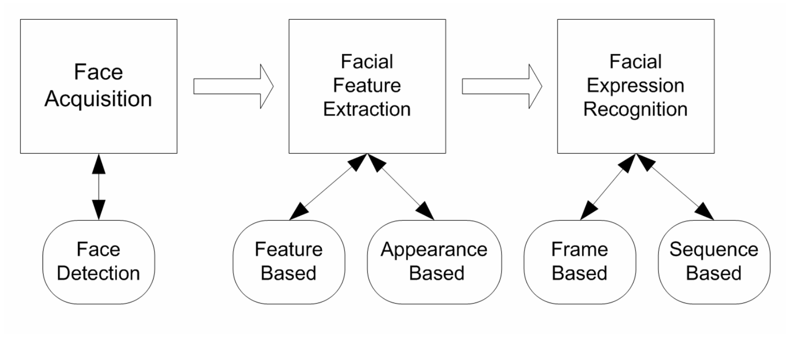


Fig. 1. Conceptual diagram of facial expression recognition system

the face region for the input images or sequences. It can be a detector to detect face for each frame or just detect face in the first frame and then track the face in the remainder of the video sequence.

After the face is located, the next step is to extract and represent the facial changes caused by facial expressions. In facial feature extraction for expression analysis, there are mainly two types of approaches: geometric feature-based methods and appearance-based methods. The geometric facial features present the shape and locations of facial components including but not limited to the mouth, eyes, brows, and nose. The facial components or facial feature points are extracted to form a feature vector that represents the face geometry. With appearance-based methods, image filters, such as Gabor wavelets, are applied to either the whole-face or specific regions in a face image to extract a feature vector. Depending on the different facial feature extraction methods, the effects of in-plane head rotation and different scales of the faces can be eliminated by face normalization before the feature extraction or by feature representation before the step of expression recognition.

Facial expression recognition is the last stage of the automatic facial expression analysis systems. The facial changes can be identified as facial action units or prototypic emotional expressions. Depending on if the temporal information is used, the facial expression can be classified as frame-based or sequence-based.

3.1 Face Acquisition

It is first necessary to localize the face in the frame. Recently, Viola and Jones [12] introduced an impressive face detection system capable of detecting faces in real-time with both high detection rate and very low false positive rates. The Viola-Jones detector consists of three parts. The first is an efficient method of encoding the image data known as an “integral image”. This allows the sum of pixel responses within a given sub-rectangle of an image to be computed quickly and is vital to the speed of the Viola-Jones detector. The second element is the application of a boosting algorithm known as AdaBoost [13] which selects appropriate Haar-like, features that together can form a template to model human face variation. The third component is a cascade of classifiers that speeds up the search by quickly eliminating unlikely face regions.

Viola et al claims a 15 frame per second rate on a 700 MHz Pentium III when processing images with 384×200 dimensions. This speed is obtained by using a cascade of classifiers to progressively filter out non-face sub-frames. Any sub-frame that makes it through this cascade of filters is identified as a face, normalized and fed to the facial feature extraction stage.

3.2 Facial Feature Extraction

The Active Appearance Models (AAM) was first described by Cootes, Edwards and Taylor in 1998 [14]. AAM is a statistical model describing an object's parameters. It combines both shape and texture, resulting in appearance. Shape is to be understood as the outlining contours of the object plus some inner edges corresponding to facial features. Appearance describes the texture of the object in a shape free space. The model becomes 'active' by being able to learn its statistical borders of representation in a one time training session. By learning a model from annotated images, one can prevent blind optimization in run time, which would slow the online process down. Instead, it is possible to optimize similarities in the model offline significantly speeding up later convergence in the underlying high dimensional space of the model.

Matching to an image involves finding model parameters which minimize the difference between the image and a synthesized model example, projected into the image. The potentially large number of parameters makes this a difficult problem. The displacing each model parameter from the correct value induces a particular pattern in the residuals. In a training phase, the AAM learns a linear model of the relationship between parameter displacements and the induced residuals using Principle Component Analysis (PCA). During search it measures the residuals and uses this model to correct the current parameters, leading to a better fit. A good overall match is obtained in a few iterations, even from poor starting estimates. Fig. 2 shows frames from an AAM search for a new face, each starting with the mean model displaced from the true face centre.

To implement the AAM to perform tracking, a training set consisting of 1066 images of three different persons from different angles and with different facial expressions is used. To overcome the problem of color constancy and lighting variations in the environment, near infrared (NIR) imagery is employed. The use of NIR imagery brings a new dimension for applications of invisible lights for computer vision and pattern recognition. This type of imaging hardware device not only provides

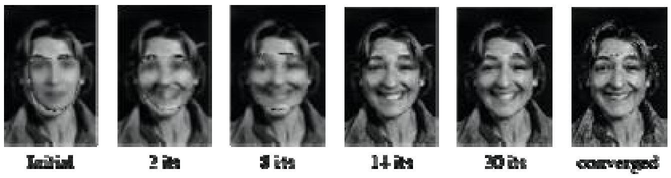


Fig. 2. AAM Example

appropriate active frontal lighting but also minimizes lightings from other sources, making it very suitable for facial feature tracking algorithm such as the AAM.

3.3 Extraction of the Facial Action

A number of studies have shown that the visual properties of facial expressions can be described by the movement of points belonging to facial features [5] [15] [16]. These feature points are typically located on the eyes and eyebrows for the upper face, the lips and nose for the lower face. Fig. 3 illustrates the 2D face model of the 23 feature points used throughout this paper. By tracking these feature points over an image sequence and analyzing their displacements over multiple frames, a characteristic motion pattern for various action units (AUs) can be established.

To understand the AU, one needs to first understand the Facial Action Coding System (FACS), which Ekman and Friesen developed in 1976 as an objective technique for measuring facial movements [17]. The FACS is a human observer based system designed to describe subtle changes in facial features. FACS consists of 44 action units. When people make faces, whether spontaneous expressions or deliberate contortions, they engage muscles around the eyes, mouth, nose and forehead. With FACS, Ekman and Friesen detail which muscles move during which facial expressions.

Table 1 describes how the head and facial AUs that are currently supported are measured. Only those AUs relevant to the facial expressions to be detected are considered. h denotes a user defined threshold. $\vee$ is the logical OR operation while $\wedge$ is the logical AND operation. $[t]$ is the measurement taken at frame t.

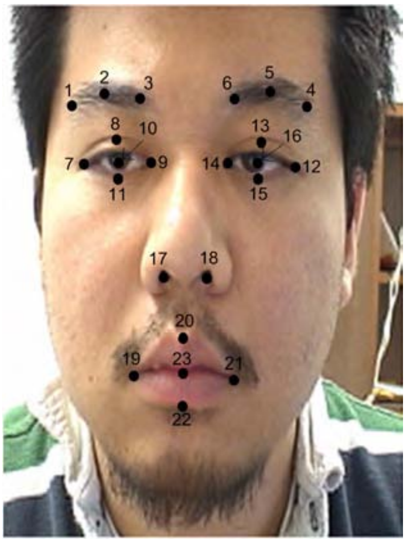


Fig. 3. Face model. 23 feature points track by AAM

Table 1. Measurement of the head and facial AUs with respect to the points in Fig. 3

AU	Action	Measurement
51/52	Head Turn Left/Right	$\frac{\text{Dist}(P_7,P_9)}{\text{Dist}(P_{14},P_{12})} > h_t \vee \frac{\text{Dist}(P_{14},P_{12})}{\text{Dist}(P_7,P_9)} > h_t$
53/54	Head Up/Down	Vertical displacement of $\left[\frac{1}{2}(P_{17}+P_{18})\right][t,t-1]$
55/56	Head Tilt Left/Right	$\text{Slope}(P_9,P_{14}) > h_a \vee \text{Slope}(P_{14},P_9) > h_a$
12	Lip Corner Pull	$\frac{\text{Dist}(P_{19},P_{21})[t]}{\text{Dist}(P_{19},P_{21})[0]} > h_{l_c}$
18	Lip Pucker	$\frac{\text{Dist}(P_{19},P_{21})[t]}{\text{Dist}(P_{19},P_{21})[0]} < h_{l_p}$
27	Mouth Stretch	$\text{Dist}\left\{\left[\frac{1}{2}(P_{19}+P_{21})\right],P_{23}\right\} > h_{m_l} \wedge \left(\frac{\text{Dist}(P_{20},P_{23})}{\text{Dist}(P_{23},P_{22})} > h_{m_r} \vee \frac{\text{Dist}(P_{23},P_{22})}{\text{Dist}(P_{20},P_{23})} > h_{m_r}\right)$
1&2	Eyebrow Raise	$\frac{\text{Dist}(P_1,P_7)[t]}{\text{Dist}(P_1,P_7)[0]} > h_{e_r} \wedge \frac{\text{Dist}(P_3,P_9)[t]}{\text{Dist}(P_3,P_9)[0]} > h_{e_r}$
4	Eyebrow Drop	$\frac{\text{Dist}(P_6,P_{14})[t]}{\text{Dist}(P_6,P_{14})[0]} > h_{e_d} \wedge \frac{\text{Dist}(P_4,P_{12})[t]}{\text{Dist}(P_4,P_{12})[0]} > h_{e_d}$

3.4 Affective Cue Computation

In the computational model of affective states, affective cues (AC) serve as an intermediate step between tracked AUs and inferred affective states. Table 2 lists the nine head and affective cues that are currently supported by this system and their underlying actions.

The head nod and shake AUs are composed of periodic motion that recurs at regular intervals. The temporal structure of a natural head shake or nod is characterized by alternating head-turn-right and head-turn-left or head-up and head-down motion cycles, respectively. An eight stage state machine is constructed for each of these cues to identify these periodic actions over frames regardless of the starting direction.

Table 2. List of affective cues and their component AUs

Affective Cue	Description
Head Nod	Alternating head up (AU53) and head down (AU54) actions
Head Shake	Alternating head turn left (AU51) and head turn right (AU52) actions
Head Tilt	Tilt in one direction (AU55 or AU56)
Head Turn	Pose of turned head (sequence of AU51 or AU52)
Lip Pull	Lip corner pull (AU 12)
Lip Pucker	Lip pucker (AU 18)
Mouth Open	Mouth stretch (AU27)
Eyebrow Raise	Eyebrow raise (AU1 + AU2)
Eyebrow Drop	Eyebrow drop (AU4)

3.5 Affective State Computation

The parameters of the dynamic Bayesian networks developed by Kaliouby [11] were trained using statistical information relating probability distribution and discriminative-power heuristic for each affective cue and affective state combination. This same information was used to determine which affective cues were most likely and most discriminative for each of the affective states to be identified.

A binary decision tree structure composed of nine affective cues is introduced to improve the performance of Kaliouby's system in terms of frames per second. Note that it is not a claim that this approach is better in recognizing user's mental states as compared to the Kaliouby's mind reading system. The system presented is merely an alternative in which the trade off between the computation time and the accuracy of the inference is important. A diagram of the binary decision tree used is presented in Fig 4.

It is imperative to understand that the choice of the order of the binary decision tree seen in Fig. 4 is designed in such a way that it prioritized the affective states we desire. For example, agreement is detected first and disagreement second as these gestures play an important role in our advance human-computer interaction environment.

Table 3. Most likely and most discrimative action cues for each class.

Affective State	Most Likely AC	Most Discriminative AC
Agreement	Lip corner pull	Head Nod
Disagreement	Head Shake	Head Shake
Concentrating	Teeth Present	Head Tilt
Interested	Eyebrow Raise, Mouth Open, Teeth Present	Eyebrow Raise, Mouth Open, Teeth Present
Thinking	Head Tilt, Head Turn	Head Tilt, Head Turn
Unsure	Head Turn	Teeth Absent, Mouth Closed
Angry	Lips Pull, Eyebrow Drop	Lips Pull, Eyebrow Drop

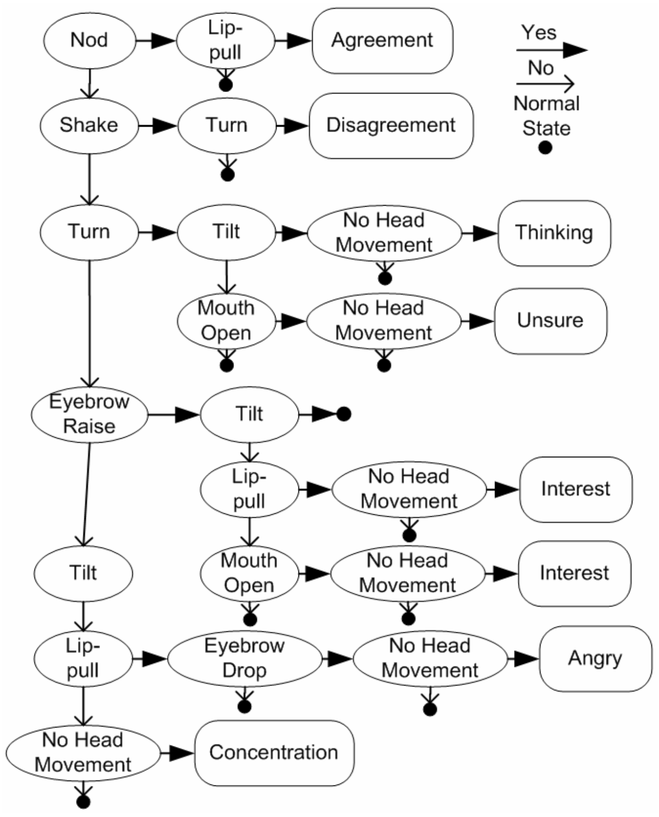


Fig. 4. Binary decision tree structure

4 Experimental Results

Each phase of the system developed was tested incrementally. An implementation of the full Viola-Jones detector was developed to localize the face in the image. The output of the face detector is an image region containing the face. The face detector successfully detects faces across all sizes. This implementation was tested for video processing on a Pentium IV 2.8GHz machine with 512MB of memory. The face detector can process a 360 by 240 pixel image in about 0.019 seconds or 52 frames per second.

In order to create more examples and to enhance the tolerance to low resolution image, motion, and variations in intensity, a series of smoothing operations are applied to the initial set of examples. This step teaches the system how to cope with situations where the AAM is fed with a weak, over smoothed or poorly-contrasted signal. Finally, the training set reached the number of 3198 face images. Each training image was hand annotated with 23 tracking points consistent with Fig. 3.

The tracker performed successful tracking in 38 frames/sec at 320 × 240 resolution with minimum misalignment to the face feature for natural human head motion,

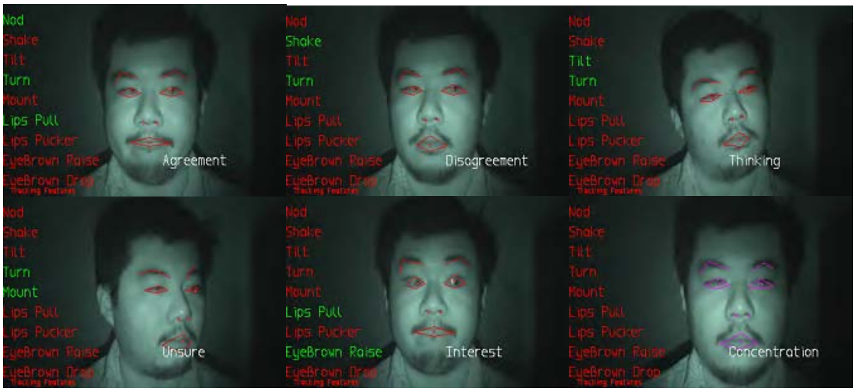


Fig. 5. Affective states computation examples

Table 4. System processing times (ms)

System	Detection/ Tracking	Action Unit	Affective Cue	Affective State	Total
Proposed	3.00	0.09	0.14	41.10	44.33
Kaliouby [11]	2.60	0.08	0.03	0.06	2.77

which is typically ranges between 70 ^{o} - 90^o of downward pitch, 55^o of upward pitch, 70 ^{o} of turn, and 55 ^{o} of tilt. The complete system ran successfully, detecting the affective cues in real-time and accurately interpreting the effective state. Example of six of the seven state detected are presented below.

The binary decision tree based affective states machined was tested on an Intel Pentium IV, 2.8 GHz processor with 512 MB of memory. The processing time at each of the levels of the system is summarized and compared with Kaliouby's system in Table 4. The mind reading machine developed by Kaliouby [11] was tested on an Intel Pentium IV, 3.4 GHz processor with 2 GB of memory. Even utilizing slower machine with less memory the total processing time for the completed system was significantly reduced. The resulting frame rate achieved was 36 frames per second.

5 Conclusion and Future Work

We have presented a system which is capable of recognizing seven affective states: agreeing, concentrating, disagreeing, interested, thinking, unsure, and angry. The ability to detect these seven states shows promise in changing the way an individual communicates with a computer. The real-time system functions in three stages. Viola Jones face detection is used to identify faces within a frame. Active Appearance Model is then used to map 23 landmark points to key locations on the face. Finally, these landmarks are used to form action units which form affective cues as successive frames pass. Depending on which action cues occur, a binary decision tree detects emotional states. The resulting system achieved a real-time frame rate of 36 frames per second.

Work is currently underway to make this system more efficient and accurate. A quad-tree based technique to pre-filter out prospective face sub-frames based on regional variance is currently under development. Also, new approaches for detecting landmark points on the face utilizing position in relation to the overall face are currently under investigation.

References

1. Bernsen, N.O.: Multimodality in language and speech systems. From theory to design support tool. In: Granström, B., House, D., Karlsson, I. (eds.) Multimodality in Language and Speech Systems, pp. 93–148. Kluwer Academic Publ., Dordrecht (2002)
2. Corradini, A., Mehta, M., Bernsen, N.O., Martin, J.C., Abrilian, S.: Multimodal Input Fusion in Human-Computer Interaction on the Example of the on-going NICE Project. In: Proc. NATO-ASI Conf. on Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management, Yerevan (Armenia), August 18th-29th (2003)
3. Darwin, C.: The Expression of Emotions in Man and Animals. In: Murray, J. (ed.), University of Chicago Press, Chicago (1965)
4. Ekman, P.: The Argument and Evidence about Universals in Facial Expressions of Emotion, pp. 143–164. Wiley, New York (1989)
5. Ekman, P., Friesen, W.: The Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, San Francisco (1978)
6. Scherer, K., Ekman, P.: Handbook of Methods in Nonverbal Behavior Research. Cambridge University Press, Cambridge (1982)
7. Suwa, M., Sugie, N., Fujimora, K.: A preliminary note on pattern recognition of human emotional expression. In: Int'l Joint Conf. on Pattern Recognition, pp. 408–410 (1978)
8. Tian, Y., Kanade, T., Cohn, J.F.: Recognizing Action Units for Facial Expression Analysis. IEEE Trans. Pattern Analysis and Machine Intelligence 23(2) (2001)
9. El Kaliouby, R., Robinson, P.: Real-Time Inference of Complex Mental States from Facial Expressions and Head Gestures. In: The IEEE Int'l Workshop on Real Time Computer Vision for Human Computer Interaction at CVPR, IEEE Computer Society Press, Los Alamitos (2004)
10. Baron-Cohen, S., Golan, O., Wheelwright, S., Hill, J.J.: Mind Reading: The Interactive Guide to Emotions. Jessica Kingsley Publishers, London (2004)
11. El Kaliouby, R.: Mind-reading machines: automated inference of complex mental states. Technical Report UCAM-CL-TR-636 (2005)
12. Viola, P.A., Jones, M.J.: Robust Real-Time Face Detection. Int'l J. Computer Vision 57(2), 137–154 (2004)
13. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of online learning and an application to boosting. J. Comp. Sys. Sci. 55(1), 119–139 (1977)
14. Cootes, T.F., Edwards, G., Taylor, C.J.: A comparative evaluation of active appearance model algorithms. In: BMVC 1998. Proc.Ninth British Machine Vision Conf, vol. 2, pp. 680–689 (1998)
15. Bruce, V.: Recognizing Faces. Lawrence Erlbaum, Hove, East Sussex (1986)
16. Pantic, M., Rothkrantz, L.J.M.: Automatic Analysis of Facial Expressions: The State of the Art. IEEE Trans. Pattern Analysis and Machine Intelligence 22, 1424–1445 (2000)
17. Ekman, P., Friesen, W.V.: Measuring facial movement. Environmental Psychology and Nonverbal Behavior 1(1), 56–75 (1976)

Building Petri Nets from Video Event Ontologies

Gal Lavee, Artyom Borzin, Ehud Rivlin, and Michael Rudzsky

Technion - Israel Institute of Technology, Haifa, Israel 32000

Abstract. Video event understanding requires a formalism that can model complex logical temporal and spatial relations between composing sub-events. In this paper we argue that the Petri-Net is such a formalism. We go on to define a methodology for constructing Petri-Net event models from semantic descriptions of events in two well known video event ontology standards, VERL and CASE^E.

1 Introduction

Understanding events in video data is a research area that has received much attention in recent years. Video events are compositional entities made up of smaller sub-events which combine in logical, spatial and temporal relationships. These sub-events can occur in sequence, be partially ordered, or occur altogether asymmetrically. An interesting problem is selecting a formalism that can be used to represent this kind of structure. Given such a formalism events of interest within a scene can be defined in the set up phase of a surveillance system and recognized as they occur.

Other formalisms such as Bayesian networks, Hidden Markov Models and Stochastic Grammars have been suggested as such formalism in previous work. However, these methods do not define semantically meaningful sub-events. Thus it is not straightforward to describe the composition of an event on a semantic level. Furthermore these models are not well equipped to describe the sometimes complex (usually temporal) relationships between the sub-events.

Another problem in the discipline of event understanding is translating user's/ expert's knowledge of events of interest in a particular domain into an event model. For the specification of domain knowledge, several ontology languages have been proposed to allow robust definition of events. It has not been established how these event descriptions can be reconciled with the machine formalisms for describing and recognizing events described above.

The main contribution of this paper is a solution towards bridging these two fields of study. Given a description of an event in an ontology language we offer a way of translating this information into a formalism understood by a machine event recognition system. We have chosen the Petri-Net formalism to this end.

This work extends previous work in modeling video events using Petri-Nets by describing a structured approach to constructing the event model. Previous work has illustrated how Petri-Nets are useful and effective for the modeling of events, however, the principles of construction of the Petri-Nets have generally been ad hoc and expert oriented.

The remainder of this paper is organized as follows: In Section 2 we will briefly define Petri-Nets and discuss their useful properties for event modeling. In Section 3 we will discuss related work and approaches to constructing Petri-Nets for video event understanding. In Section 4 we will discuss our approach for integrating the Petri-Net formalism with standard video event ontology languages. In Section 5 we will show how this formalism can be applied to video input to provide an event summary as output. In Section 6 we will offer conclusions and discussion derived from our experiments.

2 Petri-Nets and Their Properties

A Petri-Net (PN) model is graphically represented by a directed bipartite graph in which the two types of nodes (places and transitions) are drawn as circles, and either bars or boxes, respectively. The arcs of the graph are classified (with respect to transitions) as: input arcs - arrow-headed arcs from places to transitions, output arcs - arrow-headed arcs from transitions to places, inhibitor arcs - circle-headed arcs from places to transitions. Multiple arcs between places and transitions are permitted and annotated with a number specifying their multiplicities. Places can contain tokens that are drawn as black dots within places. The state of a Petri-Net is called marking, and is defined by the number of tokens in each place. The initial Petri-Net state is called the initial marking. For further details interested readers are referred to [1] and [2].

The temporal dimension is of great importance in the domain of video understanding. The possible relationships between temporal intervals, defined by Allen [3], are used to describe the relationships between sub-events within the same event. Petri-Net fragments (sub-networks) modeling each of the seven of Allen's temporal relations are well understood.

Petri-Nets are also a powerful tool for modeling other aspects of video events. It is straightforward to model logical (AND, OR and NOT) relations between sub-events as Petri-Nets fragments. Spatial distance relations can be enforced as transition enabling rules. Petri-Nets may also be used to model concurrency and partial ordering among sub-events. The fragments modeling the relationships between the sub-events can be combined hierarchically to form an event model Petri-Net.

3 Related Work

While it has been noted in the literature that Petri-Nets are a useful tool for modeling the structure of events, it has not been generally agreed upon how to build a particular event model using the Petri-Net formalism. For this reason the design of networks has been rather ad hoc and dependant on decisions of the implementor. However, two classes of approaches have emerged, the Plan Petri-Net and the Object Petri-Net. In this section we will discuss and compare these approaches.

The Object Petri-Net is a term we have defined for the class of event model that appears in [4] and [5]. It is so called because of design choices in constructing the model. In particular, a token in this type of Petri-Net corresponds to a detected object in the scene. Places in Object Petri-Nets represent particular states of the object. Enabling rules in transitions in this model class are conditions on the properties of the object tokens. The events of interest are the transitions themselves and a complex event of interest may lie at the end of a chain of a number of such events. An advantage of this style of network construction is that multiple events of interest can be considered within the same network. Because tokens take on the properties of the objects they represent, this type of network may be considered a type of colored Petri-Net.

Plan Petri-Nets are another class of event model which appear in [6]. In this type of model each place represents a sub-event and the occurrence of this sub-event is represented by a token in this place (only a one token capacity is afforded to each place). The enabling rules of the transitions in this type of network are based on scene states rather than on particular object properties (the tokens in each place represent the existence of scene states rather than particular objects). Plan Petri-Nets model each event of interest as a separate network which has a "sink" transition that indicates whether the event has occurred or not. Each internal transition is the beginning/end of a sub-event.

Seemingly, Object Petri-Net model allow a more robust model due to the fact that they can accommodate multiple tokens for multiple objects and model multiple events within the same event network. However, the complexity necessary to facilitate this is not conducive to a semi-automatic construction of Petri-Nets (Section 4) and hence the simplicity and separation of the Plan Petri-Nets gives them an advantage in this regard.

4 Ontology Languages

The domain of video events is inherently ambiguous and the definition of events of interest is usually left up to the human designers of a particular system. However, these events do have some common aspects that can be isolated. These are usually the compositional structure of the event which includes logical, spatial and temporal relations as well as hierarchical structure and causality. This observation has led to the development of video event description ontologies such as VERL [7] and CASE^E [8]. These ontologies allow a straightforward description of the event which can then be used to construct the event model. In this section we argue that these descriptions can easily be transformed into a Petri-Net model which can then be used to analyze video events.

4.1 Video Event Representation Language

The Video Event Representation Language (VERL) was introduced in [7] as representation of video events based on the Event Representation Language. Nevatia et al. explain the representation scheme and illustrate its use through examples such as a tailgating scenario in a secure facility entryway. The emphasis of this ontology language is to capture relations such as temporal and logical relationships and distinguish between single-thread (linearly ordered sub-events) and multi-thread (simultaneously occurring sub-events). The ontology language assumes that "primitive" events have been defined and are taken care of by a lower level. As we have mentioned in previous sections these aspects are well modeled by the Petri-Net formalism and thus it is reasonable to conclude that an event structure detailed in the VERL representation can be modeled into a Petri-Net. Of course, just as there are many ways to construct a Petri-Net there are equally many ways to transform the ontology representation into a suitable Petri-Net model. We propose one such transform. The advantage of having this transform is that similar events will have similar Petri-Net representations rather than ad hoc representations as we have seen in the past. We describe this technique in more detail with an example in section 4.3.

4.2 CASE^E Ontology

CASE^E is another ontology scheme proposed in [8]. It is based on a natural language scheme proposed by Fillmore in the 1960s [9] which utilizes the basic unit of a case frame. Case frames are structural entities which define the skeleton of a particular sentence. The extension proposed in the CASE^E framework is to allow a hierarchy of case frames to define a video event, allow for multiple agents to be included in the event descriptions, and include support for causal relationships. [8] gives an example of defining events in the railroad crossing domain using CASE^E. An event specified within the framework of the CASE^E ontology may also be converted into a Petri-Net representation. An example of this conversion is given in section 4.3.

4.3 Translation of Ontology Description to Petri-Net Representation

In this section we discuss our approaches for translating the ontology language event description into an equivalent Petri-Net model. It is important to point out that what we propose here is not an automated process, but rather a methodology for constructing Petri-Net event models that seeks to minimize arbitrary design differences between event models representing the same or similar events.

We offer here two different approaches to constructing the event model. The difference between these approaches is the class of Petri-Net event model they produce. One of these approaches corresponds to the Object Petri-Net and the other to the Plan Petri-Net.

We will begin the description of the transform technique by describing a special Petri-Net fragment we call the "sub-event fragment". As the name implies we utilize this fragment to represent sub-events composing our event of interest. This fragment has two versions. If we are not going to apply temporal relations to the fragment then it is sufficient to represent it as a linear sequence of a place, transition and another place. The first place represents the pre-conditions of the

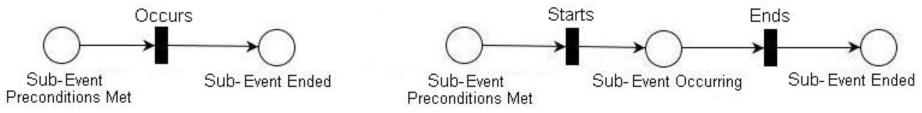


Fig. 1. Sub-Event Fragment

sub-event being met, the transition represents the occurrence of the event, and the final place represents the state of the sub-event having been concluded. We call this fragment the "simple" sub-event fragment. Having such a representation for each sub-event allows us to connect them using logical and spatial relations.

For sub-events to have temporal relations applied to them we extend the fragment into a chain of five nodes. A place node linked to a transition, place, transition and finally a place. The beginning and ending places, as in the smaller fragment, represent the preconditions for a sub-event occurring and the sub-event having concluded, respectively. The additional middle place represents the state of the sub-event currently in progress. The first of the two transitions indicates the start of the sub-event while the second represents the end of the sub-event. This structure allows us to relate fragments using temporal relation constructions (see below). For this reason we have named this type of fragment the "temporal" sub-event fragment. An example of the two possible versions of the sub-event fragment is shown in figure 1.

The next step is to connect the fragment representing the various sub-events in a manner that corresponds to their event description in the ontology language. Simple logical relations are straightforward to attach to the sub-event fragments. Temporal relations must be connected in a way that enforces that the occurrence of the sub-events complies with the temporal ordering defined by the relation. The OVERLAPS relation, for example, requires that sub-event A (the top fragment in figure Figure 2) starts before sub-event B (the bottom fragment) and ends after sub-event B has started but before sub-event B has ended. To enforce this we connect the transition "starts" in fragment A to the "Precondition" place in fragment B (i.e. A having started is a pre-condition for B to start). Similarly, we connect the "Ended" place in fragment A to the "Ends" transition in fragment B (i.e. sub-event B can only end if sub-event A has already ended. Any two sub-events which conform to the temporal relation OVERLAPS will be able to both reach their ending state. We preform a similar construction for each of the remaining of Allen's temporal relations. It may be possible to construct a temporal relation in more than one way. However, choosing a consistent representation for each of the temporal relations will allow us to avoid decisions on these matters when building the event model. Figure 2 illustrates how we have chosen to relate sub-events for each of the temporal relations.

When connecting the sub-event fragments we can fuse nodes that are shared between fragment. For example, if the ending state of a particular sub-event is the precondition of another we can merge the two places into one. As a result of this, some of the places and transitions in the resulting Petri-Net model may

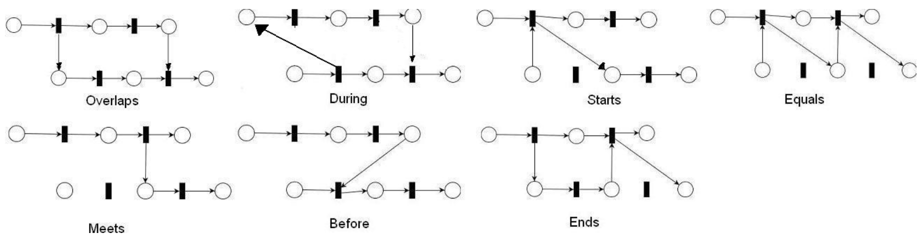


Fig. 2. Petri-Net Fragments Corresponding to Temporal Relations

be redundant. We eliminate such unconnected places and transitions to make a more compact model. The unconnected nodes are shown in the figure to illustrate this idea.

In an Object Petri-Net approach each token represents an object in the system and each place represents a possible state for this type of token. To generalize this intuitive concept into a semi-automatic construction we first define all possible objects and object states of interest to us in a particular event. We must then construct Petri-Nets that model each of these states as a place and the transition between them as a Petri-Net transition. The next step is to add the special fragments representing the structure of the event. The traversal of these structure fragments will be dependant on the objects and object states and thus we make the appropriate connections from the pertinent nodes in the object state transition fragment to their appropriate place in the event structure fragment.

In a Plan Petri-Net each place represents a system state and a token in a place indicates that the system state holds. Transitions can be defined to directly represent atomic sub-events (ontology language primitives). The resulting structure is straightforward to project upon our special event structure fragment. We will illustrate the construction processes of the Plan and Object Petri-Nets using examples from the ontology languages discussed in sections 4.1 and 4.2.

In the first example we consider the scenario of train tracks intersecting a road. Our event of interest is a "safe crossing event", that is when a train approaching has caused the signal to change, the gate to lower and the oncoming car to stop. The train can then proceed to cross the intersection zone safely. The first step in constructing an event model is building the corresponding temporal fragment for each of the sub-events (train approaching, signal change, gate lower and car stop in our example). We then relate these sub-event fragments using the appropriate temporal relations. In our scenario, each of the sub-event are related through the AFTER relation (a variant on Allen's BEFORE) and the CAUSE relation (which we also model using BEFORE) so we connect each of the sub-event fragments in accordance with the BEFORE relation in figure 2. We place a transition node at the conclusion of the fragment string to indicate our safe crossing event has occurred. The resulting construction is illustrated in figure 3. This network is pictured before possible simplification to illustrate the connection of the sub-event fragments to enforce temporal relations.

Once we have this network we can simplify it. Place nodes with with no incoming arcs are removed from the network. The middle transition, place, transition

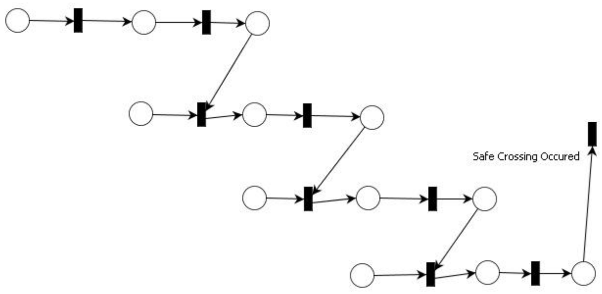


Fig. 3. Safe Crossing Event Plan Petri-Net Before Simplification

sequence of each sub-event fragment can be consolidated into a single transition if it does not affect the event composition structure (i.e. there are no arcs connected to the "event occurring" place, no external outgoing arcs from the "event started" transition and no external incoming arcs to the "event ended" transition.) The resulting simplification of the network in our example is a linear chain pictured in figure 4b (the label SE indicates each of the above mentioned sub-events).

We will now illustrate the construction of an Object Petri-Net representing this event. Our objects of interest in this event are the train, car, signal and gate. This information is given to us by the CASE^E ontology event specification (domain entities). Not provided are the possible states each of these objects can assume. This information may be implied by the domain predicates and domain event specification, however, it can also be added explicitly with a small extension to the ontology event specification. For our purposes we will assume that this object state information is available. The first step in constructing the Object Petri-Net is to construct fragments representing the state transitions of each of our objects of interest. For example a car may take on the states inscene, stopped, inzone2 and stoppedinzone2. Each of these would be represented by a place and the transitions between them would be represented by a transition node with an enabling rule possibly defined on one of the domain predicates (given by the ontology language). Figure 4a shows this construction for the car object. We construct a similar state transition fragment for Train, Gate, and Signal.

The next step is to examine the event construction, construct a Petri-Net fragment for each sub-event and combine them according to the specified relationship. The process of building this fragment is discussed above. Figure 4b shows the simplified Petri-Net fragment representing this structure. Finally we connect the two pieces of our network by connecting the appropriate object in their corresponding place in the event structure chain using arcs. An example of this is attaching the place corresponding to "`train_approaching` " to the beginning of our event structure fragment to indicate that this state must exist to begin the evaluation of this chain. After these connection we can further simplify the Petri-Net by removing any meaningless places and transitions. The resulting Petri-Net is picture in figure 5.

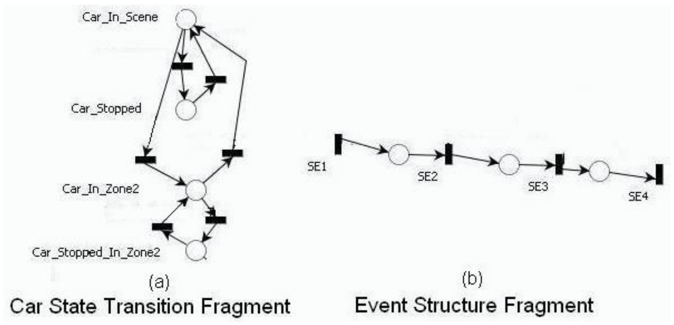


Fig. 4. Object Petri-Net Fragments

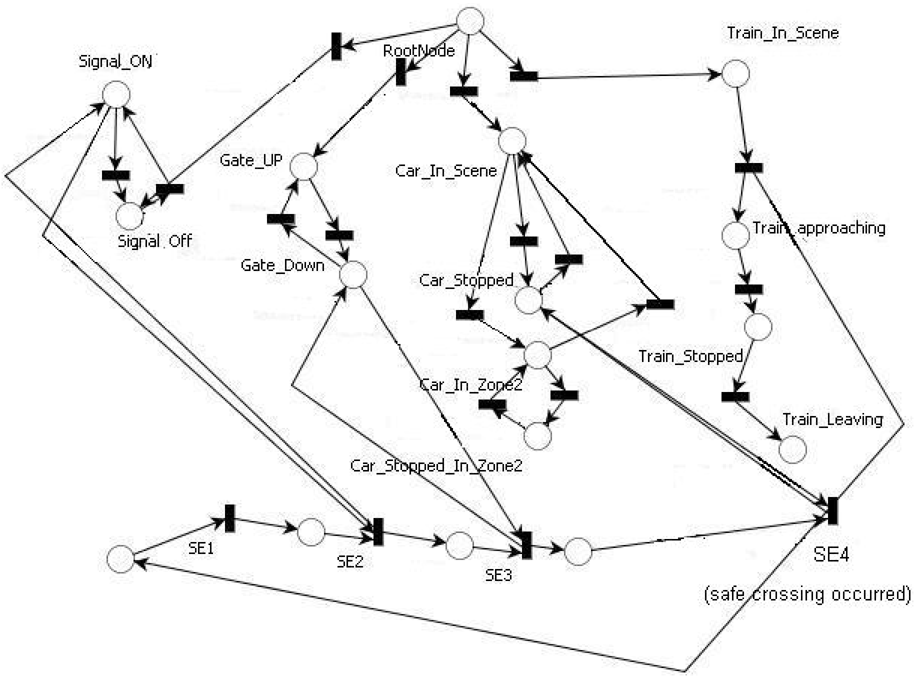
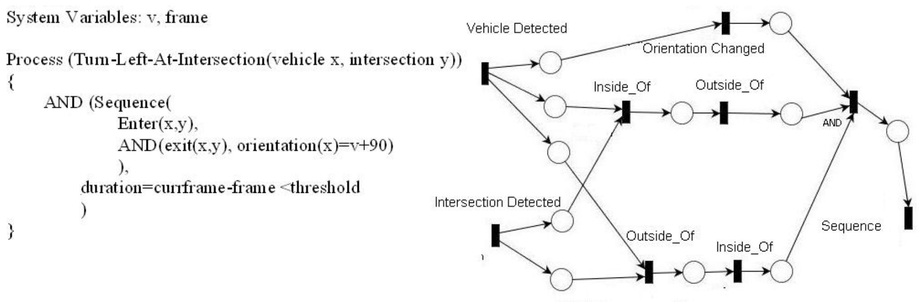


Fig. 5. Safe Crossing Event Object Petri-Net

We will now discuss implementing the same example within the framework of a Plan Petri-Net. This construction results in a more compact network since we model domain predicates as transitions within the event structure fragment. To construct this representation we build the fragment for each of the sub-events and connect them appropriately as we have shown above. In this construction approach we allow a condition on the state of the network to be in the enabling rule of each transition. Thus we can condition the start and end of each of the sub-events on system states and do not need to explicitly link them to object



```
System Variables: v, frame

Process (Turn-Left-At-Intersection(vehicle x, intersection y))
{
    AND (Sequence(
            Enter(x,y),
            AND(exit(x,y), orientation(x)=v+90)
            ),
        duration=currframe-frame <threshold
        )
}
```

Fig. 6. Left Turn Event Plan Petri-Net

states or model all possible object states. This is an advantage over the Object Petri-Net for events that consider many objects and their joint states as it allows maintaining a much simpler model. The resulting Petri-Net is thus just the event structure fragment with added transition enabling rules indicating the start and end condition of each event. The visualization of the network is the same as that pictured in figure 3.

In the next example we consider an event specified in the VERL ontology framework. Along with our fragments for temporal and logical relations we also define fragments for control structures in VERL such as "sequence" and "change". These structures are illustrated in this example. The scenario we will consider is the road intersection scenario. We wish to determine when a left-turn event has taken place. This event is related to one modeled by Higgins in [10], however we have altered the VERL expression representing the event. This event is unique because it considers object position and orientation at different times. We have decided to approach this as a plan network augmented with system variables. Each of the transitions will have an enabling rule defined by a system state as in the previous example as well as a firing rule which may set system variables. Thus, we construct sub-event fragments for each of the sub-events in our example (we add the implicit `vehicle_detected` and `intersection_detected` along with `enter_intersection` , `exit_intersection` and `orientation_changed`). To this end we use the "simple" sub-event fragments because we do not intend to relate the sub-events using temporal relations. We then connect the sub-event fragments using the defined VERL relations (Sequence, AND ,change). Figure 6 shows the VERL notation for the event along with the resulting Petri-Net representation. Note that the "enter" and "exit" processes have been defined using lower-level primitives "`inside_of`" and "`outside_of`". Also noteworthy is that the events `vehicle_detected` and `intersection_detected` have multiple place node outputs because they serve as preconditions for multiple sub events.

5 Conclusion

Petri-Nets are a powerful formalism for describing video events for event monitoring systems. In this paper we have discussed how the Petri-Net formalism can model well the properties inherent to video events such as logical,temporal and spatial composition. We have further discussed the different classes of Petri-Net based event models which appear in the literature. We went on to provide a methodology and examples on how formal ontology language definitions for an event can be transformed into a Petri-Net formalism. The models generated in this paper can be used in an event recognition system based on lower-level vision systems such as object detectors and trackers. The results of such analysis would be a video summary indicating when events of interest occurred and also which semantically meaningful sub-events composed those events. An article describing such results is currently in preparation. The methods described in this paper can in the future be extended to automatically translate ontology specified events to Petri-Net models. Similar methodologies can also be used to translate ontology events into other known formalisms to allow comparison between event models.

References

1. Balbo, G., Conte, G., Donatelli, S., Franceschinis, G., Marsan, M.A.: Modelling with Generalized Stochastic Petri Nets (Hardcover). John Wiley & Sons, Inc., New York, NY, USA (1995)
2. `http://www.informatik.unihamburg.de/TGI/PetriNets/`
3. Allen, J.F., Ferguson, G.: Actions and events in interval temporal logic. Technical Report TR521 (1994)
4. Borzin, A., Rivlin, E., Rudzsky, M.: Surveillance interpretation using generalized stochastic petri nets. In: WIAMIS 2007 (2007)
5. Ghanem, N., DeMenthon, D., Doermann, D., Davis, L.: Representation and recognition of events in surveillance video using petri nets, 112 (2004)
6. Castel, C., Chaudron, L., Tessier, C.: What is going on? a high level interpretation of sequences of images. In: ECCV (1996)
7. Nevatia, R., Hobbs, J., Bolles, B.: An ontology for video event representation. In: CVPRW 2004, Washington, DC, USA, p. 119. IEEE Computer Society Press, Los Alamitos (2004)
8. Hakeem, A., Sheikh, Y., Shah, M.: A hierarchical event representation for the analysis of videos. In: AAAI (2004)
9. Fillmore, C.J.: The case for case. Universals in Linguistic Theory, 1–88 (1968)
10. Higgins, R.P.: Automatic event recognition for enhanced situational awareness in uav video. In: Military Communications Conference (2005)

Feature-Adaptive Motion Energy Analysis for Facial Expression Recognition

Sungkyu Noh, Hanhoon Park, Yoonjong Jin, and Jong-Il Park

Department of Electrical and Computer Engineering, Hanyang University,
17 Haengdang-dong, Seongdong-gu, Seoul, Korea
{snoh,hanuni,lakobe8}@mr.hanyang.ac.kr, jipark@hanyang.ac.kr

Abstract. In this paper, we present a facial expression recognition method using feature-adaptive motion energy analysis. Our method is simplicity-oriented and avoids complicated face model representations or computationally expensive algorithms to estimate facial motions. Instead, the proposed method uses a simplified action-based face model to reduce the computational complexity of the entire facial expression analysis and recognition process. Feature-adaptive motion energy analysis estimates facial motions in a cost-effective manner by assigning more computational complexity on selected discriminative facial features. Facial motion intensity and orientation evaluation are then performed accordingly. Both facial motion intensity and orientation evaluation are based on simple calculations by exploiting a few motion energy values in the difference image, or optimizing the characteristics of feature-adaptive facial feature regions. For facial expression classification, a computationally inexpensive decision tree is used since the information gain heuristics of ID3 decision tree forces the classification to be done with minimal Boolean comparisons. The feasibility of the proposed method is shown through the experimental results as the proposed method recognized every facial expression in the JAFFE database by up to 75% with very low computational complexity.

Keywords: facial expression, feature-adaptive motion energy analysis, decision tree.

1 Introduction

Facial expression is an important element in human communication and studies have shown that facial expression reveals the underlying emotions [4]. However, it has always been a great challenge to build a machine that recognizes human facial expressions effectively and reliably. The main difficulty of automated facial expression recognition is that complex human facial features are represented with 'limited' verbal descriptions. We say 'limited' because the known verbal descriptions may not describe every little detail perceived by the human visual system. The best known facial expression analyzer i.e. human visual system is trained with tremendous amounts of data over a substantial time with the best known parallel learning system (i.e. human neurons and their network), and it may be safe to say that artificially reproducing the complex facial feature representation used by human visual system is near impossible.

Despite the difficulties of automated facial expression recognition, most previous research has heavily relied on complex face models or complicated algorithms. Complicated face models for representing various facial motions have advantages for recognition accuracy but increase computational expense. FACS [3], the most widely used face model, has 46 action units that are generated by facial muscle movements. By combining 46 action units, FACS based face model [5, 2, 15, 19] can represent detailed 2D texture information of facial expressions. 3D modeling of the human face suggests that pose-invariant recognition can be achieved by using multiple instances of 2D texture information. There are mainly two ways to construct a 3D geometric face model from 2D face texture data: one is using a single image [17] and the other is using multiple images [12, 7, 18]. Constructing the 3D model from a single image faces restrictions such that the image has enough 2D texture data (i.e. frontal or limited pose-variants) to construct the 3D model. Constructing the 3D model from multiple images relieves such restrictions; however, multiple views are not always available. Even if multiple views are available, lack of matching features result in inaccurate synthesis of the 3D model. Moreover, considering many action units (FACS) or 3D vertices (3D model) inevitably increases the computational complexity of whole facial expression analysis and recognition process. In environments with limited computational capability, complex model and complicated methods are not suitable.

On the other hand, extremely simplicity-oriented face representation namely point-wise face model [13] has point-wise areas that loosely-fit rectangular regions around six facial features (each eye, each eyebrow, nose, and mouth). Of course, point-wise face model reduces the computational complexity dramatically by simply analyzing the motion intensity of point-wise areas; such a method is more suitable for relieving the burden of computational expenses. However, this method represents very limited, only intensity-based facial motions by neglecting the analysis of motion orientations of facial features. For the motion orientation of facial features, previous efforts used the tracking-based or template-based method such as optical flow [5], HLAC features and fisher weight map for the motion vector [14]. However, they tend to be computationally expensive by computing motion vector on each feature. The computation of the motion vector requires preprocessing, cost functions for motion displacements, etc. that tend to be costly. More reviews on the state of art in the field of facial expression recognition can be found in [11, 6].

Recently, there have been increased demands on intelligent mobile or embedded systems with vision intelligence such as a cell phone recognizing characters, a robot recognizing facial expressions, etc. Accordingly, such intelligent systems have limited computational capacity compared to conventional PCs. This paper addresses the issue of reducing the computational complexity of automated facial expression analysis while recognizing various facial motions; we present feature-adaptive motion energy analysis for recognizing facial expressions.

Unlike previous efforts, our method is simplicity-oriented and yet cost-effective in terms of speed and accuracy. For the cost-effective model, we introduce a new action-based face model: (1) it has reduced ‘action units,’ and (2) it groups facial features that share the same ‘action units’ and similar appearances with symmetry into a *facial unit*.

Feature-adaptive motion energy analysis estimates facial motions in a cost-effective manner by first detecting the *adaptive facial feature regions* (preprocessing). The detected facial feature regions tightly surround the features adaptively in a rectangular shape; therefore, they have more accurate size and location, but smaller region than point-wise areas. Next, *discriminative facial features* are selected in order to minimize the computational expenses for the facial motion estimation. Motion intensity or orientation analysis is then performed adaptively on the detected facial feature regions. Both motion intensity and orientation analysis are based on simple calculations by exploiting few motion energy values within the detected feature region or optimizing the characteristics of the detected feature region. Therefore, a very small computational complexity increase is achieved for the facial motion estimation. After each feature's action is analyzed and known, the corresponding *facial unit*'s action is determined and a *facial motion descriptor* is generated for the classification. Computationally inexpensive ID3 decision tree classifies the facial expression with minimal Boolean comparisons using the *facial motion descriptor*.

The remainder of this paper will be presented in the following order. In Section 2, our action-based face model is introduced. Then, Section 3 presents our overall framework for facial expression analysis and recognition method. Experiments and discussions are covered in Section 4, and Section 5 draws the conclusion of this paper.

2 Action-Based Face Model

Table 1 shows our action-based face model, which is based on FACS [3], and the verbal descriptions of facial expressions from DataFace [4, 20].

Table 1. Action-Based Face Model

Facial Unit	Facial Feature	Action State	
1. Forehead	Forehead	Neutral	Expressed
2. Eyebrow	Left Eyebrow	Neutral	Expressed
			- Down - Up
	Right Eyebrow	Neutral	Expressed
			- Down - Up
3. Glabella	Glabella	Neutral	Expressed
4. Eye	Left Eye	Neutral	Expressed
			- Widened - Narrowed
	Right Eye	Neutral	Expressed
			- Widened - Narrowed
5. Outer Eye Corners	Left Eye Corner	Neutral	Expressed
	Right Eye Corner	Neutral	Expressed
6. Nose	Nose	Neutral	Expressed
7. Cheeks	Left Cheek	Neutral	Expressed
	Right Cheek	Neutral	Expressed

Table 1. (*continued*)

8. Mouth	Mouth	Neutral	Expressed
			- Thinned - Corners Up - Corners Down - Upper Lip Drawn Up - Open Wide
9. Chin	Chin	Neutral	Expressed

A facial unit is defined as a group of a single unique facial feature or two facial features that shares same action state and similar appearances with symmetry. Facial features in the same facial unit shares same action states, because they tend to move together toward same direction when expressed. By defining a facial feature as a subset of a facial unit, our classifier (ID3 tree) considers fewer attributes and thus the reduction in computational complexity of the classification is achieved (details in Section 3). Each facial feature has two default action states, neutral and expressed state; the expressed state is further categorized into sub-action states accordingly depending on the features.

3 Overall Framework

3.1 Preprocessing

Face Detection. Viola et al. [16] proposed the rapid object detection, including the human face, using a boosted cascade of simple features. Lienhart et al. [8] improved the performance by extending the rapid object detection framework in [16]. We used the method of face detection in an arbitrary scene as proposed in [8].

Adaptive Facial Feature Region Detection. We have adopted the facial feature detection framework proposed in [13] and modified it to detect more facial features and accurate facial features' locations and sizes. Our facial feature detection framework is shown in Fig.1 and proceeds as follows.

(1) *Person-independent*: Detected face region is rescaled into a 100x100 image to minimize individual differences.
(2) *Search region restriction*: A valley image [1] of the rescaled image and a generic feature template [9] roughly estimate the locations of main features, i.e. eyes, nose, and mouth. The valley image (See Fig.1c) clearly show main facial features and the generic feature template (See Fig.1d) segments the rescaled face image into 4 sub-regions for main features: R1 for left eye, R2 for right eye, R3 for nose, and R4 for mouth. Therefore, the generic feature template restricts the search regions of the scene where main facial features are most likely to occur.
(3) *Feature-adaptive search space regions*: To restrict even more search spaces for detecting main facial features, we assigned different sizes of rectangular search space regions (See Fig.1e) on top of the generic feature template. For example,

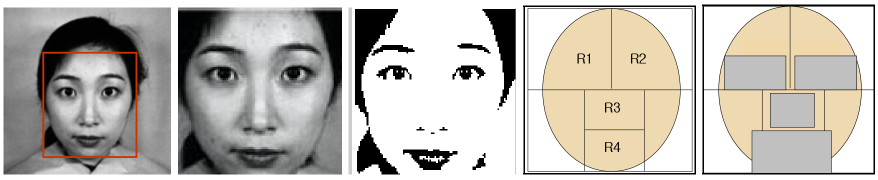


Fig. 1. Facial Feature Detection. (a) Face detection. (b) Rescaled face image. (c) Valley image. (d) Generic feature temple. (e) Rectangular search space regions for main features.

35x20 for each eye, 25x20 for nose, and 45x25 for mouth. By scanning through the specified search space regions and finding the start and end point locations, main facial feature regions are detected in fast and effective manner. Remaining facial features are detected in a same manner.

While scanning through the rectangular search space, size and location of the search space region adjusts to that of actual facial feature (See Fig.2). For the facial features whose actual sizes cannot be known from the valley image, the rectangular search space region becomes the detected feature region. Unlike the method in [13] where 20x20 rectangular regions are assigned on every detected feature, different sizes of rectangular regions that tightly surround facial features are assigned on the detected features. By doing so, noises from the neighboring regions can be reduced.

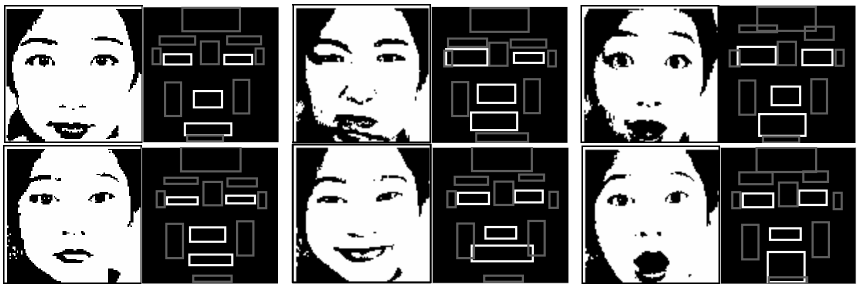


Fig. 2. Detected facial feature regions (main features: white, remaining features: gray)

3.2 Feature-Adaptive Motion Energy Analysis

Facial Feature Selection. Each facial feature's action states, as shown in Table 1, indicate the motion variability of the feature. Depending on the motion variability of the feature, facial features are divided into two categories: non-discriminative (i.e. 2 basic action states) and discriminative facial features (i.e. more than 2 basic action states). Non-discriminative facial features are selected for the feature-adaptive motion intensity evaluation while discriminative features are selected for the feature-adaptive motion orientation evaluation.

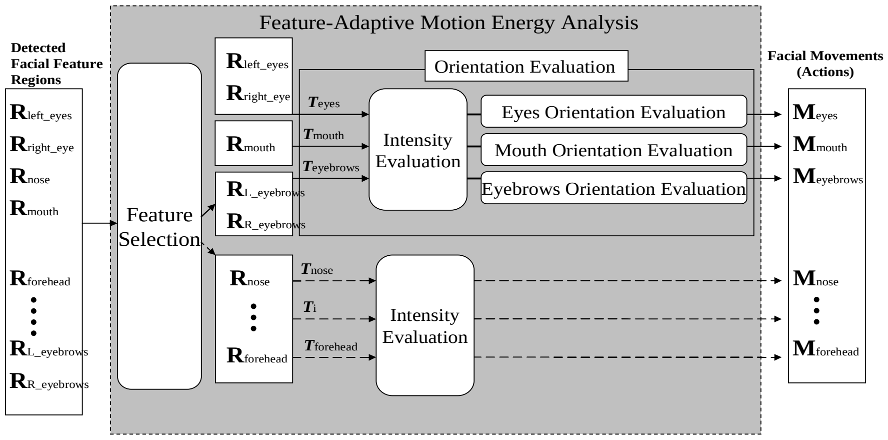


Fig. 3. Feature-adaptive motion energy analysis

Feature-Adaptive Motion Intensity Evaluation. For the non-discriminative facial features, we adopted the framework of point-wise motion energy analysis. Point-wise motion energy analysis uses the difference image and adaptive threshold values to measure the motion intensity of the specified region. The motion intensity is acquired in a fast manner by simply exploiting the few motion energy values within the specified region. More details on point-wise motion energy analysis are found in [13]. The difference is that our method uses the feature-adaptive regions that tightly surround facial features where the method in [13] uses 20x20 regions for every facial feature.

Feature-Adaptive Motion Orientation Evaluation. Feature-adaptive motion orientation evaluations for the discriminative facial features are discussed in detail here. Our approach extended the framework in [13] in three ways. First, we consider not only the motion intensity of features but also the motion orientation of features. Second, the motion orientation evaluation is performed only on selected discriminative features to reduce the computational complexity of the facial motion estimation. Third, the motion orientation of selected discriminative features optimizes the position and size of the detected feature-adaptive regions as well as the motion intensities of sub-regions in the detected feature-adaptive region. There are five discriminative features but three feature-adaptive motion orientation evaluation methods as shown in Fig. 3, because facial features within the same facial unit shares same action states (See Table 1).

Motion orientation evaluation optimizes the motion intensities of sub-regions of the detected mouth region and proceeds as follows. First of all, the location of the mouth is detected using the detected mouth region in the expressed image, R^e_{mouth}. Second, divide R^e_{mouth} into sub-regions $S_1 \dots S_n$ where n = 20, and then find each region's average intensity ($\bar{I}_1 \dots \bar{I}_n$). Third, find average mouth intensities of $\bar{I}_{mouth}$ by adding $\bar{I}_1$ through $\bar{I}_n$ and divide them by n where n = 20. Fourth, find two

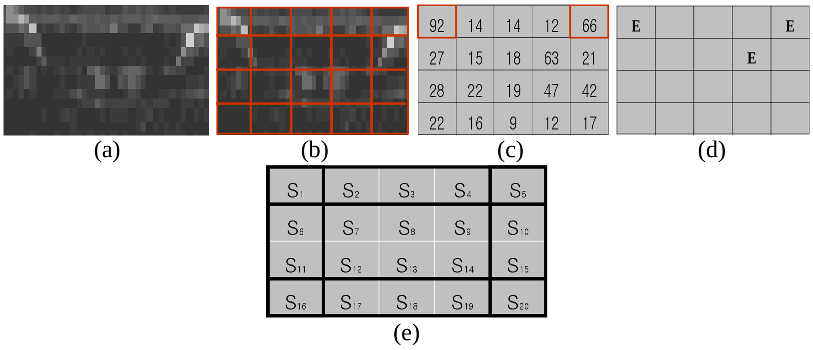


Fig. 4. Motion orientation evaluation of the expressed mouth: 'LipCornersUp' (a) Difference image of the mouth region. (b) The mouth region is divided into 20 sub-regions. (c) Average motion intensities of the sub-regions. S1 and S5 have highest motion intensities. (d) Evaluation of the sub-regions. (e) Sub-regions are organized into mouth regions. Upper mouth regions: S1 through S5. Middle mouth regions: S6 through S15. Lower mouth regions: S16 through S20. Left mouth corners: S1, S6, S11, S16. Right mouth corners: S5, S10, S15, S20.

sub-regions with highest intensities using $\overline{I}_1 \ldots \overline{I}_n$ and $\overline{I}_{mouth}$. Fifth, evaluate each sub-region and find expressed sub-regions E_i. The sub region S_i is expressed if and only if $\overline{I}_i > (\overline{I}_{mouth} .+ T_{mouth})$, where T_{mouth} is the feature-adaptive threshold for the mouth region (We let $T_{mouth} = 10$). Lastly, evaluate the mouth's motion orientation using expressed sub-regions E_i and highest two sub regions from the fourth step.

Note that sub- regions of the detected mouth region are further grouped into logical mouth regions (See Fig. 4e) for the mouth orientation evaluation, which is based on simple Boolean rules. The rule uses the properties that expressed sub-regions represent the shape of the expressed mouth and that the 2 high motion intensity regions indicate the motion orientation of the expressed mouth. Let STRONG(S_i) be strong motion intensity is detected in S_i, MID(S_i) be some motion intensity is detected in S_i, and LOW(S_i) be low motion intensity is detected in S_i. Then,

- *Lip Corners Up* if STRONG(S_1 and S_5).
- *Lip Corners Down* if STRONG(S_{16} and S_{20}).
- *Upper Lip Drawn Up* if and only if STRONG(S_2 through S_4) and MID(S_1 and S_5) and LOW(S_{16} through S_{20}).
- *Open Wide* if STRONG(S_3) and STRONG(S_{18}) and MID(S_2 and S_4) and MID(S_{17} and S_{19}) and LOW(S_7 through S_9) and LOW(S_{12} through S_{14}).
- *Thinned* if MID(S_7 through S_9) and MID(S_{12} through S_{14}) or LOW(S_1 through S_{20}).

The motion orientation evaluation of an eye is as follows. Let R^n_{eye} and R^e_{eye} be the eye region in neutral and expressed image respectively. We define the narrowness of an eye n^k_{eye} as a ratio of the detected eye region's width w^k_{eye} over h^k_{eye} where k is either n for the neutral image and e for the expressed image. Then, an eye is widened if n^e_{eye} is smaller than n^n_{eye}, narrowed if n^e_{eye} is greater than n^n_{eye}, otherwise it is neutral.

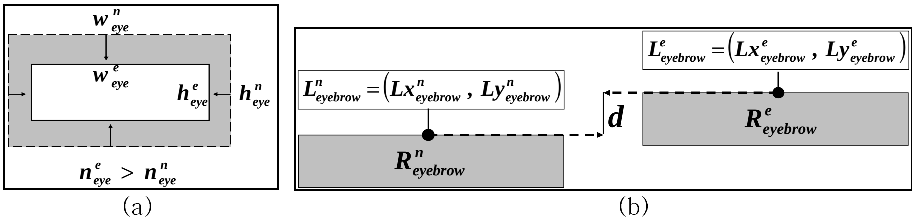


Fig. 5. (a)Eye motion orientation evaluation using the narrowness of a detected eye region. (b)Eyebrows motion orientation evaluation using the start point, a reference point to estimate the location of an eyebrow. The reference point is located at the center of upper end line of the detected eyebrow region and it is used to estimate the location of the eyebrow.

For the motion orientation estimation of eyebrows, let $L^n_{eyebrow} = (Lx^n_{eyebrow}, Ly^n_{eyebrow})$ and $L^e_{eyebrow} = (Lx^e_{eyebrow}, Ly^e_{eyebrow})$ be the reference location (See Fig.5b) of the detected eyebrow rectangular region in neutral and expressed image respectively. Then, an eyebrow is moved down if $Ly^n_{eyebrow}$ is greater than $Ly^e_{eyebrow}$, moved up if $Ly^n_{eyebrow}$ is smaller than $Ly^e_{eyebrow}$, otherwise it is neutral.

Facial Motion Descriptor Generation. A facial motion descriptor is a vector of size 10 containing facial units' actions to represent the facial motion of the given facial expression. For facial units with a single facial feature, the feature's action state becomes the facial unit's action. For facial units with two facial features, a simple rule for generating the facial unit's action is required when corresponding facial features' actions mismatch. The rule has the following assumptions:

1. Movements of neighboring facial features affect each other. Therefore, neighboring facial features' action states are checked when the mismatch occurs. For example, forehead is expressed due to the raise of eyebrows; glabella and eyes are expressed and narrowed respectively due to the pulling down of eyebrows; etc.
2. When neighboring facial features do not provide enough clues, which happen rarely, stereotypical facial feature actions are used to generate the facial unit's action. For example, when action states of left and right eye do not match, it often is the case that the person is uncomfortable i.e. angry or disgust; therefore, facial unit action of eyes becomes narrowed, which is stereotypical eye action for angry or disgust.

3.3 Facial Expression Classification Using ID3 Decision Tree

Ordinary rule-based classifier needs 'trainer' or 'expert' to decide the order of comparisons (or rule); however, it is difficult to come up with an efficient rule when the rule is complicated. We used a modified rule-based classifier with ID3 decision tree as our 'expert' for deciding the order of comparisons. We trained the ID3 decision tree by carefully meeting the requirements for the sample data used by the tree.

There are four requirements that the sample training data used to generate ID3 decision tree must follow [21]. First, the same attributes must describe each example and have a fixed number of values. Second, an example's attributes must already be defined, i.e. they are not learned by ID3 decision tree. Third, discrete classes should be used and classes broken up into vague categories are not sufficient. Lastly, there must be an adequate number of training examples to distinguish valid patterns from chance occurrences.

Our training data set for the ID3 decision tree is generated by using a predefined facial expression recognition rule and 3456 possible action combinations of facial units. We have met all the requirements that the sample training data for the ID3 decision tree must follow because the tree is trained with total of 3456 possible samples (satisfy 4th condition) that are generated from the combination of nine predefined discrete facial units (satisfy 2^{nd} and 3rd condition) and their fixed number of actions (satisfy 1st condition).

The training of the ID3 tree is a one time process for generating an efficient rule from the predefined rule. Since the decision tree is built by using the information gain heuristics of attributes in the training data sample, the attribute that best classifies the remaining samples are chosen when the tree splits. Thus, the classification is done with minimal Boolean comparisons by comparing the value of the root node to the corresponding facial unit action in the facial motion descriptor, and then going downward until it reaches the leaf node and then comparing the leaf node's value with the corresponding facial expression in the facial motion descriptor.

4 Experimental Results and Discussions

We validate the performance of the proposed method in terms of recognition accuracy and computational complexity.

For the recognition accuracy, the proposed method was tested on JAFFE [22] for quantitative analysis on six facial expressions. JAFFE has 213 images and they consist of 40 neutral and 183 expressed images from 10 individual models. For each individual model, one neutral image and all the expressed images were selected for the test; a total of 183 expressed images were tested.

The test was performed as follows. For an each expressed image in JAFFE, we first performed the preprocessing on both the expressed image and its reference neutral image to detect the face and facial features. Then, feature-adaptive motion energy analysis is performed to estimate facial motions of the given expressed image and a corresponding facial motion descriptor is generated. Finally, trained ID3 decision tree uses the generated facial motion descriptor to classify the given facial expression.

Fig. 6 shows the proposed method's performance in terms of recognition accuracy on JAFFE database. Overall recognition accuracy for JAFFE database was 75.4% with the highest recognition rates on 'Surprise' expression (83.8 %) and the lowest recognition rates on 'Fear' expression (63.3 %). We also compared the performance of our method with other simplicity-oriented method that was tested on JAFFE database, namely point-wise motion energy analysis. As shown in Table 2, our method outperforms the method in [13]. This is due to the following factors: (1) More accurate (position and size) adaptive facial feature regions are detected and it avoids

Expression	**Anger**	**Disgust**	**Fear**	**Happy**	**Sad**	**Surprise**
Anger	**20**	1	0	3	2	0
Disgust	1	**14**	0	1	1	2
Fear	2	3	**19**	0	4	2
Happy	1	2	2	**36**	3	0
Sad	1	2	3	3	**18**	0
Surprise	1	0	2	1	2	**31**

Fig. 6. Confusion Matrix

misanalysis of the feature's motion orientation. (2) A greater variety of facial motions are recognized from the feature's motion orientation evaluation.

4.1 Computational Complexity

The computational complexity of feature-adaptive motion energy analysis is discussed here. Our approach does not require calculations of motion vectors. Instead, the facial motion is estimated by simply exploiting the motion energy values within the detected feature regions or the characteristics (position and size) of the detected feature region. Therefore, the computational complexity is measured mainly using the size of detected feature regions.

For a face image of size N (i.e. 10000 for a 100x100 face image), facial features, their surrounding rectangular regions, and their areas (i.e. size or number of pixels in the region) to be $F_1...F_n$, $R_1...R_n$, and $A_1...A_n$ respectively. Then, the time complexity of motion intensity evaluations for *non-discriminative facial feature regions* (=9) is

$$\sum_{f=1}^{9} O(A_f + c) \tag{1}$$

where c is a constant time to calculate an average intensity of the facial feature region and apply threshold to see if the region is expressed. After removing the constant time complexity, equation (1) can be rewritten as

$$\sum_{f=1}^{9} O(A_f) \tag{2}$$

For the mouth orientation evaluation, the original mouth detected region is divided into 20 sub-regions with each sub-region a_s where s = 1…20 and it has a size of A_{mouth} divided by 20. Then, the time complexity to evaluate the mouth orientation is

$$\sum_{s=1}^{20} O(a_s) \tag{3}$$

and it can be rewritten in its general form of $O(A_f)$ where f = mouth. Note that the time complexity of the mouth orientation evaluation does not increase from that of the motion intensity evaluation on the mouth region A_{mouth} when we disregard the constant term. Furthermore, it takes a constant time for the motion orientation evaluation on an eye or an eyebrow as well as the generation of a facial motion descriptor. Then, the time complexity of feature-adaptive motion energy analysis, after disregarding constant terms, is

$$\sum_{f=1}^{10} O(A_f) \tag{4}$$

In our experimental environment (Intel Pentium 4 3.0 Ghz, implemented using JAVA without code optimization), it takes an average of 96 milliseconds for each facial expression to perform the preprocessing, feature-adaptive motion analysis (when the average sum of A_f was about 2500 pixels for a 100x100 facial image), and facial expression classification (using the pre-trained ID3 decision tree). Note that more than 90% of the time is spent on the preprocessing. Since detected facial feature regions tightly surround facial features, a small increase in computational complexity is achieved for the motion orientation evaluation as shown Table 2.

Table 2. Comparison of simplicity-oriented methods. ***t*** = time it takes to read a pixel

Methods	**Scope**	**Tested Database**	**Overall Accuracy**	**Time Complexity (facial expression analysis on a 100x100 face image)**
[13]	-Point-wise face model -Motion intensity analysis -Rule-based classifier	JAFFE	70% (183 images)	$\sum_{f=1}^{6} O(A_f) = 2400t$
Proposed	-Action-based face model -Motion intensity and orientation analysis -Modified rule-based classifier using ID3 decision tree	JAFFE	75% (183 images)	$\sum_{f=1}^{10} O(A_f) \approx 2500t$

5 Conclusions

We propose a cost-effective method that is capable of recognizing various facial expressions with very low computational expenses. The proposed method introduces a simplified action-based model to reduce the computational expenses of facial expression analysis and recognition procedure. Feature-adaptive facial feature regions have information about facial feature's size, position, and motion intensities within the region; therefore, feature-adaptive motion energy analysis optimizes the feature region's characteristics to effectively estimate the facial motion of the given expression.

Unlike previous approaches, our method does not neglect facial feature's motion orientation for simplicity, nor does it require a complex face model and complicated algorithms for more reliable recognition. Experimental results show that the computational complexity of the proposed method is as low as the extremely simple facial expression recognition framework that neglects the motion orientation of facial features, and yet the proposed method recognizes a greater rage of facial expressions (up to 75%). For future research, we plan to further investigate more efficient detection and manipulations of adaptive facial feature region, and extend our work in real-scenarios where facial poses and illuminations cannot be controlled.

References

1. Chow, G., Li, X.: Toward a System for Automatic Facial Feature Detection. Proc. of Pattern Recognition 26, 1739–1755 (1993)
2. Donato, G., Bartlett, M., Hager, J., Ekman, P., Sejnowski, T.: Classifying facial actions. IEEE Transactions on PAMI 21, 974–989 (1999)
3. Ekman, P., Friesen, W.V.: Facial Action Coding System. Consulting Psychologists Press Inc. (1978)
4. Ekman, P.: Facial Expression and Emotion. American Psychologists Col. 48(4), 384–392 (1993)
5. Essa, I., Pentland, A.: Coding, Analysis, Interpretation, Recognition of Facial Expressions. IEEE Transactions on PAMI 19, 757–763 (1999)
6. Fasel, B., Luttin, J.: Automatic facial expression analysis: Survey. Pattern Recognition 36, 269–275 (2003)
7. Gokturk, S., Bouguet, J., Tomasi, C., Girod, B.: Model-based face tracking for view-independent facial expression recognition. In: Proc. of IEEE Int. Conf. on FGR, IEEE Computer Society Press, Los Alamitos (2002)
8. Lienhart, R., Maydt, J.: An Extended Set of Harr-Like Features for Rapid Object Detection. Proc. of IEEE ICIP 1, 900–903 (2002)
9. Lin, C., Wu, L.: Automatic Facial Feature Extraction by Genetic Algorithms. Image Processing 8, 834–845 (1999)
10. Mitchell, T.: Decision Tree Learning. In: Machine Learning, pp. 52–81, ch. 3, McGraw-Hill, New York (1997)
11. Pantic, M., Rothkrantz, L.J.M.: Automatic Analysis of Facial Expressions: the State of the Art. IEEE Transactions on PAMI 22, 1424–1445 (2000)
12. Pantic, M., Rothkrantz, L.: Facial action recognition for facial expression analysis from static face images. IEEE Transactions on SMC-Part B: Cybernetics 34, 1449–1461 (2004)
13. Park, H., Park, J.: Analysis and Recognition of Facial Expression Based on Point-Wise Motion Energy. In: Proc. of ICIAR, pp. 700–708 (2004)
14. Shinohara, Y., Otsu, N.: Facial Expression Recognition Using Fisher Weight Maps. In: Proc. of IEEE Int. Conf. on FGR, IEEE Computer Society Press, Los Alamitos (2004)
15. Tian, Y., Kanade, T., Cohn, J.: Recognizing action units for facial expression analysis. IEEE Transactions on PAMI 23, 1–9 (2001)
16. Viola, P., Jones, M.: Rapid Object Detection Using a Boosted Cascade of Simple Features. Proc. of IEEE CVPR 1, 511–518 (2001)
17. Yepeng, G.: Automatic 3D Face Reconstruction based on Single 2D Image. In: Proc. of Multimedia and Ubiquitous Engineering (2007)
18. Zalewski, L., Gong, S.: Synthesis and recognition of facial expressions virtual 3d views. In: Proc. of IEEE Int. Conf. on FGR, IEEE Computer Society Press, Los Alamitos (2004)
19. Zhang, Y., Ji, Q.: Active dynamic information fusion for facial expression understanding from image sequences. IEEE Transactions on PAMI 27, 699–714 (2005)
20. DataFace, Available at: http://face-and-emotion.com/dataface/emotion/expression.jsp
21. ID3 algorith, Available at: http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm
22. JAFFE database, Available at: http://www.kasrl.org/jaffe.html

Boosting with Temporal Consistent Learners: An Application to Human Activity Recognition[⋆]

Pedro Canotilho Ribeiro, Plinio Moreno, and José Santos-Victor

Instituto Superior Técnico & Instituto de Sistemas e Robótica
1049-001 Lisboa - Portugal
{pribeiro,plinio,jasv}@isr.ist.utl.pt

Abstract. We present a novel boosting algorithm where temporal consistency is addressed in a short-term way. Although temporal correlation of observed data may be an important cue for classification (e.g. of human activities) it is seldom used in boosting techniques. The recently proposed Temporal AdaBoost addresses the same problem but in a heuristic manner, first optimizing the weak learners without temporal integration. The classifier responses for past frames are then averaged together, as long as the total classification error decreases.

We extend the GentleBoost algorithm by modeling time in an explicit form, as a new parameter during the weak learner training and in each optimization round. The time consistency model induces a fuzzy decision function, dependent on the temporal support of a feature or data point, with added robustness to noise. Our temporal boost algorithm is further extended to cope with multi class problems, following the JointBoost approach introduced by Torralba *et. al.* We can thus (i) learn the parameters for all classes at once, and (ii) share features among classes and groups of classes, both in a temporal and fully consistent manner.

Finally, the superiority of our proposed framework is demonstrated comparing it to state of the art, temporal and non-temporal boosting algorithms. Tests are performed both on synthetic and 2 real challenging datasets used to recognize a total of 12 different human activities.

1 Introduction

Although short-term temporal information may convey critical information for several classification problems, it is ignored by many classifiers. Nowadays, data for video applications are acquired at high frame rates in such a way that information changes smoothly in most of the cases. Human motion follows this behavior, thus generate similar motion data during several consecutive frames. An adequate model of this type of data consistency inside the classifier will improve the performance of any non-sequential classifier, that can be used, for instance, to recognize human activities in several different applications, e.g. surveillance, intelligent environments, human/robot interaction or interface.

[⋆] This work has been partially supported by Fundação para a Ciência e Tecnologia (Proj. 61911 - GESTINTERACT; plurianual funds through the POS_Conhecimento Program that includes FEDER funds) and EU Project URUS (IST-045062). Pedro Ribeiro also acknowledges the PhD grant SFRH/BD/18936/2004.

When the temporal evolution of the features is essential like in human activity recognition, there are usually two types of classifiers used: (i) non-sequential, and (ii) sequential. Non-sequential classifiers aim to maximize the number of individual labels predicted correctly, encompassing the temporal dynamics in a short-term manner, usually in the feature computation step. On the other hand, sequential classifiers predict jointly the entire sequence of labels with the highest probability. An example of a non-sequential classifier is the Adaboost cascade for pedestrian detection using pairs of images to compute motion features [1]. In the case of sequential classifiers, a recent work [2] proposes a Conditional Random Field classifier trained with gradient tree boosting. Another approach is the correlation of data volumes to match activities in video, such as the spatio-temporal descriptor based on optical flow, proposed by Efros et.al. [3].

Opposed to most of the works, including the ones referred above, that use a temporal window fixed by hand, we derive a non-sequential classifier that learns the optimal temporal window. We rely on the GentleBoost algorithm [4], that can be defined as a forward stagewise approximate optimization of the exponential loss. We propose to include explicitly short-time consistency in GentleBoost, considering the non-sequential weak classifiers. The recently proposed TemporalBoost [5] also introduced temporal consistency in a boosting procedure, by averaging previous AdaBoost weak classifiers sequentially, while the classification error decreases. However, temporal support is considered in a heuristic procedure only after training the weak classifiers, and the TemporalBoost averaged output is mapped to a binary value. This allows to use the standard AdaBoost procedure [6], but discards the advantages of a fuzzy output.

In this paper we propose to model time directly during the weak classifiers training. As the basis of our work, we consider the gentleBoost algorithm using regression stumps as weak learners. A regression stump is similar to a single node binary tree, that selects a branch for a given feature according to a threshold using a binary decision function. Alternatively, we propose to compute a new decision function, by averaging the decision value in the learned temporal window. This procedure transforms the binary decision into a fuzzy one, according to the temporal window size. This new weak classifier based on a fuzzy decision function provides two main advantageous properties to boosting algorithms: (i) added noise robustness, and (ii) better performance.

In order to extend the binary classification framework to multiclass problems, we adapt JointBoosting [7] algorithm to fit our framework. JointBoost proposes to share weak classifiers among classes by selecting the group of classes which are going to share a feature, allowing to: (i) learn the strong classifiers of each class jointly, and (ii) reduce the number of weak learners needed to attain good performance.

2 GentleBoost with Temporal Consistent Learners

The Boosting algorithm provides a framework to sequentially fit additive models in order to build a final strong classifier, $H(x_i)$. This is done minimizing, at each round, the weighted squared error, $J = \sum_{i=1}^{N} w_i(y_i - h_m(x_i))^2$, where $w_i = e^{-y_i h_m(x_i)}$ are the weights and N the number of training samples. At each round, the optimal weak classifier is then added to the strong classifier and the data weights adapted, increasing the weight of the misclassified samples and decreasing correctly classified ones [7].

In the case of GentleBoost it is common to use simple functions such as regression stumps. They have the form $h_m(x_i) = a\delta\left[x_i^f > \theta\right] + b\delta\left[x_i^f \leq \theta\right]$, where f is the number of the feature and δ is an indicator function (i.e. δ[*condition*] is one if *condition* is *true* and zero otherwise). Regression stumps can be viewed as decision trees with only one node, where the indicator function sharply chooses branch a or b depending on threshold θ and feature x_i^f. To optimize the stump one must find the set of parameters $\{a, b, f, \theta\}$ that minimizes J w.r.t. h_m. A closed form for the optimal a and b are obtained and the value of pair $\{f, \theta\}$ is found using an exhaustive search [7]. Next section shows how to include temporal consistency in the regression stumps.

2.1 Weak Learners with Temporal Consistency

In this work we add temporal consistency to the weak classifier response, h_m, using a set of T consecutive data points to perform the classification. The rationale is to use as much as possible of the information available in order to improve classifier output.

We propose to include the temporal window size T as an additional parameter of the regression stump. In this way the regression stump will only use advantageous information at each round, by choosing how many points to use depending on the feature values, opposed to the common approach of constant window length. Thus, consistency is included by defining the new Temporal Stumps as the mean classification output of the regression stump, in a temporal window of size T,

$$h_m^*(x_i) = \frac{1}{T}\sum_{t=0}^{T-1}\left(a\delta\left[x_{i-t}^f > \theta\right] + b\delta\left[x_{i-t}^f \leq \theta\right]\right). \tag{1}$$

The particular information extracted within the temporal window become clear if we put a and b in evidence,

$$h_m^*(x_i) = a\left(\frac{1}{T}\sum_{t=0}^{T-1}\delta\left[x_{i-t}^f > \theta\right]\right) + b\left(\frac{1}{T}\sum_{t=0}^{T-1}\delta\left[x_{i-t}^f \leq \theta\right]\right). \tag{2}$$

The new temporal weak classifier of Eq. 2 can be viewed as the classic regression stump with a different "indicator function". If $T = 1$ it becomes the original regression stump, and for $T > 1$ the indicator function changes. The new indicator functions

$$\varDelta_+^T(f,\theta,T) = \frac{1}{T}\sum_{t}^{T-1}\delta\left[x_{i-t}^f > \theta\right], \quad \varDelta_-^T(f,\theta,T) = \frac{1}{T}\sum_{t}^{T-1}\delta\left[x_{i-t}^f \leq \theta\right], \tag{3}$$

compute the percentage of points above and below the threshold θ, in the temporal window T and for the feature number f. The indicator functions with temporal consistency in Eq. 3, can take any value in the interval $[0\ 1]$, depending on the length of the temporal window used. For example, if $T = 2$ the functions can take 3 different values, $\varDelta_+^T \in \{0,\ 1/2,\ 1\}$, if $T = 3$ can take four values, $\varDelta_+^T \in \{0,\ 1/3,\ 2/3,\ 1\}$ and so on.

The fuzzy output of the new "indicator function", $\varDelta$, represents the confidence of threshold choice to use the data with temporal support T. Thus, at each boosting round, we use a weighted confidence of both branches, instead of choosing only one branch. We present experimental results that show that optimizing this fuzzy regression stump brings additional resistance to noise, thus increasing the generalization capabilities.

During classification of unseen data, the algorithm has the possibility to decrease the confidence measure, Δ, for instance if the new data is noisy when compared to the training data. This differs from the usual boosting binary decision and can be compared to what fuzzy trees brought to decision trees.

Replacing the weak classifier with temporal consistency of Eq. 2 in the cost function, we compute the optimal temporal stump parameters a and b,

$$a = \frac{\bar{\nu}_+\bar{\omega}_- - \bar{\nu}_-\bar{\omega}_\pm}{\bar{\omega}_+\bar{\omega}_- - (\bar{\omega}_\pm)^2}, \quad b = \frac{\bar{\nu}_-\bar{\omega}_+ - \bar{\nu}_+\bar{\omega}_\pm}{\bar{\omega}_+\bar{\omega}_- - (\bar{\omega}_\pm)^2}, \tag{4}$$

with $\begin{matrix} \bar{\nu}_+ = \sum_i^N w_i y_i \Delta_+^T, \ \bar{\nu}_- = \sum_i^N w_i y_i \Delta_-^T, \\ \bar{\omega}_+ = \sum_i^N w_i \Delta_+^T, \quad \bar{\omega}_- = \sum_i^N w_i \Delta_-^T, \quad \bar{\omega}_\pm = \sum_i^N w_i \Delta_-^T \Delta_+^T. \end{matrix}$
Note that all the above variables are functions of $\{f, \theta, T\}$ that we dropped for notation simplicity. To find the optimal f,θ and T we use exhaustive search.

Comparing the temporal weak learner with the original GentleBoost weak learner, we have an additional parameter T to optimize. The algorithm is similar to Gentleboost, now optimizing the presented temporal stump, h_m^*.

It is important to remark that the proposed framework is, as the classic boosting approaches, a non-sequential single-frame classifier. It should be used to classify data at one time instant with the internal difference that it "looks" back a few points in time, adding consistency to the decision. The data used does not need to have any special characteristic despite the fact of having some temporal sequence.

2.2 Results on Synthetic Data

We perform tests with synthetic data in order to illustrate the advantages of our algorithm over other boosting approaches: (i) improved noise robustness, and (ii) improved performance in data with large overlapping in the feature space. We create synthetic data and apply three boosting algorithms: (i) GentleBoost [4], (ii) TemporalBoost [5] and (iii) our optimal temporal boosting.

The aim is to learn two elliptic trajectories using point location as features, and then classify new points as belonging to one of the trajectories. The input features (x_i^1, x_i^2) are noisy observations of the actual ellipses, generated accordingly to: $x_i^1 = a\cos t + \mathcal{N}(0,\sigma)$ and $x_i^2 = b\sin t + \mathcal{N}(0,\sigma)$, where a and b are the major and minor axis, t represents time, and $\mathcal{N}(\mu,\sigma)$ is Gaussian noise. In Figure 1(a) we observe examples of trajectories with $\sigma = 0$, and $\sigma = 0.15$.

Figure 1(b) plots the evolution of the recognition rate along rounds for the test set. The experiment corresponds to trajectories corrupted by noise with $\sigma = 0.15$. Our boosting proposal clearly outperforms TemporalBoost and GentleBoost.

Noise robustness: We perform 15 experiments for each boosting algorithm, changing the noise variance linearly from 0 to 0.3. For each experiment, we compute recognition rate along 100 rounds and then pick the maximum value. In Figure 1(d) we plot the recognition performance maxima *vs* noise variance, showing experimentally that our

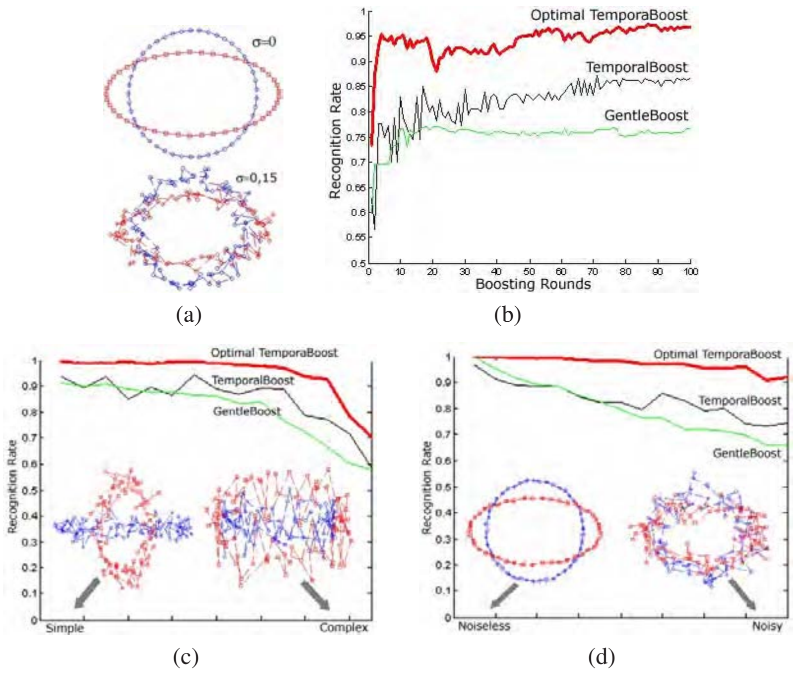


Fig. 1. Comparison of the recognition rate evolution for: GentleBoost, TemporalBoost and our OptimalTemporalBoost. The two class elliptical trajectories in (a) are used to compute the evolution along rounds, in (b), for the three algorithms. Picking the maximum of the evolution along rounds for each experiment we vary: the problem complexity presented in (c), and the varying the features noise in (d).

boosting algorithm is more robust to noise than GentleBoost and TemporalBoost. The reason for this behavior is the fuzzy output of the classifiers, that takes into account uncertainty in decisions at each boosting round.

Class overlap: In this experiment we increase gradually the amount of overlap among classes, keeping noise constant. Like in the previous experiment, we pick the best recognition rate in all rounds. In Figure 1(c) we see that our algorithm surpasses GentleBoost and TemporalBoost. The explicit inclusion of temporal parameters in the regression stump, and consequently joint optimization with the remaining ones can explain the improvement of our algorithm over TemporalBoost.

2.3 Comments

We show experimentally that our temporal GentleBoost algorithm increases noise robustness up to 20%, and is able to handle class overlapping in average with 10% better results than TemporalBoost. These results clearly indicate the advantage of using this new framework instead of other boosting approaches when working with temporal data. Additionally, the explicit time formulation in the regression stumps allow us to extend, in a straightforward manner, the temporal GentleBoost to multi-class problems.

3 Going to the Multi Class Problem

A Multi-class categorization problem is usually solved as a set of multi-binary problems where separate classifiers are trained and applied independently. As pointed by Torralba *et al* [7], for the object detection problem, this is a waste of resources because many of the features used can be shared among several classes. In the case of boosting, sharing helps to: i) reduce computational complexity by sharing weak classifiers among classes and ii) reduce the amount of training data needed in order to attain the same classification performance. We generalize the temporal GentleBoost to the multi-class problem likewise Torralba *et al* extended GentleBoost to multi-class problems.

3.1 The Temporal-JointBoosting Algorithm

The idea behind JointBoosting [7] is to share weak classifiers (and features) across classes. At each round the algorithm chooses a weak classifier that shares a feature among the subset of classes. The optimal subset of classes is chosen by minimizing the error cost function for all possible combinations.

The optimization to be solved has now one more variable, the classes, thus one must solve, $J = \sum_{c=1}^{C} \sum_{i=1}^{N} w_i^c (y_i^c - h_m(c, x_i))^2$, the new weighted least squares problem in each iteration, where $w_i^c = e^{-y_i^c h_m(c,x_i)}$ are the new class-specific weights. Shared stumps use the data that belongs to the optimal subset of classes as positive samples, and the remaining data as negative samples. For classes in the optimal subset $S(n)$, the stump function is similar to the binary case. For classes outside the optimal subset, the stump function is a class-specific constant, k^c. (see [7] for details). The shared temporal stump has the following form:

$$h_m(c, x) = \begin{cases} a_S \varDelta_+^T + b_S \varDelta_-^T & \text{if } c \in S(n) \\ k_S^c & \text{if } c \notin S(n), \end{cases} \tag{5}$$

where $\varDelta_{+/-}^T$ are the temporal consistency function defined in Eq. 3. The optimal parameters of the shared stump are:

$$a_S = \frac{\sum_{c\in S(n)} \bar{\nu}_+^c \sum_{c\in S(n)} \bar{\omega}_-^c - \sum_{c\in S(n)} \bar{\nu}_-^c \sum_{c\in S(n)} \bar{\omega}_\pm^c}{\sum_{c\in S(n)} \bar{\omega}_+^c \sum_{c\in S(n)} \bar{\omega}_-^c - \left(\sum_{c\in S(n)} \bar{\omega}_\pm^c\right)^2}, \tag{6}$$

$$b_S = \frac{\sum_{c\in S(n)} \bar{\nu}_-^c \sum_{c\in S(n)} \bar{\omega}_+^c - \sum_{c\in S(n)} \bar{\nu}_+^c \sum_{c\in S(n)} \bar{\omega}_\pm^c}{\sum_{c\in S(n)} \bar{\omega}_+^c \sum_{c\in S(n)} \bar{\omega}_-^c - \left(\sum_{c\in S(n)} \bar{\omega}_\pm^c\right)^2}, \tag{7}$$

$$k^c = \frac{\sum_i w_i^c y_i^c}{\sum_i w_i^c}, \quad c \notin S(n), \tag{8}$$

and obtain θ, f, T and $S(n)$, exhaustive search is performed [7].

3.2 Results on Synthetic Data

We apply multiclass versions of the previously used three boosting algorithms: (i) One against all version of the TemporalBoost, (ii) JointBoost, and (iii) JointBoost version of optimal temporal boost. In this case we aim to model several classes, and perform similar tests to the class overlap tests in the binary problem.

Five elliptical trajectories were generated for 10 levels of overlapping between classes, each one corresponding to 5 class classification problem. We vary the problem complexity, starting from the simplest case (easily separable), and increasing the proximity among classes toward a more complex problem. In Figure 2 we see that multi-class

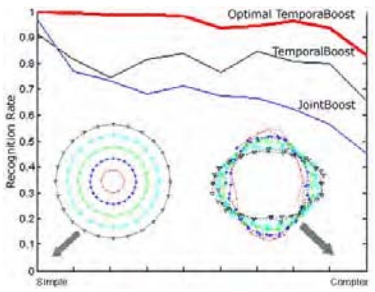


Fig. 2. Recognition rate in multi-class synthetic datasets, using 10 levels of increasing overlap among classes

boosting algorithms have a very similar behavior to the two-class boosting. In this case the classification rate improvement is larger than the binary counterpart, with our temporal version of JointBoost performing almost 20% better than TemporalBoost, and 30% better than the original JointBoost. Our JointBoost version of optimal temporal boost further improves the advantages over current state of the art methods when working with the intuitively more complex multiclass problem. The following step is to test our temporal JointBoost in real and very challenging datasets.

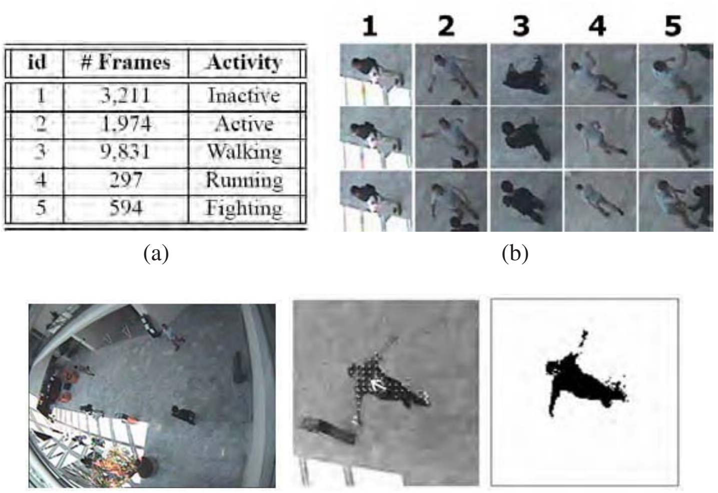
4 Human Activity Recognition

For the real datasets tests we consider the problem of human activity recognition using 2 different scenarios. Firstly we present results on the CAVIAR [8] scenario, a very challenging dataset due to: i) perspective distortion, ii) radial distortion and iii) the presence of a vanishing point in the image that makes human images varying from top view to side view. Using the CAVIAR dataset we aim to classify five general and basic human activities, {Active, Inactive, Walking, Running, Fighting}. The Active class considers movement of body parts that do not originate translation in the image.

The second dataset contains body specific movements that can be viewed as a detailed interpretation of the Active class. The movements considered are related to 2 body parts: i) the trunk and ii) the arms. For the trunk we recognize the movements of bending down/stand up and turning right/left. The arms movements comprise rising/putting down both right and left arms. We consider a total of 8 movements.

4.1 General Activities Recognition

The problem is to recognize five human activities from video sequences, {Active, Inactive, Walking, Running, Fighting}. A total of about 16,000 images with ground truth were used and are distributed according to table 3(a). Figure 3(b) shows tree examples of each considered activity. Figure 3(c) shows an image from the fighting scenario



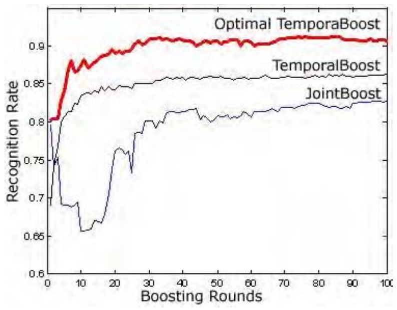
id	# Frames	Activity
1	3,211	Inactive
2	1,974	Active
3	9,831	Walking
4	297	Running
5	594	Fighting

Fig. 3. Considered activities and data distribution for the CAVIAR dataset (a) and example images for each class (b). The test scenario is exemplified in (c) and (d) presents the two types of information used to compute all the 29 features: the target velocity and the optical flow vectors (the optical flow represented in the image was sampled from the all the computed vectors).

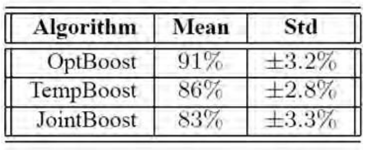
used to extract the examples from figure 3(b). Note the wide variability of this dataset, for example the third column (in figure 3(b)) correspond to the walking activity and in approximately one second the person changes from top to side view.

The features used to perform classification were obtained from the detected moving blobs in the scene that correspond to people. Once the information regarding the position of the target over time is provided, we compute 29 features based on 2 characteristics: i) the instantaneous position and velocity of the tracked subject and ii) the *optic flow* or instantaneous pixel motion inside the target's bounding box. An example of subject's velocity and optic flow is plotted in Figure 3(d). The rationale behind the 29 features is to model some important characteristics of people movements: i) speed, ii) regularity of the trajectory, iii) motion energy and iv) regularity of motion. We also use both instantaneous, averaged and second order moments of these quantities. A detailed description of the features can be found in [9].

We perform a leave one out subset process to compute the recognition rate, dividing the dataset into four different subsets (each one with similar number of frames). The definitive recognition rate is the average of the four tests. We compare three algorithms:



(a)



Algorithm	Mean	Std
OptBoost	91%	±3.2%
TempBoost	86%	±2.8%
JointBoost	83%	±3.3%

(b)

Fig. 4. Recognition rate in multi-class synthetic datasets, using 10 levels of increasing overlap among classes (a). Algorithms recognition rate comparison for the CAVIAR scenario(b) with best recognition rate and standard deviation (c).

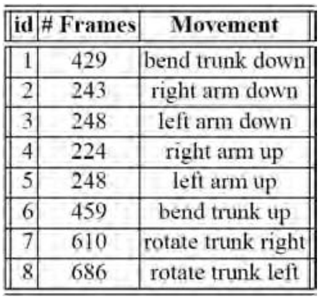
(i) One against all TemporalBoost, (ii) JointBoost and (iii) optimal temporal JointBoost. We show the average recognition rate for each boosting round in Figure 4(a) and the best recognition rate and correspondent standard deviation in table 4(b).

In this test, optimal temporal JointBoost outperforms one against all TemporalBoost by 5%, and JointBoost by 8%.

4.2 Body Parts Movements Recognition

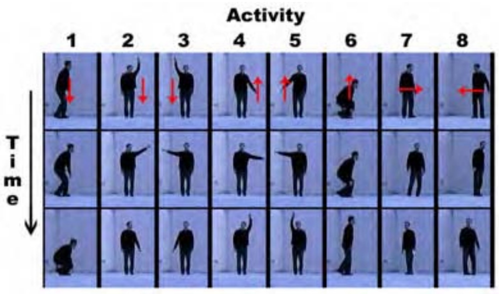
This group of tests aim to recognize body movements that do not originate translation of the target in the image. In Table 5(a) we see the data distribution for every type of body movement, and in Figure 5(b) examples of them are plotted.

The activities considered here are based on the movement of 2 body parts, the trunk and the arms, and for each movement several sequences were recorded. Each movement was performed in 2 different locations and/or performed in 2 different ways. For



id	# Frames	Movement
1	429	bend trunk down
2	243	right arm down
3	248	left arm down
4	224	right arm up
5	248	left arm up
6	459	bend trunk up
7	610	rotate trunk right
8	686	rotate trunk left

(a)



(b)

Fig. 5. Considered body part movements and data distribution in (a) and example images for the eight movements in (b).

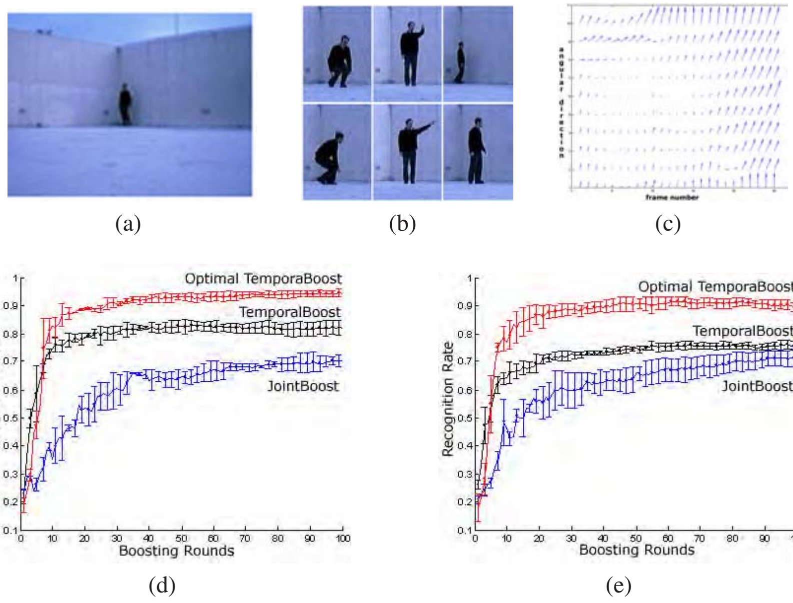


Fig. 6. Examples of: the global scenario (a), bending, rotating and rising right arm movements, illustrating the location and sigh differences (b) and the features computed for one rising left arm example (c). The algorithms recognition rate are compared when: the classifiers are trained with half of the sequences and tested with the remaining ones (d) and when train and test are done in different locations and sights (e).

example, turning is always performed in the same manner but in 2 different locations, one near the camera and one far from it. The opposite happens with the arms movement, they are performed in the same location but in two different sights. The bending movements are performed in both locations and sights (e.g. side and front sight). See figure 6(b) for example images illustrating this differences and figure 6(a) for the working scenario. In each case 4 different sequences where recorded, and the dataset contains a total of 3147 frames.

As features we consider optic flow referred to the person centroid in order to obtain only the relative motion of the different parts of the body. The idea is to have a qualitative measure of movements, without segmenting or identifying parts of the body, that would be a difficult and sometimes an impossible problem due to recurrent occlusions.

We assume that the body parts are arranged around the body centroid, thus the optical flow vectors are averaged along angular directions with the origin at the centroid. A detailed explanation of the feature extraction method can be found in [10]. The resultant features are vectors along a discrete number of angular direction that aim to represent the radial and normal movement of the person with respect to the correspondent angular direction. In figure 6(c) are plotted the features used, with 10 angular direction, for one of the rising left arm example.

To compare the performance of all the boosting algorithms we present in figure 6(d) the evolution of the recognition rate where the classifiers are trained with half of the sequences and tested with the remaining ones (performed twice by exchanging the test and training sets). In this experiment examples from the 2 locations and sights are present in the training set.

In figure 6(e) we evaluate the ability of our algorithm to recognize activities performed in different sights and image locations (scales). For this experiment we train the algorithms with sequences recorded at one location and one sight and in the test set we use sequences recorded at a different location and/or sight. Combining the two locations and sights we perform a total of four tests.

The results clearly show the advantage of using our algorithm, performing more than 11% better than TemporalBoost and 20% than JointBoost. In the more complicated test (figure 6(e)) the differences between the methods are even greather (15% in relation to TemporalBoost), with a 3% decreasing in the recognition rate of our method, when compared with the previous and much simpler test. These results indicate that we are able to recognise types of activities, even when performed differently, rather than exact types of movements.

4.3 Discussion

The single-frame overall recognition rate of 91%, in the general human activity, and 94%, in the specific body parts movements recognition tests are very good result, taking into acount the wide type of distortions present in the first scenario, that makes it one of the most challenging scenarios for this kind of task, and that we do not used any dynamic model of activity transitions. The inclusion of transitional dynamic models depends on the desired application, but the results presented here can be straightforward used as a lower level for several applications, e.g. urban surveillance, intelligent environments or human/robot interfaces and interaction. Furthermore, the inclusion of such higher level models (restrictions) should increase the recognition rate making the system very robust on real world aplpications.

5 Conclusions

We have proposed a methodology to handle temporal data consistency in non-sequential single-frame classification problems. Although the temporal evolution of the data might be crucial for certain classification problems (e.g. human activity), even when performing single-frame classification, it is rarely addressed at the level of the classifier.

We have adopted the boosting framework and propose a method whereby time is taken into account in the boosting optimization steps. In our work, temporal consistency is treated as part of the overall optimization procedure, contrasting with the heuristic approach adopted in the recently proposed Temporal AdaBoost.

More specifically, we extend the GentleBoost algorithm by modeling time as an explicit parameter to optimize. As a consequence of this time consistency model, we obtain a fuzzy decision function, depending on the temporal support of the data, which brings additional robustness to noise. Finally, we allow our temporal boosting algorithm to cope with multi class problems, following the JointBoosting approach.

We have conducted extensive tests to demonstrate the superiority of our approach when compared to the Temporal Boost and the GentleBoost algorithms. We use synthetic datasets with increasing complexity as well as real video data recognizing human activities. The results show that our method clearly outperform the Temporal Boosting algorithm by $5 - 10\%$ and standard GentleBoost algorithm by as much as $10 - 20\%$.

Using the real datasets we achieve performance always greater than 90% for a very challenging scenario, due to large image distortions. In the classification of more specific body parts movements the recognition rate is superior to 94%. This are very good results taking into acount that we perform single-frame classification without modeling the activities dynamics.

We present results that clearly show the importance of temporal data consistency in single-frame classification problems as well as the importance of as handling time in an optimal, fully consistent manner. Additionally, this new framework can be used with any type of sequential data, thus being applicable to a wide range of other problems, rather than the ones discussed here.

References

1. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. International Journal of Computer Vision (2), 153–161 (2005)
2. Dietterich, T.G., Ashenfelter, A., Bulatov, Y.: Training conditional random fields via gradient tree boosting. In: Proceedings of the ICML 2004, NY, USA, p. 28 (2004)
3. Efros, A.A., Berg, A.C., Mori, G., Malik, J.: Recognizing action at a distance. In: Proceedings of the ICCV 2003, p. 726. IEEE Computer Society Press, Los Alamitos (2003)
4. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28(2), 337–407 (2000)
5. Smith, P., da Vitoria Lobo, N., Shah, M.: Temporalboost for event recognition. In: Proceedings ICCV 2005, vol. 1, pp. 733–740 (2005)
6. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)
7. Torralba, A., Murphy, K., Freeman, W.: Sharing visual features for multiclass and multiview object detection. IEEE Transactions on PAMI 29(5), 854–869 (2007)
8. `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`
9. Ribeiro, P., Santos-Victor, J.: Human activities recognition from video: modeling, feature selection and classification architecture. In: BMVC Workshop on HAREM (2005)
10. Pla, F., Ribeiro, P.C., Santos-Victor, J., Bernardino, A.: Extracting motion features for visual human activity representation. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. LNCS, vol. 3522, Springer, Heidelberg (2005)

Automated Scene-Specific Selection of Feature Detectors for 3D Face Reconstruction

Yi Yao, Sreenivas Sukumar, Besma Abidi, David Page, Andreas Koschan, and Mongi Abidi

Imaging, Robotics, and Intelligent System Lab
The University of Tennessee, Knoxville, TN, 37996

Abstract. In comparison with 2D face images, 3D face models have the advantage of being illumination and pose invariant, which provides improved capability of handling changing environments in practical surveillance. Feature detection, as the initial process of reconstructing 3D face models from 2D uncalibrated image sequences, plays an important role and directly affects the accuracy and robustness of the resulting reconstruction. In this paper, we propose an automated scene-specific selection algorithm that adaptively chooses an optimal feature detector according to the input image sequence for the purpose of 3D face reconstruction. We compare the performance of various feature detectors in terms of accuracy and robustness of the sparse and dense reconstructions. Our experimental results demonstrate the effectiveness of the proposed selection method from the observation that the chosen feature detector produces 3D reconstructed face models with superior accuracy and robustness to image noise.

1 Introduction

The 3D reconstruction from uncalibrated video sequences has attracted increasing attention recently. Most of the proposed algorithms regarding feature matching and projective/metric reconstruction have applications in 3D reconstruction of man-made scenes [1, 2]. Recently, because of the difficulties in 2D face recognition caused by illumination and pose variations, recognition algorithms using 3D face models have emerged [3], which calls for reconstruction algorithms designed particularly for faces. Hu *et al.* used salient facial feature points to project a 2D frontal view image onto a 3D face model automatically [4] and illustrated improved face recognition rates using the 3D model despite pose and illumination variations. Chowdhury *et al.* reconstructed 3D facial feature points and obtained a 3D face model by fitting these points to a generic 3D face model [5].

Most existing 3D reconstruction algorithms start with feature selection and matching [1, 2, 5]. Based on the matched features in consecutive frames, 3D projective and metric structures are recovered. Therefore, the accuracy and robustness of feature detection and matching directly affect the overall performance of the reconstruction. Popular features for 3D reconstruction are image corners and lines. For a man-made scene, there exist well-defined corners, which facilitate the use of fast and straightforward feature detectors such as Harris corners. However, for face images, corners are not as distinguishable as in man-made scenes. In addition, face images include smooth areas, for example cheek and forehead, where feature matching becomes more ambiguous. Therefore, it is important to find an appropriate feature detector, which can make full use of facial features and avoid smooth areas simultaneously for 3D face reconstruction.

In this paper, we propose a data driven feature detector selection algorithm, where the optimal detector is dynamically selected according to different scene structures using a cost function based on information complexity (ICOMP) [6]. Our selection framework, referred to as *MuFeSaC* [7], improves existing algorithms by using an adaptive strategy for automatic extraction of relevant features from face images that contribute to facial structure and thus lead to improved accuracy and robustness of the resulting 3D reconstruction. An example face image, detected corners, and reconstructed 3D face model are shown in Fig. 1.

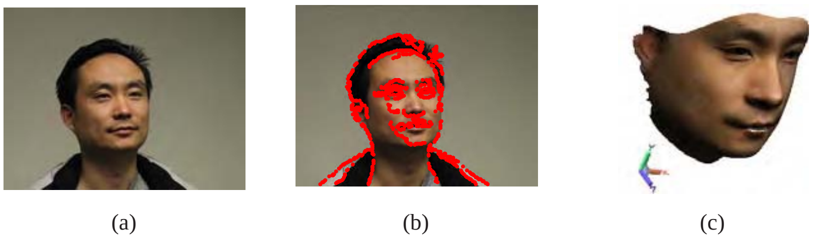


Fig. 1. Reconstructing 3D models of human faces from uncalibrated video sequences. (a) Input face image. (b) Corners relevant for 3D reconstruction. (c) Reconstructed 3D face model.

A performance comparison of various corner detectors using repeatability and information content can be found in [8]. The comparison was conducted based on various scene structures, including man-made and natural, and intended to provide general conclusions independent of input scene structures. In comparison, our algorithm is a data driven method which dynamically chooses the optimal feature detector based on the input sequences. Such a method is particularly useful for face reconstruction from surveillance videos where faces are tracked in various backgrounds.

We apply our selection scheme to face sequences, evaluate and compare the performance of the chosen detector against various types of feature detectors in terms of accuracy and robustness of the 3D face reconstruction, and prove that the chosen feature detector produces the best performance.

The major contributions of this paper are: (1) defining a data-driven selection framework that automatically chooses an appropriate feature detector for a face and eventually produces a better 3D reconstruction and (2) investigating the importance of feature detection methods in 3D reconstruction of faces by comparing results of several widely used corner detectors. Section 2 describes the 3D face reconstruction pipeline. Our automatic feature detector selection algorithm is discussed in Section 3. Experimental results are presented in Section 4 before drawing conclusions in Section 5.

2 3D Face Reconstruction

Since 3D face reconstruction is the target application of our automated feature selection algorithm, the overall process of constructing 3D face models is presented in this section. A schematic illustration of 3D reconstruction from a 2D uncalibrated image sequence with automated feature selection mechanism is shown in Fig. 2, which includes feature detection and matching, projective/metric reconstruction, 3D deformation, and texture mapping.

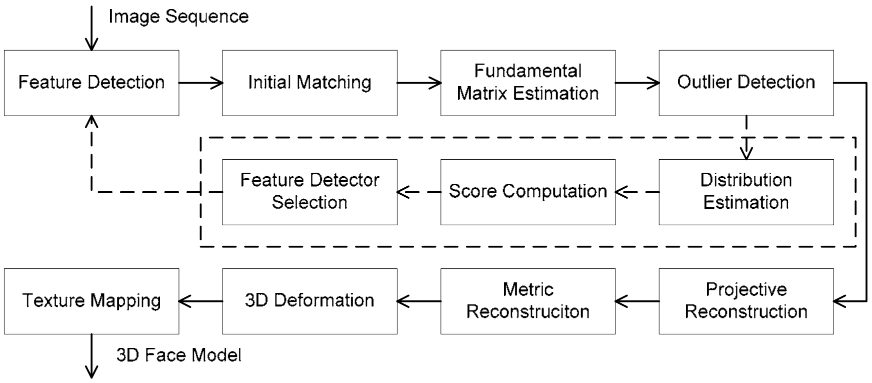


Fig. 2. Schematic illustration of 3D face reconstruction with automated selection of feature detectors

A two step feature matching process is employed. The initial matching is performed using a correlation based method. The RANSAC method is used for outlier detection and robust matching based on estimated fundamental matrices. Our automated feature selection method will be discussed in detail in Section 3. The selection of feature detectors can be performed offline for applications with consistent scene structures. Once the optimal feature detector is chosen, it can be applied directly for 3D reconstruction. However, for dynamic scenes where the characteristics of the scene structures are time varying, online selection is necessary.

We use the factorization approach for projective reconstruction, as demonstrated in [9]. Let the j^{th} point in the i^{th} frame, $\mathbf{x}_{ij}$, be projected from the scene point X_j by $\lambda_{ij}\mathbf{x}_{ij} = P_i X_j$, where λ_{ij} and P_i denote the projective depths and projection matrices, respectively. Given N_p matched points in N_f frames we have:

$$\begin{bmatrix} \lambda_{11}\mathbf{x}_{11} & \lambda_{12}\mathbf{x}_{12} & \cdots & \lambda_{1N_p}\mathbf{x}_{1N_p} \\ \lambda_{21}\mathbf{x}_{21} & \lambda_{22}\mathbf{x}_{22} & \cdots & \lambda_{2N_p}\mathbf{x}_{2N_p} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_{N_f1}\mathbf{x}_{N_f1} & \lambda_{N_f2}\mathbf{x}_{N_f2} & \cdots & \lambda_{N_fN_p}\mathbf{x}_{N_fN_p} \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{N_f} \end{bmatrix} \begin{bmatrix} X_1 & X_2 & \cdots & X_{N_p} \end{bmatrix}, \quad (1)$$

where the matrix on the left hand side is the measurement matrix. We start with an initial estimate of the projective depths. The initial depths can be set to ones or obtained using Sturm and Triggs' method [10]. The depths are then normalized so that $\sum_{i=1}^{N_f} \lambda_{ij}^2 \mathbf{x}_{ij}^T \mathbf{x}_{ij} = 1$ and $\sum_{j=1}^{N_p} \lambda_{ij}^2 \mathbf{x}_{ij}^T \mathbf{x}_{ij} = 1$. We find the nearest rank-4 approximation of the measurement matrix using SVD, based on which 3D reconstructed points are derived. These reconstructed points are reprojected into each image to obtain new estimates of the depths. The process is repeated until the variations in the projective depths are negligible.

We then find a transformation matrix H and upgrade the projective structure by HX_j. Using the dual absolute quadric Ω^*, we have $\omega_i^* \sim P_i \Omega_1^* P_i^T$ where $\omega_i^* = K_i K_i^T$ with K_i as the camera's intrinsic matrices [11]. A linear solution of Ω^* can be obtained by imposing additional constraints on the camera's intrinsic parameters, such as zero skew, unit aspect ratio, and zero principal point, and the rank-3 property is applied for improved accuracy. The transformation matrix is then obtained by forcing $H\Omega_1^* H^T = diag(1,1,1,0)$ and projective reconstruction is elevated to metric reconstruction by $P_{E,i} = P_i H^{-1}$ and $X_{E,j} = HX_j$. Finally bundle adjustment is carried out to minimize the projection errors: $\min \sum_{i,j} \left\| \mathbf{x}_{ij} - P_{E,i} X_{E,j} \right\|^2$.

The output of the previous module is a cloud of points but not a smooth surface representing a human face. By assuming that sufficient face information is embodied in the sparse point cloud, we deform a generic 3D face model. The vertices in the generic mesh are deformed using energy minimization principles similar to [5]. The procedure matches features in the generic mesh model by rotating R, translating $\mathbf{t}$ and scaling s 3D points and aligning the two models using an energy minimization function $E(s,R,\mathbf{t}) = \sum_j \| X_{E,j} - X_{M,j} \|^2$, where $X_{E,j}$ is the reconstructed point and $X_{M,j}$ refers to the point in the generic mesh. Once the initial alignment is obtained, the points in the generic mesh model are refined by weighing the distance along the surface normal of the mesh and the nearest reconstructed points and then deformed to preserve the features of the face reconstructed from the image sequence.

Once a 3D face mesh model is obtained, texture mapping is carried out by constructing the texture on a virtual cylinder enclosing the face model. A color is associated with each vertex of the deformed model by computing blending weights of each rectified image used for 3D reconstruction. The blending weights are based on the angle between the reconstructed camera's direction and the surface normal of the deformed mesh model.

3 Feature Selection

With the processing pipeline described in Section 2, our initial experiments indicated that sparse reconstruction had a major influence on the accuracy of dense reconstruction, and that the choice of the feature detection method was critical in converting the image sequence into a meaningful and accurate sparse 3D point cloud. This observation motivated our study of different feature detectors leading to the definition of *MuFeSaC* [7], short for multiple feature sample consensus, as an extension of RANSAC used in the sparse reconstruction pipeline to include multiple feature detectors.

The contribution of *MuFeSaC* over RANSAC is an inference engine that, in addition to finding the parameters of the interest model fit based on noisy data, also evaluates the confidence in the parameter estimates. *MuFeSaC* operates towards computing confidence scores from RANSAC iterations at the same time combining the confidence from one single feature detector with the information from other competing interest points, thereby reducing the risk due to the choice of the feature detector. We list the different stages of the *MuFeSaC* procedure in Table 1 and explain the model selection criteria called information complexity that acts as the consensus scoring tool. The implementation details of single feature outlier consensus and competing feature consensus are discussed in Section 3.1 and 3.2, respectively.

Table 1. Pseudo code of *MuFeSaC*

```
1.  For each feature detector $FD_i$, $i = 1,2,3\ldots N$
        a) Extract interest points from two successive frames.
        b) Find the putative matches using proximity and cross correlation.
        c) Perform RANSAC and iterate to a convergence. Collect d-estimated parame-
           ters S of model M fitted during the iterations of RANSAC.
        d) Estimate probability distribution $B_i$ based on n ($n > 30$) iterations of parame-
           ter estimates ($S_1 \ldots S_n$) collected.
    End
2.  Score Single Feature Outlier Consensus ($SFOC_i$) using the model selection criterion.
3.  Compute Competing Feature Consensus Score ($CFCS_i$) by evaluating competing
    distributions $B_i$ for different hypothesis.
4.  Choose the optimal feature detector with minimum $SFOC_i + CFCS_i$.
  Repeat steps 1-4 every k frames. (Typically k=10 for face videos)
```

3.1 Single Feature Outlier Consensus

If we were to choose the optimal feature detector based on the RANSAC convergence consensus alone, we would ideally want to pick the method that is indicative of maximum likelihood of the parameters of the model fitted by RANSAC with minimum uncertainty, or in simpler words B_i with minimal variance. This can be mathematically expressed as the minimizer of criterion (2) that simultaneously considers the likelihood and also penalizes the uncertainty associated with the likelihood of the parameters of model *M*. This model selection criterion in the statistics literature [6] is known as ICOMP and derives from the Kullback-Liebler (KL) distance between estimated and unknown underlying probability densities. Without much modification, we are able to apply this criterion in evaluating the confidence in the model fit during the iterations of RANSAC with each feature. We note that Eq. (2) does not involve distributional assumptions and can be applied to even Parzen window estimates of B_i.

$$\begin{aligned} SFOC_i &= \text{Lack of fit} + \text{Profusion of uncertainty} \\ &= -2 \log (Likelihood\ of\ \mu_i) + 2\, C_1 (F^{-1}(\Sigma_i)), \end{aligned} \tag{2}$$

where F^{-1} is the inverse Fisher information matrix, μ_i and Σ_i are the maximum likelihood estimates of the mean and covariance computed as the first two moments of B_i. The C_1 measure and the F^{-1} are computed using Eq. (3) and (4):

$$C_1(F^{-1}(\Sigma_i)) = \frac{s}{2}\log\left[\frac{tr(F^{-1}(\Sigma_i))}{s}\right] - \frac{1}{2}\log\left|F^{-1}(\Sigma_i)\right|, \tag{3}$$

where s denotes the rank of F^{-1}, |•| refers to the determinant, *tr* refers to the trace of the matrix, and

$$F^{-1}(\Sigma_i) = \begin{bmatrix} \Sigma_i & 0 \\ 0 & D^{+}_{p}(\Sigma_i \otimes \Sigma_i)D^{+}_{p}{}' \end{bmatrix}, \tag{4}$$

with D^{+}_{p} being the Moore-Penrose inverse of vectorized Σ_ι and $\otimes$ representing the Kronecker product. The C_1 measure for penalizing uncertainty is obtained by maximizing mutual information in d-dimensions. We direct the reader to [6] for more implementation details on the finite sampling form of Eq. (2).

3.2 Competing Feature Consensus

The $CFCS_i$ quantifies the agreement between the competing models M fit by RANSAC from each feature detector. The score is obtained by first evaluating different hypothesis listed below and then choosing the optimal consensus combinatorial cluster among competing feature detectors:

Case 1: All B_i's maximizing the likelihood of the same parameters for model M. All μ_i*'s* equal and Σ_i*'s* equal.
Case 2: All μ_i*'s* equal but Σ_i*'s* are not.
Case 3: All μ_i*'s* and Σ_i*'s* unequal, but there exists a maximal cluster of μ_i*'s* equal.

The verification of these hypotheses is like performing multi-sample clustering based on information distances in an entropic sense as described by Bozdogan in [6]. We follow a similar approach to verify these three cases, by considering the samples that contribute to distributions B_i to have to come from the same distribution and evaluate the complexity in model-fitting as the criterion to decide which of the three cases has occurred. We use the Akaike information criterion (AIC) to score the different hypothesis based on the likelihood of feature cluster L and parameter parsimony estimation m.

$$\begin{aligned} AIC(\mu_i, \Sigma_i, \kappa) &= -\text{(Likelihood of feature cluster)} + \\ &\quad \text{Parameter parsimony after clustering} \\ &= -2\log L + 2m. \end{aligned} \tag{5}$$

The evaluation of the likelihood of feature cluster L only considers the samples that contributed to the distributions B_i's within the cluster evaluated for consensus. We evaluate the parameter parsimony factor for the 2^N different cluster combinations based on the formulae listed in Table 2. The hypothesis that has minimum AIC is the statistical decision. Initially, we only evaluate the three case hypotheses. This initial

3-case hypothesis verification can avoid the combinatorial evaluations when all methods are accurate. We assign the minimizer of the AIC for the 3-case hypothesis as $CFCS_i$ to the corresponding feature detectors. If the minimizer indicates the occurrence of Case 2 or 3, we perform the evaluation on all combinatorial "feature detector" clusters shown under Case 3 in Table 2. The minimizer of the AIC score on these sub-clusters points to the cluster with maximal κ feature detectors contributing to the same model parameters. This AIC score is assigned only to the "feature detectors" within the maximal cluster. This cluster evaluation procedure eliminates the possibility that we select a feature detector that has minimal outliers but is giving us totally different parameters after RANSAC convergence.

We just add the two definitions of information measures which is a common practice with log utility functions. Also, we note that our formulation with $SFOC_i$ minimizes the error in the model used for estimating the geometry from successive views. On the other hand, $CFCS_i$ takes care of the risk in the model itself by inferring from different model generators in feature detectors. Thus, with *MuFeSaC* automatically selecting the interest points from a face video sequence, we now present experimental results.

Table 2. An example of parameter parsimony estimation (m) for a simple d-parameter model M with $N = 3$

	κ	**Clustering**	m
Case 1	3	(F_1,F_2,F_3)	$d+ d(d+1)/2$
Case 2	1	$(F_1)(F_2)(F_3)$	$Nd +d(d+1)/2$
Case 3	2	$(F_1,F_2)(F_3)$ $(F_1,F_3)(F_2)$ $(F_2,F_3)(F_1)$	$\kappa d + \kappa d(d+1)/2$

4 Experimental Results

Eight face images of each subject, collected from different viewpoints, are used for 3D face reconstruction. Five types of corner detectors are implemented: curvature corner [12], Harris corner [13], STK [13], phase congruency corner (PCC) [14], and FAST [15]. These feature detectors are chosen for the different heuristics that motivates them, Harris corners being intensity gradient-based, phase congruency being spatial-frequency inspired, and curvature corners being edge-derived. We compare the 3D reconstruction using the sparse point cloud and dense 3D face model. The sparsely reconstructed point cloud is an intermediate output, the accuracy of which determines the performance of the successive dense reconstruction. The performance of the sparse reconstruction provides a direct measure of evaluating the influence of the corner detector on the final 3D face reconstruction excluding factors which may come into play from 3D deformation. Hence we compare results at both stages of reconstruction. In the interest of space, only the results from one subject are presented with similar observations applicable to other tested subjects. The curvature corner was chosen as the optimal detector for this sequence by our automated selection algorithm. Therefore, in the following discussion, its performance is compared against the performances of other tested corner detectors.

4.1 Sparse Reconstruction

Fig. 3 illustrates our experimental results and compares the performance of the chosen feature detector, curvature corner, against other corner detectors. Sample images with detected corners show that corner detectors behave in different ways for face images which include both clustered corners (eyes and mouth) and smooth areas (cheek and forehead). Curvature, FAST, and PCC detectors focus on image edges while the corners detected by Harris and STK also appear in smooth areas.

The reconstructed 3D point clouds are shown from the best viewpoint in terms of illustrating the facial structure. From visual inspection of these 3D plots, we see that the reconstructed structure using curvature corners, the chosen feature detector, yields the best visual representation. From Harris and FAST corner detectors, visible facial structures are obtained with an increased number of noisy points. As for the STK corner detector, the noise level further increases resulting in a structure barely distinguishable. In comparison, the reconstructed structure from the PCC detector has most of the points with good approximation but a few points with large errors.

We use the variations in the projective depth to describe the accuracy and stability of the reconstruction. From actual measurement, the variation in depth in the world coordinates should be within 30cm, which corresponds to 0.375 when normalized to the camera coordinates. The back projected depths in the camera coordinates are plotted for all detected corners in all frames in Fig. 3. We use the percentage of the 3D points with a projective depth exceeding the theoretical range to quantitatively evaluate and compare the performance of various feature detectors, as shown in Fig. 4. All of the back projected depths from the curvature corner detector fall within the theoretical range. The variations from other detectors exceed the theoretical range with the STK detector yielding the most variations. The frequency and magnitude of variations exceeding the theoretical range are from noisy reconstruction caused by erroneous corner matching. Based on this quantitative measure, the chosen feature detector, curvature corner, has the best performance. From both visual inspection and quantitative measure, the chosen detector produces properly spaced corners around major facial features and hence generates the most accurate and robust sparse reconstruction.

4.2 Dense Reconstruction

The final 3D dense model is obtained by deforming a generic face model using the point cloud recovered from sparse reconstruction. The reconstructed face model is compared with a 3D reference model scanned using a GenexFace Cam. The original scan is processed in RapidForm to remove holes and spikes. The deviation from the reference model describes the accuracy of the reconstruction and hence the efficiency of the corresponding corner detector. Fig. 5 illustrates the recovered 3D dense face models and their deviation from the reference model. The result from the FAST detector is not included because the sparse reconstruction does not have sufficien depths recovered on identifiable facial features. From Fig. 5, we observe that the dense reconstruction based on the chosen detector, curvature corner, yields the smallest maximum deviation (5.32) as compared with Harris (5.50), STK (5.93), and PCC

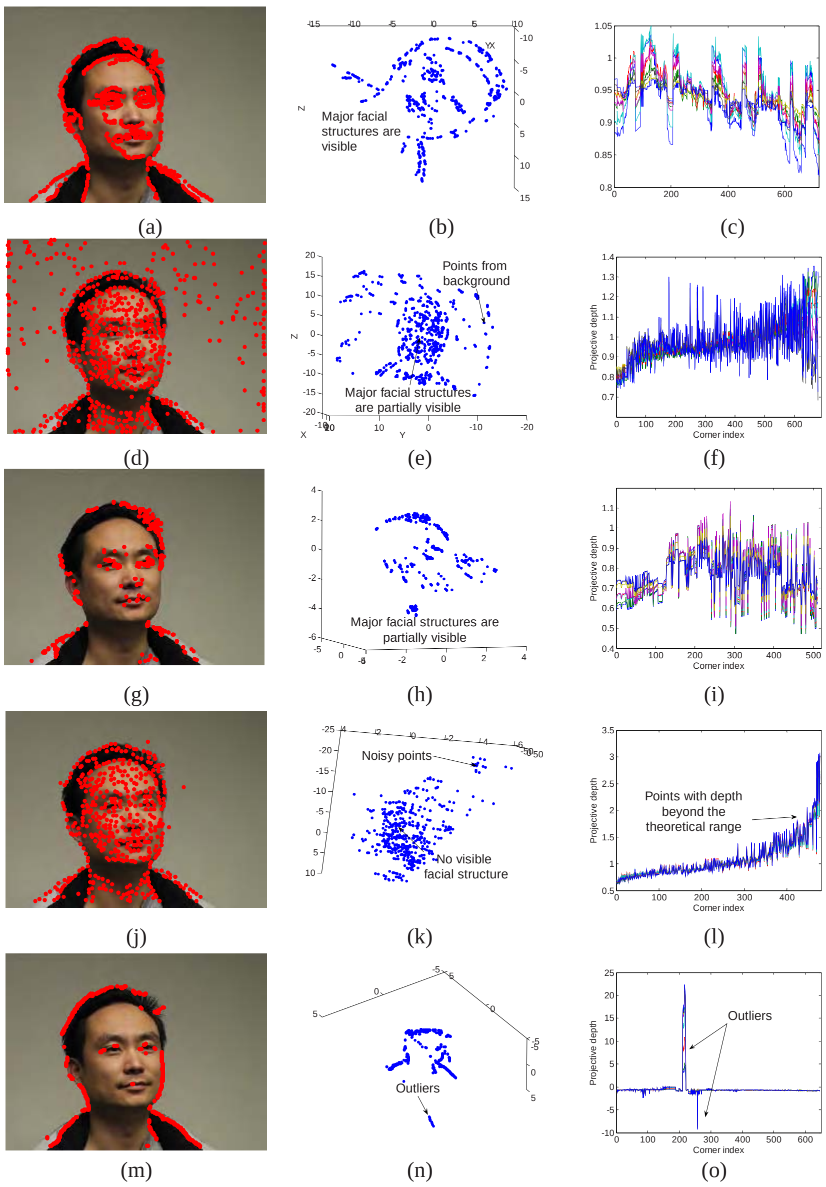


Fig. 3. Performance comparison of sparse reconstruction. Sample face images with detected corners: (a) curvature corner (the chosen feature detector by our automated selection algorithm for the illustrated sequence), (d) Harris, (g) FAST, (j) STK, and (m) PCC. 3D plots of sparsely reconstructed point clouds shown in the best view for illustrating facial structures: (b) curvature corner, (e) Harris, (h) FAST, (k) STK, and (n) PCC. Plots of projective depths: (c) curvature corner, (f) Harris, (i) FAST, (l) STK, and (o) PCC. Colored curves illustrate the projective depths from different frames. The 3D sparse reconstruction using the chosen feature detector, curvature corner, preserves major facial structures and produces accurate and robust reconstruction with all projective depths in the theoretical range.

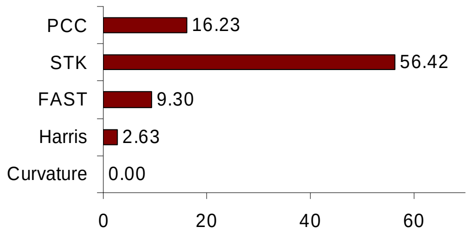


Fig. 4. Quantitative performance comparison of sparsely reconstructed 3D point clouds using various corner detectors: the percentage of reconstructed points with a projective depth exceeding the theoretical range. A smaller number suggests better accuracy. The point cloud from the chosen feature detector, curvature corner, has the most accurate variations in depth estimation.

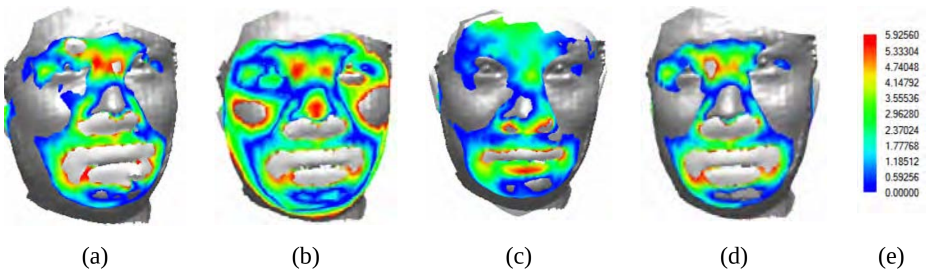


Fig. 5. Performance comparison of 3D dense reconstruction: (a) curvature corner, (b) Harris, (c) STK, and (d) PCC. The reconstructed 3D face model is compared with a reference 3D scan of the face. The reference 3D face model is shown in grey. The deviations from the reference model are shown in color code (e). The reconstruction based on the chosen detector, curvature corner, produces the smallest maximum and average deviations.

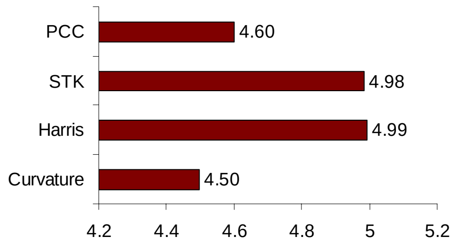


Fig. 6. Quantitative performance comparison (Hausdorff distance) of the recovered 3D face model using various corner detectors. The reconstruction using the chosen detector, curvature corner, yields the smallest deviation from the reference model.

(5.36). We also list the Hausdorff distance as a quantitative measure of the accuracy of dense reconstruction in Fig. 6. We have used the Hausdorff distance instead of the commonly used ICP metric because the point clouds are not of comparable resolution. Moreover, Hausdorff distance provides an unbiased basis for the comparison of point clouds from different feature point detectors. As expected the 3D model using the chosen detector, curvature corner, produces the smallest Hausdorff distance and thus the best accuracy. Note that to obtain a meaningful reconstruction, the outliers from PCC are excluded before deforming the generic face model. Therefore, the final 3D reconstruction using PCC presents a comparable accuracy as curvature corners
Compared with the results from sparse reconstruction, we observe that the performance gap between the curvature and other corner detectors decreases for dense reconstruction. This is due to the use of the generic face model as a constraint in the optimization process solving for the final dense reconstruction, which imposes additional bounds on the variations in the structure and reduces the influence of the errors from 3D sparse reconstruction. However, the use of the generic model may also reduce the useful discriminant structures for differentiating different subjects. In our future work, principal component analysis (PCA) will be explored for improved accuracy in 3D deformation. With the introduction of more principal components in addition to the generic face model, which is actually an averaged face model, more useful discriminant structures can be reserved. With PCA, the accuracy of sparse reconstruction plays more important role and hence more substantial performance difference can be observed by using different feature detectors.

5 Conclusions

In this paper, we proposed an automatic feature selection algorithm that adaptively chooses the optimal feature detector according to the input data for the purpose of 3D face reconstruction from uncalibrated image sequences. Several widely used corner detectors were implemented and served as competitive candidates for our selection algorithm. Experiments based on various subjects were performed to qualitatively and quantitatively compare the accuracy and robustness of the sparse and dense 3D face reconstructions. The chosen detector by our method produced a 3D face reconstruction with the best accuracy and robustness, which proves the effectiveness of the proposed selection algorithm.

References

1. Pollefeys, M., Gool, L., van Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J.: Visual modeling with a hand-held camera. Int'l J. of Computer Vision 59, 207–232 (2004)
2. Kien, D.T.: A review of 3D reconstruction from video sequences. MediaMill3D Technical Report, University of Amsterdam (2005)
3. Lu, X., Jain, A.K., Colbry, D.: Matching 2.5D face scans to 3D models. IEEE Trans. on Pattern Analysis and Machine Intelligence 28, 31–43 (2006)
4. Hu, Y., Jiang, D., Yan, S., Zhang, L., Zhang, H.: Automatic 3D reconstruction for face recognition. In: IEEE Int'l Conf. on Automatic Face and Gesture Recognition, Washington D.C, pp. 843–848 (2004)

5. Chowdhury, A.R., Chellappa, R., Krishnamurthy, S., Vo, T.: 3D face reconstruction from video using a generic model. In: IEEE Int'l Conf. on Multimedia and Expo, Lausanne, Switzerland, pp. 449–452 (2002)
6. Bozdogan, H.: Akaike's information criterion and recent developments in information complexity. J. of Mathematical Psychology 44, 62–69 (2000)
7. Sukumar, S.R., Bozdogan, H., Page, D., Koschan, A., Abidi, A.: MuFeSaC: Learning when to use which feature detector. In: Accepted to the IEEE International Conference on Image Processing, San Antonio, TX (2007)
8. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. Int'l J. of Computer Vision 37, 151–172 (2000)
9. Mahamud, S., Hebert, M., Omori, Y., Ponce, J.: Provably-convergent iterative methods for projective structure from motion. In: IEEE Conf. on Computer Vision and Pattern Recognition, Kauai, Hawaii, pp. 1018–1025 (2001)
10. Sturm, P., Triggs, B.: A factorization based algorithm for multi-image projective structure and motion. In: European Conf. on Computer Vision, Cambridge, England, pp. 709–720 (1996)
11. Pollefeys, M., Koch, R., van Gool, L.: Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters. In: IEEE Int'l Conf. on Computer Vision, Bombay, India, pp. 90–95 (1998)
12. He, X.C., Yung, N.H.C.: Curvature scale space corner detector with adaptive threshold and dynamic region of support. In: Int'l Conf. on Pattern Recognition, Cambridge, England, pp. 791–794 (2004)
13. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. Int'l J. of Computer Vision 1, 63–68 (2004)
14. Kovesi, P.: Phase congruency detects corners and edges. In: Australian Pattern Recognition Society Conf., Sydney, Australian, pp. 309–318 (2003)
15. Rosten, E., Drummond, T.: Machine learning for high speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)

A Dynamic Component Deforming Model for Face Shape Reconstruction*

Xun Gong[1,2] and Guoyin Wang[2]

[1] School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, 610031, P.R. China
gongxun@foxmail.com
[2] Institute of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, 400065, P.R. China

Abstract. A novel method, called dynamic component deforming model, is proposed to reconstruct the face shape from a 2D image based on feature points. Assuming that human face belongs to a linear class, principal components learned from a 3D face database are used in order to constrain the results. Different from the fixed components used in the traditional methods, the significance of each component is investigated while the most correlative ones are selected as the basic space. This novel representation is able to fit a more exact 3D shape for an individual than the known methods as the useless data are excluded. Comparison results show that the proposed method achieves good results on both contrived data with known ground truth together with real photographs.

1 Introduction

Accurate face modeling has extensive application in film, games, video conference and so forth. Since the early 1970's, in fact, modeling photorealistic 3D face has been a fascinating yet challenging task in computer vision and graphics.

3D face reconstruction involves two phases, shape recovering and texture synthesis. Most of the current researches as well as this paper mainly focus on how to modeling an accurate shape, which is considered as the essential attribute of a face. Shape obtained by laser-scanner is accurate enough but the requirements of an expensive equipment along with unknown manually processes have prevented its popularity in practice [1,2]. On the other hand, reconstruction from multiple view images or video has produced a number of structure-from-motion techniques [3,4,5]. Some of those approaches could recover shape information accurately in a certain degree. But long video sequences or large number of images are taken as inputs, the computational complexities of those methods are always high and their results are highly impacted by the feature correspondence between frames. Another impressive approach, named morphable model [6,7,8,9], could get face model reconstruction automatically from a single image, has demonstrated the

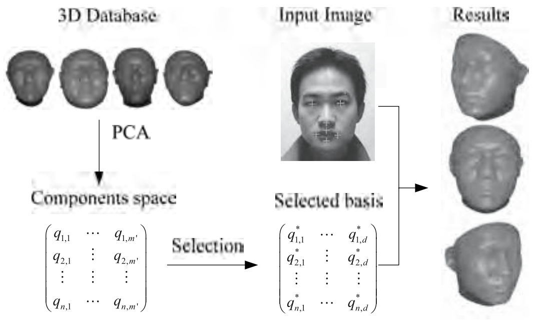
* This paper is partially supported by Program for New Century Excellent Talents in University (NCET).

advantage of using linear classes model. However, the iterative optimization in this analysis-by-synthesis system is computationally expensive and always converges to local minimum leading to unrealistic faces.

Generating a 3D facial model from a few feature points is potentially providing a tradeoff between quality and speed. A representative work had been done by Chai and Jiang [10,11]. However, their method aimed for aiding recognition, thus did not provide accurate reconstruction. A similar method has also been proposed by Vetter *et al.* [12,13], which are used for restoring the missing data. A dynamic component deforming model (DCDM) proposed in this paper is related to this approach. The main insight of our work is that overfitting could be alleviated by appropriately selecting the correlative components. Modeling on synthesized faces has validated this point when comparing with the ground truth. The detailed procedure of proposed method is depicted in Fig.1.

2 Shape Reconstruction from Features

Prototypic 3D faces in BJUT-3D Face Database [14] are used as prior information. Dense point-to-point correspondence between faces is solved automatically by a 2D templates based algorithm we developed in literature [15].



$$\begin{pmatrix} q_{1,1} & \cdots & q_{1,m'} \\ q_{2,1} & \vdots & q_{2,m'} \\ \vdots & \vdots & \vdots \\ q_{n,1} & \cdots & q_{n,m'} \end{pmatrix} \xrightarrow{\text{Selection}} \begin{pmatrix} q^*_{1,1} & \cdots & q^*_{1,d} \\ q^*_{2,1} & \vdots & q^*_{2,d} \\ \vdots & \vdots & \vdots \\ q^*_{n,1} & \cdots & q^*_{n,d} \end{pmatrix}$$

Fig. 1. System overview. A 3D shape is computed from a set of 2D features.

2.1 Statistic Model of Linear Shape Class

After correspondence, a face shape $\boldsymbol{s}_i$ can be denoted by a vector:

$$\boldsymbol{s}_i = (x_1, y_1, z_1, \cdots, x_k, y_k, z_k, \cdots x_n, y_n, z_n)^T \in \Re^{3n}, i = 1, \cdots, m, \tag{1}$$

where (x_k, y_k, z_k) is the coordinates of the k-th vertex v_k, n is vertices's number. A group of faces can be denoted by $\boldsymbol{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_m)$, then, a novel face $\boldsymbol{s}_{new}$ is a linear combination of the known faces:

$$\boldsymbol{s}_{new} = \boldsymbol{S} \cdot \boldsymbol{\alpha}, \tag{2}$$

where, $\boldsymbol{\alpha} = (a_1, \cdots a_i, \cdots, a_m)^T$, $a_i \in [0, 1]$, and $\sum\limits_{i=1}^{m} a_i = 1$.

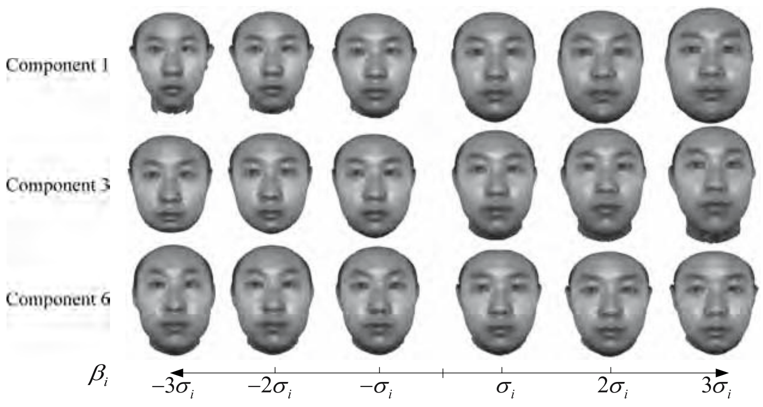


Fig. 2. Deforming effect of eigenvectors. The average face is morphed by the 1st, 3rd and 6th eigenvectors, respectively, the coefficients β_i changes between $-3 \sim +3$

To remove the correlation between faces, the principal component analysis (PCA) is applied to the prototypes. Let $\boldsymbol{Q} = (\boldsymbol{q}_1, \cdots, \boldsymbol{q}_{m'})$ be the eigenmatrix by concatenating the prior $m'(\leq m-1)$ eigenvectors, the corresponding eigenvalues are $\boldsymbol{\sigma} = (\sigma_1^2, \cdots, \sigma_{m'}^2)$, where $\sigma_1^2 \geq \cdots \geq \sigma_{m'}^2$. Using the scaled eigenmatrix $\boldsymbol{Q}_s = (\sigma_1 \boldsymbol{q}_1, \cdots, \sigma_{m'} \boldsymbol{q}_{m'})$ as the basis [12], then, Eq. (2) can be rewritten as:

$$\boldsymbol{s}_{new} = \bar{\boldsymbol{s}} + \boldsymbol{Q}_s \cdot \boldsymbol{\beta} = \bar{\boldsymbol{s}} + \Delta \boldsymbol{s}, \tag{3}$$

where $\boldsymbol{\beta} = (\beta_1, \cdots \beta_i, \cdots, \beta_{m'})^T \in \Re^{m'}$, $\bar{\boldsymbol{s}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{s}_i$, the probability of the parameter is given by $p(\boldsymbol{\beta}) \sim e^{-\frac{1}{2}\|\boldsymbol{\beta}\|^2}$. Fig.2 illustrates the effect of principal components and their coefficients $\boldsymbol{\beta}$ in face deformation. Obviously, getting $\boldsymbol{\beta}$ is the key of computing $\Delta \boldsymbol{s}$, which will be discussed in the next subsection.

2.2 Adjusting Shape Based on Features

Eq. (3) indicates that a special face can be obtained by adding a deriation $\Delta \boldsymbol{s}$ to an average face. Since the displacements of only $k(k \ll n)$ feature points $\Delta \boldsymbol{s}^f$ are known (see Fig. 3), we have to use these values to modify the position of remaining vertices. Let $\boldsymbol{s}^f \in \Re^l$ (for 2D coordinates $l = 2k$, $l = 3k$ while use 3D coordinates) be the features' position on a special face, then

$$\boldsymbol{s}^f = \boldsymbol{L}\boldsymbol{s}, \qquad \boldsymbol{L} : \Re^{3n} \mapsto \Re^l, \tag{4}$$

where, $\boldsymbol{L}$ is an implicit mapping, which is a projection operation for feature selection. Likewise, a feature based scaled eigenmatrix is:

$$\boldsymbol{Q}_s^f = \boldsymbol{L}\boldsymbol{Q}_s = (\sigma_1 \boldsymbol{q}_1^f, \cdots, \sigma_{m'} \boldsymbol{q}_{m'}^f) \in \Re^{l \times m}. \tag{5}$$

Base on Eq. (3) and Eq. (4), thus,

$$\Delta \boldsymbol{s}^f = \boldsymbol{L}(\boldsymbol{s} - \bar{\boldsymbol{s}}) = \boldsymbol{L}(\boldsymbol{Q}_s \cdot \boldsymbol{\beta}) = \boldsymbol{Q}_s^f \cdot \boldsymbol{\beta}. \tag{6}$$

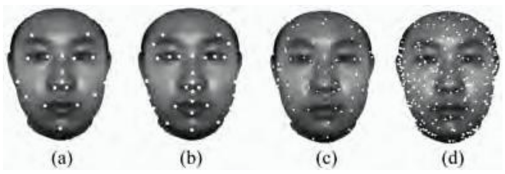


Fig. 3. The sets of 18, 29, 80 and 300 feature points on an average face used for modeling. (a),(b) are the salient features, (c),(d) are selected randomly.

To compute $\boldsymbol{\beta}$ directly from Eq. (6) may lead to overfitting [10] and even get unhuman being like faces because of noises. An alternative solution given by Vetter *et al.* [12] is using a statistical optimum approach to get the coefficients. The object function based on Bayesian posterior probability is given as

$$E(\boldsymbol{\beta}) = \left\|\boldsymbol{Q}_s^f \cdot \boldsymbol{\beta} - \Delta \boldsymbol{s}^f\right\|^2 + \eta \cdot \|\boldsymbol{\beta}\|^2 , \tag{7}$$

where the first term is the reconstruction error of the control points, and the second term is a regularization to control the plausibility of the reconstruction. $\eta \geq 0$ is a weight factor to control a tradeoff between the precision of feature point matching and prior probability. It has been proved that:

$$\arg\min \|E(\boldsymbol{\beta})\| = \boldsymbol{V} \cdot \left(\frac{\lambda_i}{\lambda_i^2 + \eta}\right) \cdot \boldsymbol{U}^T \cdot \Delta \boldsymbol{s}^f , \tag{8}$$

where, according to singular value decomposition (SVD), $\boldsymbol{Q}_s^f = \boldsymbol{U\Lambda V^T}$, $\boldsymbol{U} \in \Re^{l\times l}$, $\boldsymbol{V} \in \Re^{m\times m}$, $\boldsymbol{\Lambda} = diag(\lambda_i) \in \Re^{l\times m}$.

2.3 Dynamic Components Selection

As mentioned in section 2.1, PCA based methods commonly choose the prior m' components (eigenvectors), corresponding to the maximum eigenvalues, to form the eigenmatrix [6,10]. The eigenvalue embodies the information amount of its corresponding component learnt from the training set. These components control different aspects of a face shape, such as gender, thin or fat, the size of cheekbone, etc (as shown in Fig.2). Due to the limitation of shape space, some components do not have strong correlativity to a novel face. In other words, an important component in training set may be useless to a test face. Contrarily, a component with less importance in training set may be useful to a new face. Therefore, the conventional scheme that uses the fixed eigenmatrix lacks generality in modeling. What's more, the regression theory indicates that the estimation precision decreases if too many useless components are included. Therefore, it would be better to select the most effective variables than the total model.

Base on this point, we propose a dynamic component deforming model (DCDM). DCDM uses t-test to determine the correlativity of each component to the novel face, then concatenates the most correlative ones to form the eigenmatrix. Let

$R = \left\| \boldsymbol{Q}_s^f \cdot \boldsymbol{\beta} - \Delta \boldsymbol{s}^f \right\|^2$ be the squared sum of reconstruction error of feature points, the freedom degree is $f_R = (l - m' - 1)$, the covariance matrix of $\boldsymbol{Q}_s^f$ is $\boldsymbol{C}_{\boldsymbol{Q}_s^f} = \left(\boldsymbol{Q}_s^f\right)^T \cdot \boldsymbol{Q}_s^f$. Then, the null hypothesis for the i-th components $\sigma_i \boldsymbol{q}_i^f$ is

$$H_{0,i} : \quad \beta_i = 0, \qquad 1 \le i \le m'. \tag{9}$$

According to t-test, $H_{0,i}$ is rejected if

$$W = \left\{ t_i > t_{\alpha/2}(f_R) \right\}, \tag{10}$$

where,

$$t_i = \left(|\beta_i| \cdot \sqrt{f_R} \right) \Big/ (R \cdot c_{i,i}), \tag{11}$$

α is the significance level, $c_{i,i}$ is the i-th element of the diagonal of $\left(\boldsymbol{C}_{\boldsymbol{Q}_s^f}\right)^{-1}$.

Accepting $H_{0,i}$ means that the i-th component has less relationship to current face, and should be removed from the eigenmatrix. In general, there are more than one components that are not correlative to a novel face. Because the deletion of a component may have influence to the t value of the others, we do not delete all the irrelevant components once but one by one—each time remove the one with the lest t_i and update $\boldsymbol{\beta}$. Given k feature points $\boldsymbol{s}^f$, the procedure of modeling a new face shape $\boldsymbol{s}_{new}$ by DCDM can be summarized as follows:

(a) Computing $\boldsymbol{Q}_s^f$ and $\Delta \boldsymbol{s}^f$ by Eq. (5) and Eq. (6), respectively;
(b) According to Eq. (8), getting initial value of $\boldsymbol{\beta}$;
(c) Computing t_i based on Eq. (11). If $\forall t_i > t_{\alpha/2}(f_R)$, $1 \le i \le m'$, go to step (e); else, remove the j-th (t_j is the lest) column from both $\boldsymbol{Q}_s$ and $\boldsymbol{Q}_s^f$, then go to step (d);
(d) Updating $\boldsymbol{\beta}$: $\beta_i^* = \beta_i - \frac{c_{j,i}}{c_{j,j}} \beta_j$, $1 \le i \le m'$, where $c_{j,i}$ is the element in the position $(j,\, i)$ of $\left(\boldsymbol{C}_{\boldsymbol{Q}_s^f}\right)^{-1}$, then $\boldsymbol{\beta}^* = (\beta_1^*, \cdots \beta_i^*, \cdots, \beta_{m'}^*)^T$, go to step (c);
(e) Computing displacements: $\Delta \boldsymbol{s} = \boldsymbol{Q}_s^* \cdot \boldsymbol{\beta}^*$, hence, $\boldsymbol{s}_{new} = \bar{\boldsymbol{s}} + \Delta \boldsymbol{s}$.

3 Experiments and Discussion

400 laser-scanned faces are randomly selected from the BJUT-3D Face Database, which are then split into a training set and a test set of m=200 faces each. The training set provides the exemplars that are learnt for linear model. From these, we compute m'=199 components which serve afterward as basic space. To quantify the modeling precision on 2D images, we synthesize a 'photo' database of the faces of test set by orthogonally projecting them to a 2D plane.

The tests of the algorithms (SRSD [12] and SDM [10] are adopted as counterparts which are related to our methods) are based on the following quantities, which are averaged across all the testing faces.

(a) Average Euclidean distance between vertices [12]:
$Euc(\boldsymbol{s}_r, \boldsymbol{s}_t) = \frac{1}{|V|} \sum_{i \in V} \|\boldsymbol{v}_{r,i} - \boldsymbol{v}_{t,i}\|$.

(b) Average derivative distance between edges [13]:
$Der(\boldsymbol{s}_r, \boldsymbol{s}_t) = \frac{1}{|E|} \sum_{i,j \in E} \left\|(\boldsymbol{v}_{r,i} - \boldsymbol{v}_{r,j}) - (\boldsymbol{v}_{t,i} - \boldsymbol{v}_{t,j})\right\|$.

(c) Average normal distance between triangles:
$Nor(\boldsymbol{s}_r, \boldsymbol{s}_t) = \frac{1}{|T|} \sum_{i \in T} \|\boldsymbol{n}_{r,i} - \boldsymbol{n}_{t,i}\|$.

These quantities evaluate a reconstruction through three aspects: vertex, curve, surface. Where, $\boldsymbol{s}_r$ and $\boldsymbol{s}_t$ are reconstructed surface and ground truth surface. V, E, T are sets of vertices, edges and triangles. $\boldsymbol{v}_{r,i}$ and $\boldsymbol{v}_{t,i}$ are corresponding vertices on $\boldsymbol{s}_r$ and $\boldsymbol{s}_t$. $\boldsymbol{n}_{r,i}$ and $\boldsymbol{n}_{t,i}$ are the normals of the triangles.

3.1 Evaluation on Test Set

First, we investigate how the component number affects fitting quality. In Fig. 4, we demonstrate the performance of SRSD and DCDM varying in components number, k=80. To model the training faces, as expected, SRSD could achieve the best reconstruction by all 199 components. However, it's not the case for novel faces that SRSD and DCDM achieve their best results at the point $m' \approx$50 as more components might lead to overfitting. So, unless otherwise noted, we adopt this empirical value m'=50 in the following tests.

Comparison of reconstruction precision on test set is illustrated in Fig. 5. Feature number k ranges from 18 to 300, m'=50. For SRSD and DCDM, the regularization η=0.00001 (more details of η see [12]), and α=0.05. It can be seen that the performance of SRSD and SDM declined quickly along with the decreasing of feature number. On the contrary, DCDM shows robustness to the change of feature number, especially in 2D cases. The inflexion at the point k=29 in 2D experiments is caused by feature location errors. Examples presented in Fig. 6 are two faces in the test set which are reconstructed by DCDM.

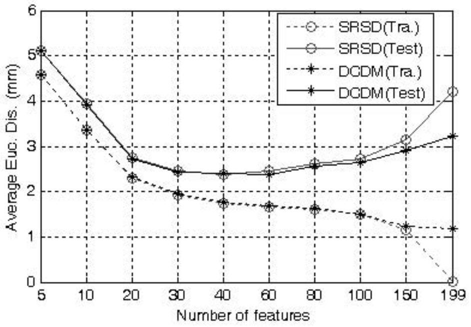


Fig. 4. The average 3D shape *Euc* error of SRSD and DCDM in modeling 200 faces, 100 from training set (denotes as 'Tra.') and 100 from novel set (denotes as 'Test')

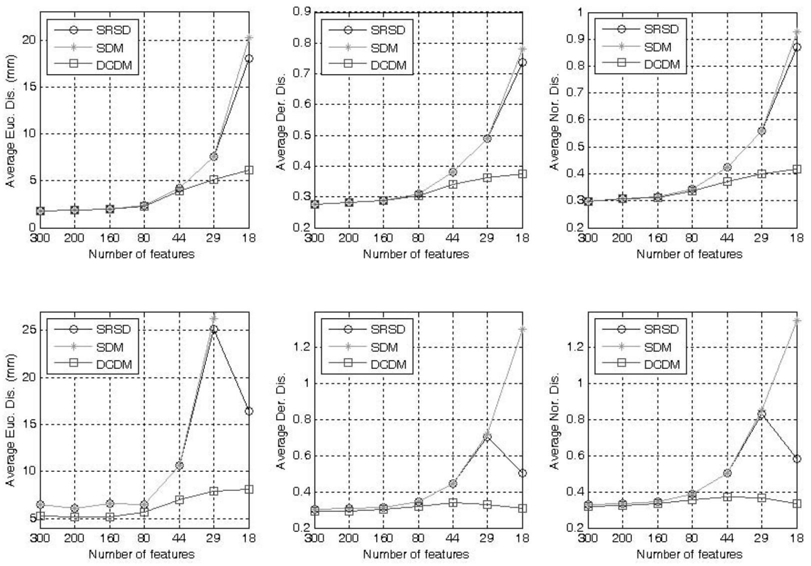


Fig. 5. Comparison of reconstruction errors. The tests in first row use 3D features while the second row use 2D features.

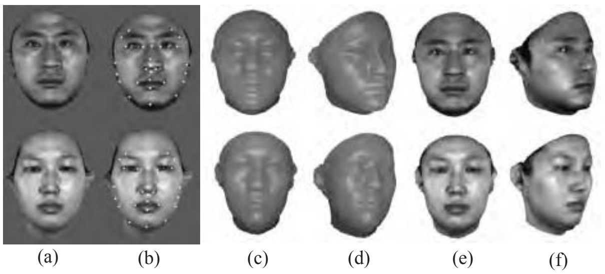


Fig. 6. Examples of reconstruction. (a) are the scanned faces; (b) illustrates the used 29 feature points; (c,d) are views of reconstructed shapes; (e,f) are textured results.

There are two possible factors that might enable DCDM achieve higher precision than SRSD. One is the , the other is the difference of components. To validate the selection strategy of DCDM, we first record the components' number (m') used in DCDM, then choose the prior m' components for SRSD. Thus, both DCDM and SRSD reconstruct faces by the same number of components; the only difference is which components are used. The comparison data provided in Table 1 have validated the effectiveness of DCDM.

Time costs listed in Table 2 indicate that DCDM could recover a face shape in 2 seconds. Although a bit slower than the other two methods, it could still meet the requirement of most current applications.

Table 1. Comparison of Euclidean distance errors (mm). SRSD and DCDM use the same number of components which are listed in the first row. The last row is the improvement of DCDM comparing SRSD. '3D' and '2D' denote the type of feature.

Method	k=18		k=29		k=44		k=80		k=300	
	3D	2D	3D	2D	3D	2D	3D	2D	3D	2D
Number	16	50	20	24	27	24	33	24	40	24
SRSD	8.876	16.398	5.928	11.332	4.161	9.811	2.336	7.655	1.908	6.203
DCDM	6.166	8.116	5.140	7.929	3.923	7.039	2.316	5.702	1.818	5.277
Imp.(%)	30.529	50.502	13.279	30.036	5.720	28.260	0.834	25.511	4.739	14.921

Table 2. Comparison of modeling time costs. 2D features are used for modeling, k=29, m'=50, (CPU: ADM2500+, RAM: 512M).

Methods	SRSD	SDM	DCDM
Time costs	0.83s	0.68s	1.49s

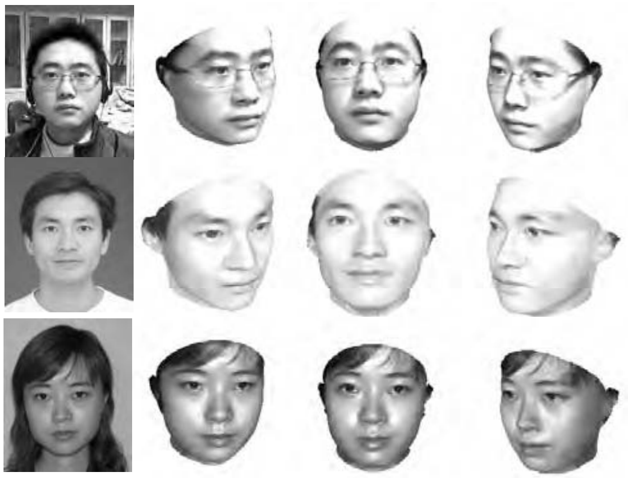


Fig. 7. Results of modeling in real photographs

3.2 Reconstruction on Real Images

In previous experiments, we have shown that the proposed method performs better than existing algorithms, especially only a few features are known. So we have good reason to believe that DCDM is appropriate to model from a 2D image based on features. In this experiment, we reconstruct 50 3D faces from real face pictures. Nine salient features (corners of eyebrows and eyes, nose tip, corners of mouth) are used to model the shape at first, and the image is then projected orthogonally to create the texture. Examples presented in Fig. 7 shows that the generated 3D faces are very realistic within a certain rotations.

4 Conclusion

A dynamic component deforming model is proposed to reconstruct the 3D face shape from a 2D image based on a set of feature points. In order to get plausible results, human faces is assumed belong to a linear class, principal components learned from a 3D face database are utilized as shape space. To model a novel shape, rather than using the fixed eigenvectors, the significance of each component is investigated at first, and then the most correlative ones are selected as the eigenmatrix. The main insight of our work is that, by appropriately selecting the correlative components, overfitting is alleviated and modeling results are enhanced. Experimental results show that this novel deforming model reduces the Euclidean distance between a reconstruction shape and its ground truth while preserving its smoothness and increasing its perceptual quality. Tests on real pictures have also achieved good results.

It's clear that a small set of feature points is not sufficient to recover a detailed surface, such as the shape of the nose and the ears. However, this technique could reliably estimate the overall shape and run quickly, which can be useful for such applications as face recognition, animation, etc. Since only a few points are needed, our method could also be applied to restore missing data of 3D surfaces.

In the future, we will forward to model face under variant poses and illuminations. We are also going to explore further 3D face applications based on this work, such as 3D faces recognition and animation.

Acknowledgement

Portions of the research in this paper use the BJUT-3D Face Database collected under the joint sponsor of National Natural Science Foundation of China, Beijing Natural Science Foundation Program, Beijing Science and Educational Committee Program. In particular, the authors wish to thank our colleagues for providing their portraits to present in this paper.

References

1. Lee, Y.C., Terzopoulos, D., Waters, K.: Realistic face modeling for animation. In: Siggraph Proceeding, Los Angeles, USA, pp. 55–62 (1995)
2. Zha, H.B., Wang, P.: Realistic face modeling by registration of 3D mesh models and multi-view color images. In: Proceeding of the 8th Int. Conf. Computer Aided Design and Computer Graphics, Macao, China, pp. 217–222 (2003)
3. Pighin, F., Hecker, J.: Synthesizing realistic facial expressions from photographs. In: Proceedings of the 25th Annual Conference on Computer Graphics, Orlando, USA, pp. 75–84 (1998)

4. Liu, Z.C., Zhang, Z.Y., Jacobs, C.: Rapid modeling of animated faces from video. The Journal of Visualization and Computer Animation 12, 227–240 (2001)
5. Dimitrijevic, M., Ilic, S., Fua, P.: Accurate face models from uncalibrated and ill-lit video sequence. In: Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, pp. 188–202. IEEE Computer Society Press, Washington, DC (2004)
6. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D-faces. In: SIGGRAPH 1999 Conference Proceedings, Los Angeles, USA, pp. 187–194 (1999)
7. Romdhani, S.: Face image analysis using a multiple feature fitting strategy. Ph.D. thesis, University of Basel, Switzerland (2005)
8. Hu, Y.L., Yin, B.C., Gu, C.L., Cheng, S.Q.: 3D Face reconstruction based on the improved morphable model. Chinese Journal of Computers 10, 1671–1679 (2005)
9. Wang, C.Z., Yin, B.C., Sun, Y.F., Hu, Y.L.: An improved 3D face modeling method based on morphable model. Acta Automatica Sinica 3, 232–239 (2007)
10. Chai, X.J., Shan, S.G., Qing, L.Y.: Pose and illumination invariant face recognition based on 3D face reconstruction. Journal of Software 3, 525–534 (2006)
11. Jiang, D.L., Hu, Y.X., Yan, S.C.: Efficient 3D reconstruction for face recognition. Pattern Recognition (6), 787–798 (2005)
12. Blanz, V., Vetter, T.: Reconstructing the complete 3D shape of faces from partial information. it+ti Oldenburg Verlag 6, 295–302 (2002)
13. Knothe, R., Romdhani, S., Vetter, T.: Combining PCA and LFA for surface reconstruction from a sparse set of control points. In: Proceeding of IEEE Conference on Automatic Face and Gesture Recognition, Southampton, UK, pp. 637–644. IEEE Computer Society Press, Los Alamitos (2006)
14. The BJUT-3D large-scale chinese face database. Technical Report No ISKL-TR-05-FMFR-001. Multimedia and Intelligent Software Technology Beijing Municipal Key Laboratory, Beijing University of Technology, 8 (2005)
15. Gong, X., Wang, G.Y.: Automatic 3D face segmentation based on facial feature extraction. In: Proceeding of IEEE Conference on Industrial Technology, Mumbai, India, pp. 1154–1159. IEEE Computer Society Press, Los Alamitos (2006)

Facial Feature Detection Under Various Illuminations

Jingying Chen and Bernard Tiddeman

School of Computer Science,
University of St Andrews, Fife, UK

Abstract. An efficient and robust method to locate eyes and lip corners under various illumination conditions is presented in this paper. First, a global illumination balance method is presented to compensate for variations in illumination and accentuate facial feature details. Next, a human face is distinguished from background based on Haar-like features. After that, eyes are found based on their intensity characteristics and Haar-like features. The lip region is estimated using the positions and sizes of face and eyes. Then, a novel local illumination balance technique, based on an integral projection analysis, is proposed to correct for the non-uniform illumination in the lip region. Finally, lip corners are detected based on their intensity and geometric characteristics. Encouraging results have been obtained using the proposed method.

1 Introduction

Automatic facial feature detection is important in many computer vision applications such as human machine interaction [1], facial expression analysis [2] and facial image transformation [3]. In Chen and Tiddeman's work, they are developing a robust real-time stereo tracking system, with automatic initialization of feature points and recovery from temporary occlusion [4]. These applications need to detect the facial features robustly and efficiently. However, the unknown and variable illumination conditions make the detection task difficult. Hence, we correct for variations in illumination using a mixture of global and local illumination balance techniques.

The facial feature detection literature includes image-based approaches [5, 6], template-based approaches [7, 8] and appearance-based approaches [9, 10]. Image-based approaches use color information, properties of facial features and their geometric relationships to locate facial features. Stiefelhagen et al. [5] used color information and certain geometric constraints on the face to detect six facial feature points (i.e. pupils, nostrils and lip corners) in real time for lip reading. This method works properly under good lighting conditions, however the lip corners may drift away when the illumination is not uniform (e.g. half the face is bright and the other half is dark, see Fig. 1). Hsu et al. [6] proposed a lighting compensation and color transformation methods to detect facial features. Their lighting compensation method considers the pixels with the top 5% of luma (nonlinear gamma-corrected luminance) values in the image as the "reference white" only if the number of those pixels is sufficiently large (>100). They assume the dominant bias color always appears as "real white", however, for images with a low illumination background (e.g. the human face is the brightest part), their method may fail. Furthermore, this lighting compensation method does not improve the lip corner detection when the illumination is not uniform. Template based approaches are usually applied to intensity images where a predefined template of facial feature is matched against image blocks. Kapoor and Picard [7] used eyebrow and eye templates to locate upper facial features in a real time system. However, specialized hardware (an infrared sensitive camera equipped with infrared LEDs) is needed to produce the red eye effect in order to track the pupils. Matsumoto and Zelinsky [8] detected the facial features using an eye and mouth template matching method, which was implemented using the IP5000 image processing board. Both of their methods are preformed under good lighting conditions. Appearance-based approaches use facial models derived from a large amount of training data. These methods are designed to accommodate possible variations of human faces under difference conditions. Cootes et al. [9] proposed active appearance models (AAM) and Matthews and Baker [10] improved the performance of the original AAM. However, these methods need large amounts of delineated training data and involve relatively expensive computations.

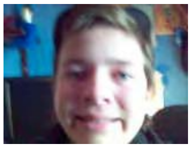


Fig. 1. An example of non-uniform illumination on the face

In this paper, an efficient and robust feature detection method is proposed for locating eyes and lip corners under various illumination conditions. A mixture of global and local illumination balance methods is proposed to compensate for the variations in illumination and to accentuate facial feature details. The eyes are found based on their intensity characteristics (e.g. pixels around eyes are more varied in value than other parts of face) and Haar-like features. Lip corners are found based on their intensity and geometric characteristics, e.g. lip corners are extremities of dark regions. The proposed method has been tested using four face image databases including different skin color under various illumination conditions with encouraging results.

The outline of the paper is as follows. Section 2 presents the global illumination balance technique. The face and facial feature (i.e. eyes and lip corners) detection are given in Section 3. Section 4 describes the experimental results while Section 5 presents the conclusions.

2 Global Illumination Balance

Global illumination accounts for the overall intensity level in an image. Poor illumination is usually caused by insufficient or strong lighting. Here, a global illumination balance technique is proposed to alleviate the illumination effect on facial features detection and tracking under various illumination conditions. Savvides et al. [11] applied a non-linear logarithm transformation to normalize the illumination, which extends low grey level and compresses high grey level. This transformation can improve the illumination deficient greatly. Liu et al. [12] integrated a high pass filter before the logarithm transformation which preserves edge or high frequency features. Their transformation methods are useful for dark images, however, it is not effective for bright images. Hence, we improve the logarithm transformation by extending low and high grey levels while compressing mid-tone. This is because the facial features used in this work have obvious illumination characteristics, e.g. the pupils are dark blobs inside white regions and the lip corners are the extremities of dark region. The modified logarithm transformation is given below:

$$\left\{\begin{array}{ll} g(x,y) = a + \dfrac{\log(f(x,y)+1)}{b} & f(x,y) < T \\ g(x,y) = a + \dfrac{\log((T+1)^2/(2T+1-f(x,y)))}{b} & f(x,y) > T \end{array}\right\}$$

where *f(x, y)* and *g(x, y)* are the grey level of the original and the logarithm transformed images respectively. *a* and *b* changes the shape and position of the curve. *T* is the average grey value of the original image. An example of the logarithmic mapping curve is shown in Fig. 2.

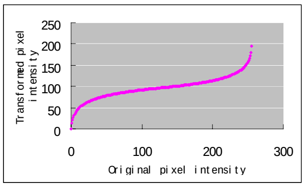


Fig. 2. The proposed logarithmic transformation of pixel intensities

Fig.3 shows illumination balance examples using the original logarithm transformation and the proposed method. From the figures, one can see that the proposed method balances the illumination reasonably and accentuates facial feature details.

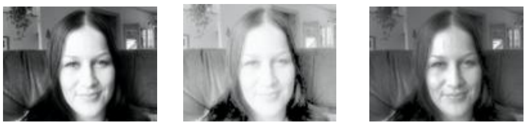

(a) (b) (c)

Fig. 3. Examples of the global illuminati on balance, (a) the original image, (b) the original log-transformed image and (c) the proposed method transformed image

3 Facial Feature Detection

3.1 Face Detection

After the global illumination is balanced, a human face is distinguished from the background and then facial feature detection is performed based on the detected face.

Viola and Jones's [13] face detection algorithm based on Haar-like features is used here. Haar-like features encode the existence of oriented contrast between regions in the image. A set of these features can be used to encode the contrast exhibited by a human face and their special relationships. In Viola and Jones's method, a classifier (i.e. a cascade of boosted classifier working with Haar-like features) is trained with a few hundreds of sample views of face and non-face examples, they are scaled to the same size, i.e.24x24. After the classifier is trained, it can be applied to a region of interest in an input image. To search for the face, one can move the search window across the image and check every location using the classifier.

After the face is detected, Principal Component Analysis (PCA) [14] is employed within face skin region for estimation of the face direction. In 2D shape, PCA can be used to detect principal directions of the spatial shape. Since faces are approximately symmetric and there are many features around eyes, the first principal axis indicates the upright direction of the face while the second principal axis gives the direction of eyes (i.e. the line connecting the two eyes). Fig. 4 shows the face edge map and the principal directions of the edge map.

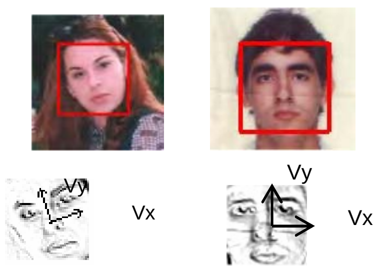


Fig. 4. Principal axes of face skin region. The major axis Vy represents face upright direction while the minor axis Vx represents eye direction.

After obtaining the eye direction, one can rotate the face until the eye direction is parallel to the horizontal axis using the middle point of the line connecting the two eyes as the rotation centre, which facilitates the following eyes and lip corners detection.

3.2 Eye Detection

3.2.1 Eye Region Localization. After the face is detected and aligned, the eye region can be located based on the face vertical edge map. Pixel intensities change in vertical direction more near the eyes than the other parts of face (e.g. the eyebrows and the side of face boundary). Hence it is reasonable to use a vertical edge map to decrease false positive eye detection. First, a horizontal integral projection [15] is used on the upper face vertical edge map to estimate the vertical position of eyes. It is obvious that the vertical position of the eyes is the global maximum of the horizontal integral projection in the upper face. According to the vertical position and the height of face, we can estimate the vertical eye region (See Fig. 5).

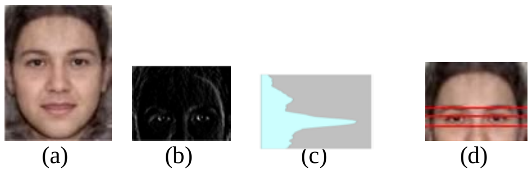


Fig. 5. Vertical eye region detection, (a) face image, (b) upper face vertical edge map, (c) horizontal projection performed on image(b),and (d) estimated vertical eye region

Second, a vertical projection is performed on the face vertical edge map to estimate the right and left boundary of face which correspond the two peaks of projection values (See Fig. 6).

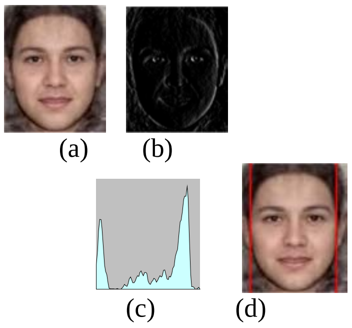


Fig. 6. Face boundary detection, (a) face image, (b) face vertical edge map, (c) vertical projection performed on image (b), and (d) estimated right and left face boundary

Finally, the eye region can be located based on the vertical eye region and right and left face boundary (see Fig. 7).

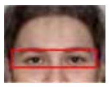


Fig. 7. Estimated eye region

3.2.2 Eye Detection. In order to improve the accuracy and efficiency of detection, eyes are searched for within the obtained eye region instead of the entire face, which decreases false positive and speeds up the detection process. Similar to face detection described above, eyes are found using a cascade of boosted tree classifiers with Haar-like features. A statistical model of the eyes is trained in this work. The model is made of a cascade of boosted tree classifiers. The cascade is trained on 1000 eye and 3000 non-eye samples of size 18x12. The 18x12 window moves across the eye region and each sub-region is classified as eye or non-eye. An example of the eye detection is shown in Fig. 8.

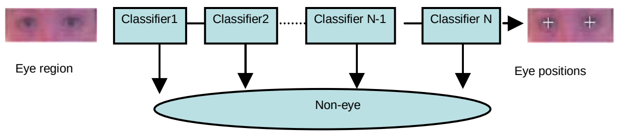


Fig. 8. Example of the eye detection using a cascade of boosted tree classifiers

3.3 Lip Corner Detection

3.3.1 Local Illumination Balance. Non-uniform illumination, as shown in Fig. 1, makes the lip corner detection difficult. Hence, a local illumination balance approach based on the vertical integral projection analysis is proposed in this work. First, a vertical integral projection is performed on the intensity image in the lip region (see Fig. 9). Lip region are found based on the face size and eye positions. From the figures, one can see that the projection curve is almost symmetric according to the facial symmetric axis under good illumination conditions (Fig. 9 (a)). However, the curve is asymmetric when the illumination is not uniform (e.g. Fig. 9 (b) the right part of the lip region is brighter than the left part). Asymmetry can be detected by comparing two mean values of the vertical integral projection value, P_v, between the left and right parts (according to the facial symmetry axis) of the images. If the difference of the two mean values is greater than a predefined threshold, the asymmetry is detected. Second, an intensity transformation is performed to balance the illumination. The curve, P_v, in Fig. 9 (b) is rotated clockwise using the middle point as the rotation center until the difference between the two mean values is less than a threshold (see Fig. 9 (c)). Then, we adjust the image pixel value using the equations below:

$$BalancedPixelValue = OrigianlPixelValue + \Delta T$$
$$\Delta T = (P_{v\_rotated} - P_{v\_original}) / imageheight$$

For each column of the image, the difference between the original P_v and the rotated P_v is distributed evenly into every pixel. From the upper image of the Fig. 9 (c), one can see that the illumination is balanced.

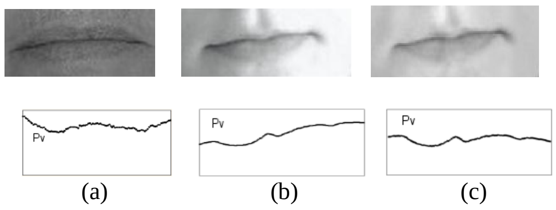


Fig. 9. Illustration of vertical integral projection curves under different illumination. The upper images are lip regions, the lower images plot the vertical integral projection values, P_v, against corresponding columns, (a) uniform illumination, (b) non-uniform illumination and (c) corrected illumination.

3.3.2 Lip Corner Detection. After the lip region has been illumination-balanced lip corner detection is performed. First, a horizontal integral projection is used on the intensity image in the lip region to find the vertical position of the shadow line between the lips. The shadow line is the darkest horizontally extended structure in the lip region, its vertical position can be found where the horizontal integral projection value, P_h, is the global minimum (see Fig. 10). The position can be considered as the approximate vertical position of the lip corners.

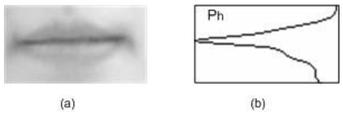


Fig. 10. Finding the vertical position of the shadow line between the lips using the horizontal integral projection in the lip region, (a) lip region and (b) horizontal integral projection value P_h plotted against corresponding rows

Second, a vertical integral projection is performed on the horizontal edge map in the lip region. The horizontal edge map can be obtained by applying the Sobel horizontal edge detector. The horizontal positions of the lip corners are estimated by examining the vertical integral projection values, P_v, the locations where the values exceed or fall below a certain predefined threshold are considered as the estimated horizontal positions of the lip corners (Fig. 11).

Finally, the positions of the lip corners are refined. We search for the extremities of the dark areas around the positions (found above) of the left and right corners in the two small search windows (Fig. 12). In the search windows the darkest 5% of pixels are selected and the right and left-most dark pixels are considered as the lip corners.

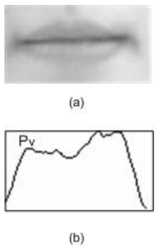


Fig. 11. Finding the horizontal position of the lip corners using the vertical integral projection on the horizontal edge map of the lip region, (a) lip region and (b) vertical integral projection value, P_v, plotted against corresponding columns

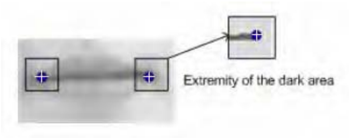


Fig. 12. Refining the lip corners in the two search window

4 Experimental Results

The proposed method has been implemented using C++ under MS windows environment and tested with four databases that include different skin color under various illumination conditions. The current implementation runs in real time on a PC with Pentium M 715 processor. The first database includes 181 face images with standardized pose, lighting and camera setup. The second database includes 121 unstandardized mobile phone face images with different lighting, pose and face expression. The third database is an unstandardized set of face images from the internet containing 200 face images including different skin color, various illumination and face pose and expression. The final database contains 100 face images in stereo pairs, representing the texture maps of 3D face models. The detection rates (i.e. images with correctly detected eyes or lip corners relative to the whole set of the four database images) of the eyes and lip corners are 96.5% and 95.3%. The detected facial features using the proposed method are shown below. The white crosses represent the detected features.

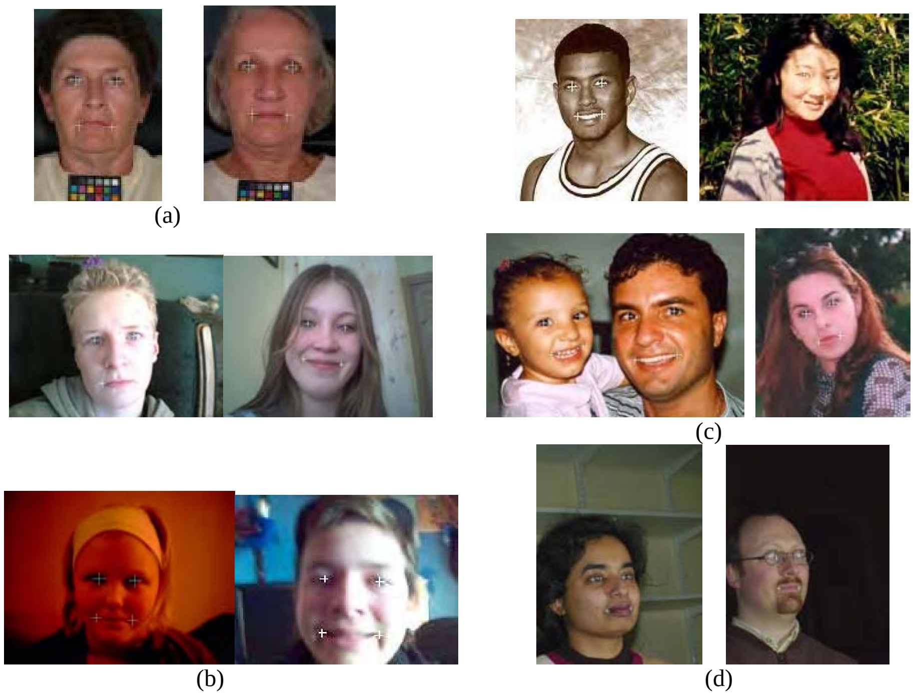


Fig. 13. The results of detected eyes and lip corners from four face image databases. (a) The database of highly standardised photos, (b) the database of unstandardized mobile phone image, (c) unstandardize images from the internet and (d) rotated face images from stereo image pairs.

From these results, one can see that the proposed approach can detect the eyes and lip corners accurately under various illumination conditions. However, false detection could exist under some circumstances (see Fig. 14). One can notice that the eyebrows and eyes are quite similar from Fig. 14 (a), and the moustache occludes the lips from Fig.14 (b). The measurements of the detection accuracy of the proposed method were taken using 100 images from the databases (i.e. 25 images from each database) in this experiment. Manually determined positions of the feature points were used as reference for measuring displacement error. The absolute error in pixels from the known location of each point was divided by the true eye separation in order to control for different size images. This provided a percentage error in the x and y directions, and the average of these errors are shown in Table 1.

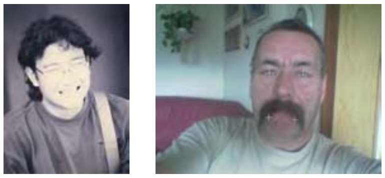


Fig. 14. False detection on eyes (a) and lip corners (b)

Table 1. Average feature point displacement errors in x and y directions in percentage of the known eye separation

	Error x	Error y
Eyes	2.9%	3.1%
Lip corners	4.1%	4.2%

Fig.15. gives displacement error computed as the Euclidean distance between the reference points and the points obtained by the proposed method divided by the known eye separation. The highest point in Fig. 15(a) corresponds to the false eye detection as shown in Fig. 14(a).

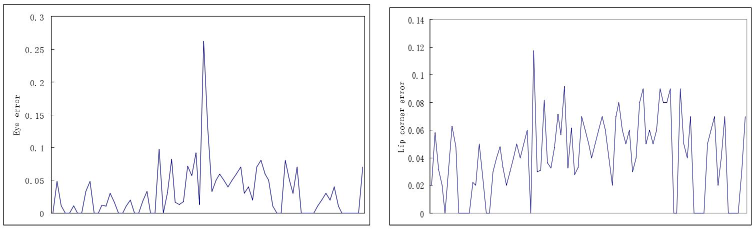


Fig. 15. The error computed as the Euclidean distance (between the reference point and the detected point) divided by the known eye separation, (a) eye error and (b) lip corner error

Compared with other facial feature detection methods [5, 6], the proposed approach is capable of detecting features under various illumination conditions, even under non-uniform illumination. The approach is efficient and accurate, also, it is easy to implement.

5 Conclusions

In this paper, the global and local illumination balance techniques are proposed to facilitate the eyes and lip corners detection under various illumination conditions. The eyes are found based on their intensity characteristics and Haar-like features. Lip corners are found based on their intensity and geometric characteristics. The proposed method has been tested using four face image databases including different skin color under various illumination conditions. The obtained results suggest a strong potential alternative for building a robust and efficient facial feature detection system.

References

[1] Barreto, J., Menezes, P., Dias, J.: Human-robot interaction based on haar-like features and eigenfaces. In: Proceedings of the International Conference on Robotics and Automation, New Orleans, pp. 1888–1893 (2004)

[2] Fasel, B., Luettin, J.: Automatic facial expression analysis: a survey. Pattern Recognition 36(1), 259–275 (2003)

[3] Tiddeman, B., Perrett, D.: Moving facial image transformations using static 2D prototypes. In: WSCG 2001. Proceedings of the 9-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2001, Plzen, Czech Republic, 5-9th February (2001)

[4] Chen, J., Tiddeman, B.: A real time stereo head pose tracking system. In: 5th IEEE International Symposium on Signal Processing and Information Technology, Greece, December 18-21 (2005)

[5] Stiefelhagen, R., Meier, U., Yang, J.: Real-time lip-tracking for lipreading, Proceedings of the Eurospeech 1997, 5th European Conference on Speech Communication and Technology, Rhodos, Greece (1997)

[6] Hsu, R.L., Abdel-Mottaleb, M., Jain, A.K.: Face detection in color image. IEEE transactions on Pattern Analysis and Machine Intelligence 24(5), 696–706 (2002)

[7] Kapoor, A., Picard, R.W.: Real-time, fully automatic upper facial feature tracking. In: Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition, Washington DC, USA, pp. 10–15 (May 2002)

[8] Matsumoto, Y., Zelinsky, A.: An algorithm for real time stereo vision implementation of head pose and gaze direction measurement. In: Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition, France, pp. 499–505 (2000)

[9] Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(6), 681–685 (2001)

[10] Matthews, I., Baker, S.: Active appearance models revisited, Technical report: CMU-RI-TR-03-02, the Robotics Institute Carnegie Mellon University (2002)

[11] Savvides, M., Vijaya Kumar, B.V.K.: Illumination normalization using logarithm transforms for face authentication. In: Proceedings of International Conference on Audio and Visual Biometrics based Person Authentication (AVBPA), Surrey, UK, pp. 539–556 (2003)

[12] Liu, H., Gao, W., Miao, J., Li, J.: A novel method to compensate variety of illumination in face detection. In: Proceedings of the 6th Joint Conference on Information Sciences, North Carolina, pp. 692–695 (2002)

[13] Viola, P., Jones, M.: Robust real time object detection. In: 2nd International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing and Sampling, Vancouver, Canada (July 2001)
[14] Feng, G., Yuen, P.: Multi-cues eye detection on grey intensity image. Pattern Recognition 34, 1033–1046 (2001)
[15] Kanade, T.: Picture processing by computer complex and recognition of human faces, Technical report, Kyoto University (1973)

Comparing a Transferable Belief Model Capable of Recognizing Facial Expressions with the Latest Human Data

Zakia Hammal, Martin Arguin, and Frédéric Gosselin

Département de psychologie, Université de Montréal, Canada
zakia_hammal@yahoo.fr, martin.arguin@umontreal.ca,
frederic.gosselin@umontreal.ca

Abstract. Despite significant amount of research on automatic classification of facial expressions, recognizing a facial expression remains a complex task to be achieved by a computer vision system. Our approach is based on a close look at the mechanisms of the human visual system, the best automatic facial expression recognition system yet. The proposed model is made for the classification of the six basic facial expressions plus *Neutral* on static frames based on the permanent facial features deformations using the Transferable Belief Model. The aim of the proposed work is to understand how the model behaves in the same experimental conditions as the human observer, to compare their results and to identify the missing informations so as to enhance the model performances. To do this we have given our TBM based model the ability to deal with partially occluded stimuli and have compared the behavior of this model with that of humans in a recent experiment, in which human participants had to classify the studied expressions that were randomly sampled using Gaussian apertures. Simulations show first the suitability of the TBM to deal with partially occluded facial parts and its ability to optimize the available information to take the best possible decision. Second they show the similarities of the human and model observers performances. Finally, we reveal important differences between the use of facial information in the human and model observers, which open promising perspectives for future developments of automatic systems.

Keyword: Facial Feature Behavior, Facial Expressions Classification, Transferable Belief Model, Bubbles.

1 Introduction

Facial expressions reveal something about the emotional status of a person and are important channel for nonverbal communication. Facial expressions communicate information from which we can quickly infer the state of mind of our peers, and adjust our behavior accordingly [1]. They are typically arranged into six universally recognized basic categories *Happy, Surprise, Disgust, Fear, Sadness* and *Anger* that are similarly expressed across different backgrounds and cultures [2], [3]. Despite significant amount of research on automatic classification of facial expressions [4], [5], [6], [7], [8], recognizing a facial expression remains a complex task to be achieved by a computer vision system. Notably the enhancement of the classifications performances is not only due to the increase of the used visual cues available in the human face but on their optimal use. The best recognition system is the human visual system. It uses cues that are robust and non-accidental in real-life situations. It should therefore inform our efforts to build automatic facial expressions classifier. We thus turned to the psychological literature to find the most informative data about how humans recognize facial expressions.

Smith *et al* [9] examined the recognition of the six so-called "basic" facial expressions as well as *Neutral* using *Bubbles*, a psychophysical procedure that prunes stimuli in the complex search spaces characteristic of visual categorization to reveal their effective information for driving a given behavioral response during a recognition task [10]. Here we model the Smith *et al* experiment adapting a previously developed model for the classification of static stimuli displaying the six basic facial expressions plus *Neutral*. Its is based on the comparison of the permanent facial features (eyes, eyebrows and mouth) deformations to their neutral state using the Transferable Belief Model (TBM). We thus show how the TBM can be adapted to sparse stimuli encountered in a *Bubbles* experiment and how it can be compared with the human participants of Smith *et al*. The comparison between the modified TBM model and humans reveal important differences that should lead to major improvement in future implementations.

2 Bubbles Experiment

Developed by Gosselin and Schyns [10] *Bubbles* is a technique capable of revealing the stimulus information that drives any measurable responses. In [9] the authors applied the *Bubbles* technique to identify the information underlying the recognition of the six basic facial expressions plus *Neutral* (from the Dailey-Cottrell database[1]), each one displaying the six basic facial expressions and *Neutral*. Fourteen participants were each submitted to 8400 sparse facial expression stimuli and were instructed to identify which one of the seven studied facial expression was displayed. The stimuli were produced by randomly sampling grayscale facial expression images at five scales using scale-adjusted Gaussian filters (see Fig. 1 a). The total number of Gaussian apertures was modified on each trial, independently for each expression, to maintain 75% of correct categorization (see [9] for details). The experiment revealed the precise effective filters for the categorization of the basic six expressions plus *Neutral* (see Fig. 1 b).

To benchmark the information available for performing the task, and therefore to be able to rank human use of information for each expression and scale, a model observer was also built. The model was submitted to the same experiment as the human observers. For each trial, the model determined the Pearson

[1] CAFE database, http://www.cs.ucsd.edu/users/gary/CAFE

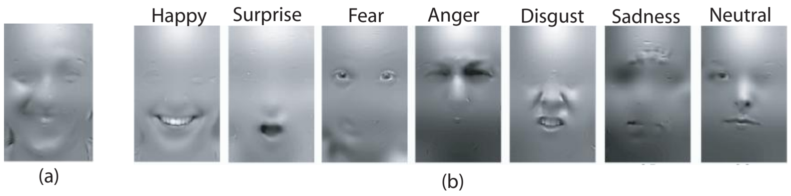


Fig. 1. (a): Example of stimulus, (b): Effective faces representing diagnostic filtering functions for the human observers [9]

correlation between a sparse input and each of the 70 possible original images revealed with the same *Bubbles* apertures. Its categorization response was the category of the original image with the highest correlation to the sparse input.

The experiment revealed the precise effective filters for the categorization of the basic six expressions plus *Neutral* (see Fig. 1 b).

To deal with the sparse stimuli used in Smith *et al* experiment, we adapted the TBM based model proposed in [8].

3 Facial Expression Modeling

In order to recognize the six basic facial expressions plus *Neutral* and *Unknown* expressions (none of the six facial expressions), the contours of the permanent facial features (eyes, eyebrows and mouth) are extracted from each frontal face image (see Fig. 2 a). Based on the segmentation results, the permanent facial features deformations occurring during facial expressions according to the Neutral state are measured by five characteristic distances D_1, D_2, D_3, D_4 and D_5 (see Fig. 2 a) [8]. A numerical to symbolic conversion is carried out using a fuzzy-like model for each characteristic distance D_i (see [8] for more detailed description) and allows to convert each numerical value to a belief of five symbolic states according to how is different its value from the corresponding value in the *Neutral* state. S_i if the current distance is roughly equal to its corresponding value in the *Neutral* expression, C_i^+ (resp.C_i^-) if the current distance is significantly higher (resp. lower) than its corresponding value in the *Neutral* expression and $S_i \cup C_i^+$ [2] (resp. $S_i \cup C_i^-$) if the current distance is neither sufficiently higher (resp. lower) to be in C_i^+ (resp. C_i^-), nor sufficiently stable to be in S_i. In order to determine the expression corresponding to the knowledge brought by each distance state, a fusion process of the belief of the characteristic distances states is then made based on the TBM [11]. Based on this modeling process the aim of this study is to go further and to measure the different contributions of the facial features for the facial expressions classification. We adapted the TBM based model to make it robust to occlusions such as the ones occurring during the Smith *et al Bubbles* experiment.

[2] The sign $\cup$ means logical OR.

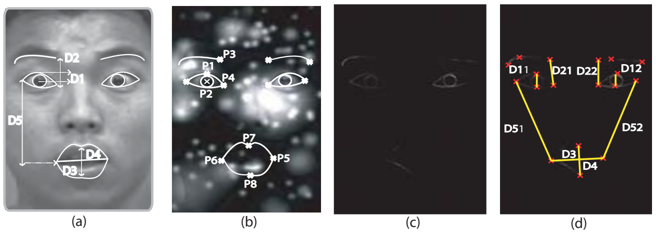


Fig. 2. (a): Facial features segmentation and the corresponding five characteristic distances, (b): superposition of the segmentation results of the facial features contours on the frame (a), (c): revealed facial features contours, (d): identified characteristic points and their corresponding characteristic distances

4 The Transferable Belief Model Applied to the *Bubbles* Experiment

The TBM is a model of representation of partial knowledge [12]. The facial expression classification is based on the TBM fusion process of all the information resulting from the characteristic distances states. The first step consists in defining the Basic Belief Assignment BBA $m_{D_i}^{\Omega_{D_i}}$ of each characteristic distance states D_i as:

$$m_{D_i}^{\Omega_{D_i}} : 2^{\Omega_{D_i}} \to [0,1] A \to m_{D_i}^{\Omega_{D_i}}(A), \sum_{A \in 2^{\Omega_{D_i}}} m_{D_i}^{\Omega_{D_i}}(A) = 1 \tag{1}$$

where $2^{\Omega_{D_i}} = \{\{C_i^+\}, \{C_i^-\}, \{S_i\}, \{S_i, C_i^+\}, \{S_i, C_i^-\}, \{S_i, C_i^+, C_i^-\}\}$ is the frame of discernment, $\{S_i, C_i^+\}$ (resp. $\{S_i, C_i^-\}$) the doubt state between C_i^+ (resp. C_i^-) and S_i. $m_{D_i}^{\Omega_{D_i}}(A)$ is the belief in the proposition $A \in 2^{\Omega_{D_i}}$ without favoring any of propositions of A. Total ignorance is represented by $m_{D_i}^{\Omega_{D_i}}(\Omega_{D_i}) = 1$. To simplify $\{C_i^+\}$ is noted C^+ and $\{S_i, C_i^+\}$ is noted $S \cup C^+$.

4.1 *Bubbles* Modeling Process

Distance measurements. The first step is to identify among the five characteristic distances only those retrievable from the facial features revealed by the Gaussian apertures on each trial. First the permanent facial features are segmented from the facial expression (see Fig. 2 a). Second the Gaussian apertures use to sample the facial expression are applied to the segmented permanent features (see Fig. 2 b). The intersection between the contours and the results of the application of the *Bubble's* mask is done revealing the contours of the permanent facial features (see Fig. 2 c). The intensity of the contours depends on the size, the position and the number of Gaussian apertures (see Sect. 2). Finally based on these contours the characteristic points for which the pixels intensities are different from

0 are identified (crosses in Fig. 2 d). All distances computed from contour points different from 0 are taken into account in the classification process.

If we assume that facial expressions are symmetrical each distance is considered as the mean between its corresponding left and right side. $D_i = (D_{i1} + D_{i2})/2$ if D_{i1} and D_{i2} are revealed or D_{i1} (resp. D_{i2}) if D_{i2} (resp. D_{i1}) is not revealed (1 ≤ i ≤ 5) (see Fig. 2 d). Once the intersection step is completed, the BBAs of the identified characteristic distances are computed (see Sect. 4). In order to model the stimuli used by human observer, the appearance intensities of the corresponding characteristic distances are taken into account in the fusion process.

Discounting. *Discounting* [11] was used to weaken or to inhibit the characteristic distances used for the classification process by weighting them by their intensity following the sampling by the Gaussian apertures (see Fig. 2 c). More specifically, it consisted in defining a parameter $\alpha \in [0, 1]$ which allows to compute the new piece of evidence ${}^{\alpha}m$ for each proposition A_i according to its current piece of evidence as:

$$ {}^{\alpha}m(A_i) = \alpha.m(A_i); {}^{\alpha}m(A_i \cup \overline{A_i}) = 1 - \alpha.(1 - m(A_i \cup \overline{A_i})) \tag{2} $$

where $\overline{A_i}$ corresponds to the complement of A_i.

Discounting was used to weight the contribution of each characteristic distance D_i according to its intensity noted $inten(D_i)$. This leads to five *discounting* parameters α_i one for each characteristic distance D_i. Each parameter α_i was computed on line proportional to $inten(D_i)$ as:

$$ \alpha_i = inten(D_i) \tag{3} $$

Each D_i is computed by measuring the distance between two points relatively to their distance in the neutral state. Thus its intensity corresponds to the mean intensities of its associated characteristic points. For example α_1 the *discounting* parameter of D_1 was computed as:

$$ \alpha_1 = inten(D_1) = (inten(P1) + inten(P2))/2 \tag{4} $$

where $inten(P1)$ and $inten(P2))$ correspond, respectively, to the intensities of pixels $P1$ and $P2$ (see see Fig. 2 b).

To evaluate the influence of the intensities of the characteristic distances we also consider the special case where the *discounting* parameters of all used distances ($D_i \neq 0$) were equal to 1, that is:

$$ \alpha_i = 1 (1 \leq i \leq 5). \tag{5} $$

Once the *discounting* parameters α_i of all used characteristic distances are set, the corresponding BBAs are redefined according to equation 2.

4.2 Fusion Process

Each expression is characterized by a combination of a set of characteristic distances states according to the rules displayed in Table 1 (see [8] for details).

Table 1. D_i states corresponding to each facial expression

	D_1	D_2	D_3	D_4	D_5
$HappyE_1$	C^-	$S \cup C^-$	C^+	C^+	C^-
$SurpriseE_2$	C^+	C^+	C^-	C^+	C^+
$DisgustE_3$	C^-	C^-	$S \cup C^+$	C^+	$S \cup C^-$
$AngerE_4$	$C-$	C^-	S	$S \cup C^-$	S
$SadnessE_5$	C^-	C^+	S	C^+	S
$FearE_6$	C^+	$S \cup C^+$	$S \cup C^-$	$S \cup C^+$	$S \cup C^+$
$NeutralE_7$	S	S	S	S	S

From these rules and the BBAs of the states of the characteristic distances $m_{D_i}^{\Omega_{D_i}}$, a set of BBAs on facial expressions $m_{D_i}^{\Omega}$ is derived to each one of the characteristic distance D_i. Ω = { *Happy* (E_1), *Surprise* (E_2), *Disgust* (E_3), *Fear* (E_4), *Anger* (E_5), *Sadness* (E_6), *Neutral* (E_7) } the set of expressions. In order to combine all this available information, a fusion process of the BBAs $m_{D_i}^{\Omega}$ of all the states of the characteristic distances is performed using the conjunctive combination rule. For example, considering two characteristic distances D_i and D_j with two BBAs $m_{D_i}^{\Omega}$ and $m_{D_j}^{\Omega}$, the joint BBA $m_{D_{ij}}$ is given using the conjunctive combination as:

$$m_{D_{ij}}^{\Omega}(A) = (m_{D_i}^{\Omega} \oplus m_{D_j}^{\Omega})(A) = \sum_{B \cap C = A} m_{D_i}^{\Omega}(B) m_{D_j}^{\Omega}(C) \tag{6}$$

where A, B and C denote propositions and $B \cap C$ denotes the conjunction (intersection) between the propositions B and C. This leads to propositions with a lower number of elements than previously and with a more accurate piece of evidence.

4.3 Decision Process

To compare directly our classification results with the human participants of Smith *et al* [9], the decision was made using the pignistic probability $BetP$ [13], which only deals with singleton expressions:

$$BetP : \Omega \rightarrow [0,1]; C \rightarrow BetP(C) = \sum_{A \subseteq \Omega, C \in A} \frac{m^{\Omega}(A)}{(1 - m^{\Omega}(\emptyset)).Card(A)}, \forall C \in \Omega \tag{7}$$

5 Simulation Results

The simulation results were obtained on all basic facial expressions stimuli employed by Smith *et al* [9] that is, 7200 stimuli per subject*14 subjects. Three simulations were carried out: first using all the characteristic distances as a reference to analyze the behavior of the system to the inhibition or weighting of the characteristic distances; second with the *discounting* by inhibition of the characteristic distances; and, finally with the *discounting* according to the characteristic distances intensities.

5.1 Results with All Characteristic Distances

For this simulation all characteristic distances were used. Three kinds of results are reported on the Fig. 3. First, the singleton results which are associated with a high level of confidence and where only one expression is recognized; second, the double results where two expressions were recognized at the same time (this occurs when the model hesitates between two expressions); third, the Total results which is the sum of the singleton results and of the corresponding double results.

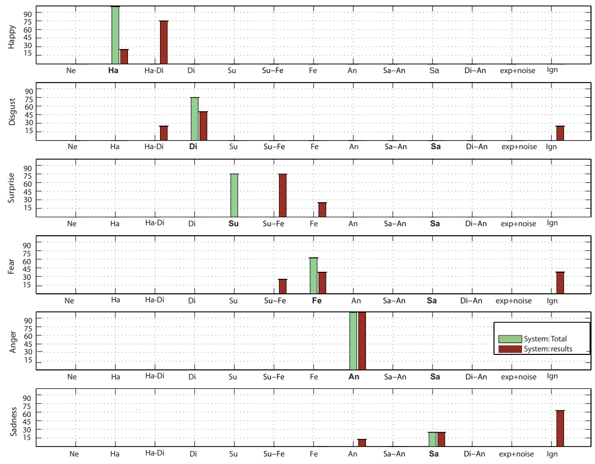


Fig. 3. Classification results with all the characteristic distances. Ne: *Neutral*, Ha: *Happy*, Di: *Disgust*, Su: *Surprise*, Fe: *Fear*, An: *Anger*, Ign: *Total ignorance*.

One of the most important characteristics of the TBM is that it allows to model doubt. The classification results show that two double results occur frequently. The doubt between the *Happy* and *Disgust* facial expressions appears to be due to the double states of the Table 1. Another pair of expressions between which the system has difficulty to choose are *Surprise* and *Fear*. Interestingly, these expressions are also notoriously difficult to be distinguished for human observers. Fig. 4 shows an example for which the system remains doubtful instead of taking the risk of making a wrong decision. Based on the pignistic probability decision, such a doubt is transformed into a double result with the same pignistic probability. Considering the doubt states we obtain the final results doing the sum of the singleton and the corresponding double results leading to the Total

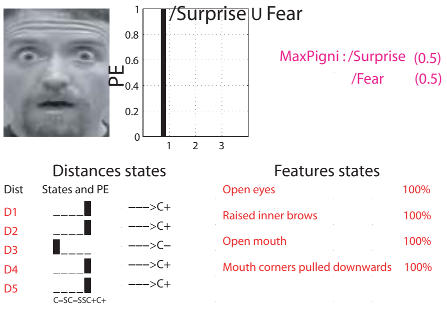


Fig. 4. Example of classification of *Surprise* expression

results. The best classification rates are obtained for *Happy*, *Anger*, *Surprise* and *Disgust* (see Total bars in Fig. 3). Poor performance is obtained for *Sadness*. The reasons could be the difficulty to simulate this facial expression and may also reflect the fact that the classification rule we used for *Sadness* lacks important pieces of information. The last bar for all the expressions corresponds to the total ignorance of the system (see Fig. 3 Ign.) It corresponds to the facial features deformations which do not correspond to any of the seven considered expressions and thus recognized as *Unknown* expressions. However the pignistic probability is based on normalized BBAs (see Sect. 4.3) where the piece of evidence for *Unknown* is redistributed on the whole set of the expressions.

5.2 Results with the Inhibition of the Distances

For the simulation reported in this section all revealed characteristic distances were used (see Sect. 4.1). The classification performances for the human observers are also reported. The best performances were obtained for *Anger*, *Surprise* and *Fear*. This is consistent with human observers (see Fig. 5). The classification rate for *Happy* is lower than in the first simulation and lower than human observers'. The worst classification rate was obtained with *Sadness*.

The inhibition process seems to have affected the model's behavior in three ways. First, the appearance of new double results (*Sadness* and *Anger*, *Disgust* and *Anger*) is due to the inhibition of the specific characteristic distances which previously allowed to avoid their appearance. Second, compared to the previous simulation for which all characteristic distances were included fully, this simulation resulted in similar but noisier classifications. In Fig. 6, it can be observed the system hesitation between *Happy, Disgust, Surprise* and *Fear* based on the distance D_4, the only one available on this particular combination of facial expression image and Gaussian apertures. The characteristic distances required to dissociate between them are inhibited. Third, the inhibition of some characteristic distances reduced the Ignorance rates for *Disgust, Fear* and *Sadness* (see Fig. 5 Ign).

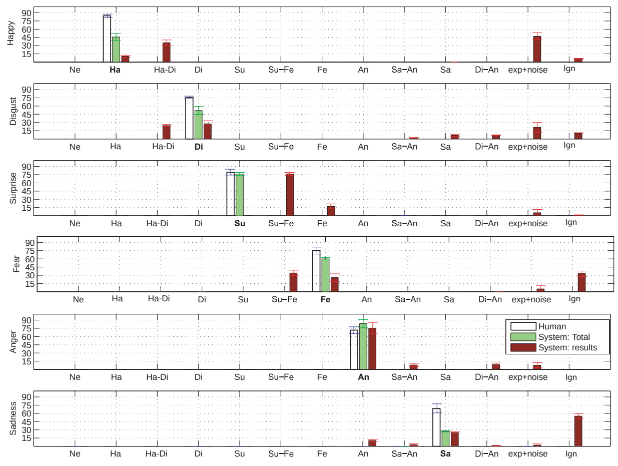


Fig. 5. The mean and standard deviation of the classification results with the inhibition process of the characteristic distances. Ne: *Neutral*, Ha: *Happy*, Di: *Disgust*, Su: *Surprise*, Fe: *Fear*, An: *Anger*, Ing: *Total ignorance*.

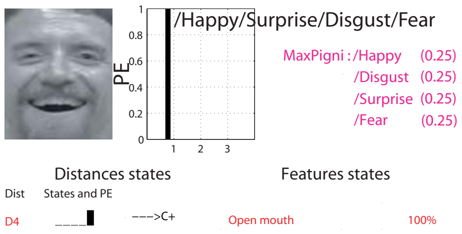


Fig. 6. Classification results of the *Happy* expression with the inhibition of the characteristic distances D_1, D_2, D_3 and D_5

5.3 Results with the Appearance Intensity of the Distances

The best classification rates were obtained for *Anger*, *Surprise* and *Fear* (see Fig. 5). The classification rates for *Sadness* and *Disgust* increased compared to the second simulation in which the characteristic distances revealed by the Gaussian apertures were fully used. *Happy* results do not change. The classification rates of all the expressions increased compared to the second simulation except for *Happy*. Moreover it can be seen a decrease of the Ignorance rates of all the expressions. Based on the same data the classification rates are very similar to those of human observers except for *Happy* (see Fig. 7). Compared with the inhibition process, the addition of the appearance intensity of the characteristic distances improved the classification. This result can be explained by the fact that the used characteristic distances have not the same importance in the recognition of the studied expressions. Weighting them in a random fashion

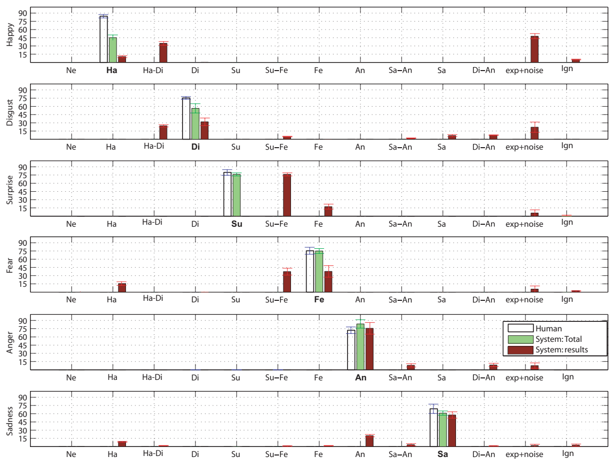


Fig. 7. The mean and standard deviation of the classification results with a discounting process according to the appearance intensities of the characteristic distances. Ne: *Neutral*, Ha: *Happy*, Di: *Disgust*, Su: *Surprise*, Fe: *Fear*, An: *Anger*, Ign: *Total ignorance*.

such as happens in a *Bubbles* experiment influences the classification results. In fact, this is the rational for *Bubbles* experiments.

At this point, it is clear that the inhibition of some characteristic distances leads to the decrease in classification performance and an increase in noise. In contrast, weighting the characteristic distances with the mean of the intensity of their end points leads to a better fit between the model and human observer. However, what is the difference between the visual cues used by human observers and the characteristic distances used by the proposed system?

5.4 Relative Importances of the Characteristic Distances

To better understand the results of the last simulation, we examined the relative importance of the five characteristic distances for the recognition of the basic facial expressions. The classification efficiency F of the system for each expression according to each characteristic distance is computed using $d^{'}$:

$$d^{'} = Z(correct) - Z(incorrect). \tag{8}$$

The $d^{'}$ associated with all characteristic distances for each facial expression is called $d^{'}_{E_e}$. The $d^{'}$ associated with all the characteristic distances D_j $(1 \leq j \leq 5)$ except one D_i $(1 \leq i \leq 5,\ i \neq j)$ is called $d^{'}_{D_i,E_e}$. Based on $d^{'}_{E_e}$ and $d^{'}_{D_i,E_e}$, the efficiency F_{D_i,E_e} of classification of each facial expression E_e according to each characteristic distance D_i is computed as:

$$F_{D_i,E_e} = \left(\frac{d'_{D_i,E_e}}{d^{'}_{E_i}}\right)^2 \tag{9}$$

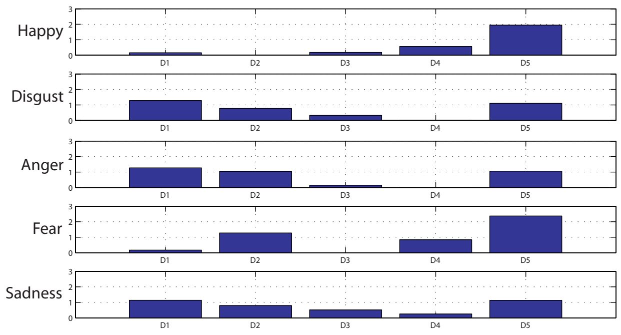


Fig. 8. Classification efficiencies in function of which characteristic distances was inhibited

We shall focus on three aspects of the obtained efficiencies displayed in Fig. 8 [3]. The system's efficiencies related to the inhibition of the 5 characteristic distances are compared to Smith *et al* results. Except for *Anger* expression their is an excellent correspondence between the most important characteristic distances for the proposed model and the facial cues used by the Smith *et al* model observer. The later uses all the information available to perform the task optimally. These results allow to conclude that the used characteristic distances summarize the most important information necessary for the classification of the facial expressions. However the visual cues used by human observer are different from those used by the *Smith et al.* model observer and the proposed model. In some cases human observers show a suboptimal use of the available information for the facial expressions classification [9]. For example, humans use the mouth but do not use the eyes for *Happy* and they use the eyes but not the mouth for *Fear*. In other cases human uses information which are not optimal; for example the nasolabial furrow in the case of *Disgust* (the model only uses the eyebrows) and the wrinkles in the case of *Sadness* (the model uses eyebrows and mouth). Given that humans easily outperform machines in everyday situations at recognizing facial expressions, however, it appears more likely that their alleged "suboptimalities", in fact, reflects robust everyday facial expression statistics. Thus it could be of great interest to use these "suboptimal" features for the facial features classification in future implementation of the proposed model.

6 Conclusion

The originality of the proposed work is the modeling of the *Bubbles* based experiment and the comparison of its classification results with those of human observers. We have modified the TBM based model for recognizing facial

[3] The classification efficiency is not reported for the *Surprise* expression because it is always recognized as a mixture of *Surprise* and *Fear*.

expressions proposed by [8] to allow it to process partially occluded faces. The modified TBM based model better fits the human data than the original TBM based model. However, further analysis revealed important differences between the behavior of the modified TBM based model and human observers. Given that humans are extremely well adapted to real-life facial expression recognition, future work will focus on weighting each visual cues during the classification process according to its importance for the expected expression and on adding other visual cues used by human observers such as wrinkles. This will be made easy thanks to the fusion architecture based on the TBM.

References

1. Darwin, C.: The expression of the emotions in man and animals. London, Murray (1872)
2. Ekman, P., Friesen, W.V., Ellsworth, P.: Emotion in the human face. Pergamon Press, New York (1972)
3. Izard, C.: The face of emotion. Appleton-Century-Crofts New York (1971)
4. Pantic, M., Rothkrantz, L.: Automatic analysis of facial expressions: The state of the art. IEEE Transactions On Pattern Analysis and Machine Intelligence 22(12), 1424–1445 (2000)
5. Gao, Y., Leung, M., Hui, S., Tananda, M.: Facial expression recognition from line-based caricatures. IEEE Transaction on System, Man and Cybernetics - PART A: System and Humans 33(3) (2003)
6. Abboud, B., Davoine, F., Dang, M.: Facial expression recognition and synthesis based on appearance model. Signal Processing: Image Communication 19(8), 723–740 (2004)
7. Cohen, I., Cozman, F., Sebe, N., Cirelo, M., Huang, T.: Learning bayesian network classifiers for facial expression recognition using both labeled and unlabeled data. Proc. IEEE Computer Vision and Pattern Recognition (2003)
8. Hammal, Z., Couvreur, L., Caplier, A., Rombaut, M.: Facial expressions classification: A new approach based on transferable belief model. International Journal of Approximate Reasoning (2007), doi: 10.1016/j.ijar.2007.02.003
9. Smith, M., Cottrell, G., Gosselin, F., Schyns, P.: Transmitting and decoding facial expressions of emotions. Psychological Science 16, 184–189 (2005)
10. Gosselin, F., Schyns, P.: Bubbles: A technique to reveal the use of information in recognition. Vision Research 41, 2261–2271 (2001)
11. Smets, P.: Data fusion in the transferable belief model. In: Proc. of International Conference on Information Fusion, Paris, France, pp. 21–33 (2000)
12. Smets, P., Kruse, R.: The transferable belief model. Artificial Intelligence 66, 191–234 (1994)
13. Smets, P.: Decision making in the tbm: the necessity of the pignistic transformation. International Journal of Approximate Reasoning 38, 133–147 (2005)

3D Face Scanning Systems Based on Invisible Infrared Coded Light

Daniel Modrow[1], Claudio Laloni[1], Guenter Doemens[1], and Gerhard Rigoll[2]

[1] Siemens AG, Corporate Technology,
Machine Vision & Nondestructive Testing, Munich, Germany
[2] Technische Universität München
Institute for Human-Machine Communication, Munich, Germany

Abstract. In this paper we present two new sensor systems for 3D data acquisition. Both systems are primarily designed for face scanning applications, but their performance characteristics are useful for technical applications as well. Their operating principle is based on structured light, more precisely on color coded light, implemented in the invisible near infrared spectrum. The resulting 3D data acquisition works in real-time, shows high robustness against ambient light and provides accurate high resolution depth maps.

Thus, most of the limiting factors known from other face scanning systems are eliminated, such as visible annoying and blinding light patterns, motion constraints and highly restricted application scenarios due to ambient light constraints.

1 Introduction

Obtaining 3D data from general objects has been one of the most important tasks in computer vision over the last years. The increasing demand on 3D data from human faces is motivated by current entertainment, guidance or learning applications, from video games to information systems, which make more and more use of human like avatars (e.g. [1]). In the field of face recognition based security applications, the additional use of 3D face data has been proved to substantially increase the reliability and robustness ([2]).

Most current 3D data acquisition techniques have significant disadvantages recording human faces. These are maybe acceptable for specific 3D acquisitions, e.g. a single scan for an avatar modeling process, but not when it comes to a day-to-day face recognition security application.

All systems using visible structured or coded light ([3], [4]) suffer in general from inconvenient blinding effects. Depending on the applied light coding scheme, a sequence of images has to be acquired, which additionally leads to user motion constraints during data acquisition.

Systems that are using quite simple structured infrared light patterns ([3], [5]) or time of flight methods ([6]) produce quite inaccurate depth data of about $1-5mm$ at their sensor resolution. Line scanning systems, e.g. laser scanners ([7]), are capable acquiring accurate depth data ($< 1mm$), but need a comparably long acquisition time ($> 0.3s$), leading to motion constraints as mentioned above.

In this paper we present two 3D sensor systems that are primarily designed for face scanning applications, especially day-to-day face recognition security applications. Both systems eliminate the disadvantages of currently available systems.

2 The Working Principle

The working principle of the proposed sensors is based on the color coded triangulation (CCT) method that has been developed by Forster in [4]. With this method, a projection unit illuminates a given scene, e.g. a face, with a 1-dimensional spatial encoded pattern composed of stripes of 8 different colors. Typically a standard video projector or a custom made one with a special interference slide is used for this purpose. Simultaneously an image of the illuminated scene is taken by a color camera. The color stripe sequence is encoded such that each stripe can be identified unambiguously. Consequently triangulation can be applied to compute a 3D depth map from a single acquired image of the scene, which makes the method real-time capable. For more details on the color coded triangulation method we refer to Forster's work, a general overview on coded light techniques can be found in [8].

The key idea of the sensors proposed here is to keep the unique color coding scheme and to apply this to a pattern that is projected in the invisible near infrared spectrum. Thus the real-time requirement as well as the requirement for an unobtrusive sensing technology can be met.

3 The One-Shot Dual Wavelength System

As mentioned above, the light pattern of the color coded light approach consists of stripes of 8 different colors. These are combinations of the primary colors red (R), green (G) and blue (B), which can be easily produced and recorded in the visible light spectrum.

Discerning between distinct colors in the visible light spectrum is an easy task for a color camera that uses a Bayer pattern or even a 3-chip design. The same approach could be used in the infrared band, but special cameras would be needed. Since the idea is to build a sensor mainly with off-the-shelf components, we have decided to use a different approach that uses only two different wavelengths in the infrared light spectrum. This simplifies the problem to distinguish a certain wavelength, but results only in two possible color channels with respectively four different colors, as shown as follows:

3 primary colors (R, G, B) $\Rightarrow$
8 resulting colors: black, red, green, blue, yellow, magenta, cyan, white
2 basic wavelengths (λ_1 and λ_2) $\Rightarrow$
4 resulting colors: black, λ_1, λ_2, $\lambda_1\lambda_2$

As presented in table 1 the use of 8 colors allows a large number of codewords. For a coded light system as depicted in [4], a pattern with 106 codewords has been encoded. This type of pattern consists only of codewords with a change of at least two channels per edge. Thus it is robust against some decoding errors and further allows their correction due to its built in redundancy. In contrast to this, the 4 color approach provides only 104 four letter codewords without any redundancy and only a single changing channel. But since the targeted type of scene - human faces - offers quite manageable surface conditions, a-priori knowledge can be used to guarantee an exact de-coding of the pattern.

Table 1. Number of codewords (CW) in a 4 letter encoding, depending on the number of used color channels and changes per edge

$NR_{channels}$	NR_{colors}	$\frac{channels}{edge}$	$CW_{possible}$	CW_{valid}
3	8	3	8	8
3	8	2	512	500
3	8	1	2744	2426
2	4	2	4	4
2	4	1	108	104

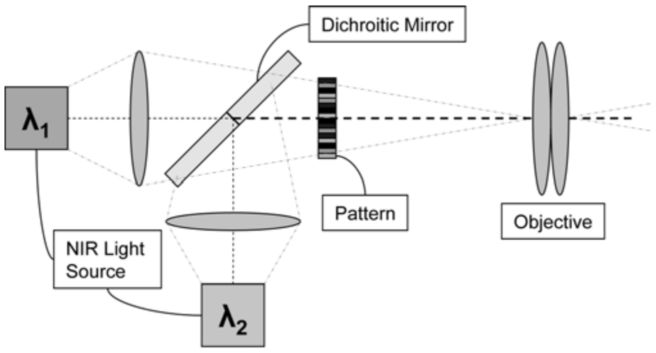


Fig. 1. Projection unit for the one-shot dual wavelength approach. The two wavelengths are combined and pass the interference slide to the objective lenses.

To generate a pattern with these "colors", a light source with two different wavelengths λ_1 and λ_2 is necessary. An appropriate way to solve this is the combination of two infrared light sources by a dichroitic mirror that is penetrable in each direction only for one wavelength and reflects the other one. The layout of such a projection unit is shown in Figure 1. A customized interference slide with a size of $10 \times 10mm^2$ and a stripe width of $50\mu m$ is placed in the optical path of the merged two-wavelength light and produces the final pattern with the color code.

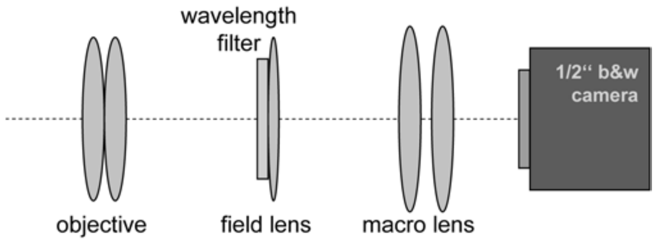


Fig. 2. Acquisition unit for the one-shot dual wavelength approach. The image passes a wavelength filter that divides the two wavelengths to different pixels whereby the distinct "colors" can be allocated.

For the image acquisition a standard Firewire video camera with a 1.3 megapixel Sony ICX-285 CCD sensor is used. This sensor has its main spectral response in the visible light spectrum, but is also quite sensitive in the NIR spectrum. At the used wavelengths, which are $\lambda_1 = 760nm$ and $\lambda_2 = 850nm$ in this case, the relative spectral response of the sensor lies between 35% and 50%.

To distinguish the four "colors" in the camera image, an additional filter has been placed into the optical path of the camera as shown in Figure 2. This filter works similar to the Bayer filter used in standard color cameras to produce color dependent pixel information on a single light sensitive chip. In our case the filter consists of stripes that are alternating transparent for λ_1 and λ_2. This design allows to determine which pixel is receptive for which wavelength. With this information we are able to build up images with four infrared "colors" by interpolating the missing pixels in each of the two channels. The loss of resolution incurred by this approach is the reason for using a higher resolution camera to receive a depthmap at the desired video resolution with approximately 450.000 measurement points. On this image we can then apply an algorithm for extracting the codewords and computing the range data.

4 Dual-Shot One Wavelength System

The second approach puts even more emphasis on the goal of using mainly off-the-shelf components than the first one. Thus, the underlying principle is quite similar, what means the pattern used to encode the light planes is still 1-dimensional and constructed by using 2 channels, implicating again 4 colors.

But this time the pattern is not only spatial encoded and projected at once. It could be more precisely called "timed-color-coded", because it is a combination of time and spatial coding. The pattern is divided into two channels, resulting in 2 single slides which are projected consecutively [Figure 3]. Because these slides can be either penetrable or reflective for all wavelengths and not only for specific ones, the use of simple chrome masks is possible. Compared to the interference slides they are easier and cheaper to manufacture and furthermore provide sharper edges, due to the manufacturing process. Again, the size of each chrome mask is $10 \times 10mm^2$ with a code stripe width of $50\mu m$.

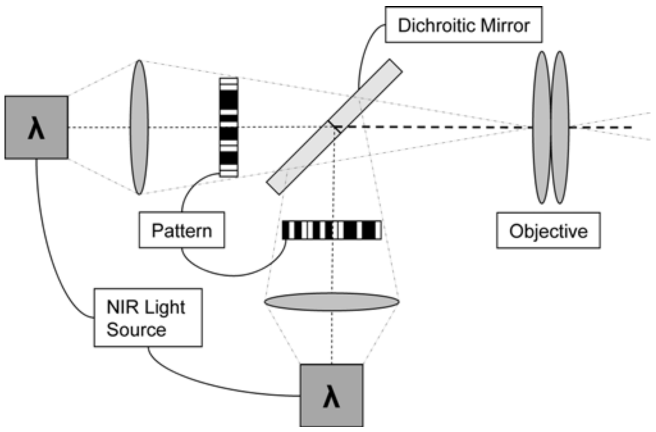


Fig. 3. Projection unit for the dual-shot one wavelength approach. The coded light pattern is divided into two slides, which are combined by a dichroitic mirror to the final 2-channel pattern.

With the consecutive projection of the patterns, two grayscale images are received. Their combination again comes out as the desired 2 channel coded light image as in the following listing, whereas t_i denotes the presence of wavelength λ at time t_i.

1 basic wavelength λ, consecutive projected $\Rightarrow$
    4 resulting colors: black, λ_{t_1}, λ_{t_2}, $\lambda_{t_1}\lambda_{t_2}$

On the camera side, in this case no special optics with a wavelength filter is needed, because we simply have to record standard grayscale images. This allows the use of a cheaper, lower resolving camera as for the one-shot approach, because no image reconstruction and wavelength detection is applied. However, the fact of recording two or more images causes an increase in total acquisition time - or demands a faster and more sensitive camera. To reach an acceptable agreement between performance and sensitivity in the infrared band, for this approach a Firewire camera with the Sony ICX-415 CCD sensor is used. This sensor has a relative spectral response of 50% at the used wavelength, which is $\lambda = 735nm$ in this case. For further development, the more sensitive ICX-429 sensor will be investigated. To cut off the unwanted visible and infrared light and make the image acquisition and measurement more robust, a bandpass filter has been designed with transmission range at the projected wavelength. It is placed in the optical path of the camera and allows acquisition under different illumination conditions.

5 Comparison of the Two Approaches

Looking at the first design with two different wavelengths, the main advantage is clearly that a single shot image provides a full 3D depth map. That permits capturing 3D data at full video rate and fully eliminates problems with moving objects.

Due to the manufacturing process of the interference slides two adjacent code stripes have no clear border, but rather a small region of undefined filtering conditions. This leads to aberrations in the required color image reconstruction and causes artifacts in the resulting depthmap, especially under bad illumination conditions or at objects with abrupt changing surface heights. The resulting errors can be compensated up to a certain level by interpolation algorithms - or in future by an improved filter manufacturing process.

In contrast to this, the chrome mask in the second approach can be manufactured very accurately, resulting in very exact stripe widths and clear borders. Furthermore, the use of the same wavelength for both patterns no longer requires wavelength dependent image reconstruction. It allows the use of lower resolving cameras, because the full resolution now can be used for each image and interpolation isn't needed anymore.

A minor drawback is the fact that two images have to be taken to get all data needed for a 3D reconstruction. But with the high illumination power of today's LEDs, the shutter time of the camera can be reduced so far that the acquisition time for both images is less than $50ms$, what means a possible capture rate of 20 frames per second.

The performance of both approaches can be further improved when a reference image illuminated under the used wavelength - without any pattern - is recorded and added to the algorithm. This image is used to compensate the scene reflectance and supports the correct edge decoding even on heavily textured surfaces.

6 Experimental Results

6.1 Resolution and Accuracy

Figure 4 presents the sensor systems that have been built up according to the two approaches and corresponding sample data. The head has been recorded frontally, thus the images show the coverage of a single acquisition.

Both presented approaches allow the measurement in a working space of $50 \times 50 \times 50 cm^3$ in a distance of approximately $60cm$ from the sensor. Depending on the camera and pattern resolution, the accuracy in depth is about $0.3mm$ at more than 450.000 measurement points. The acquisition of a depthmap takes between 50 and $100ms$. Thus, it is possible to capture and compute a depthmap of a scene in full video resolution at a video rate of 10-20fps.

The depth error δz of a structured light system and the depth deviation $\sigma(\delta z)$ can be computed with the following formulas (see also [4]),

$$|\delta z| = \frac{dx\, z}{b\, f} \tag{1}$$

$$\sigma(\delta z) = \frac{dx\, z^2}{\sqrt{12}\, f\, (b + z_P\, tan\theta)} \tag{2}$$

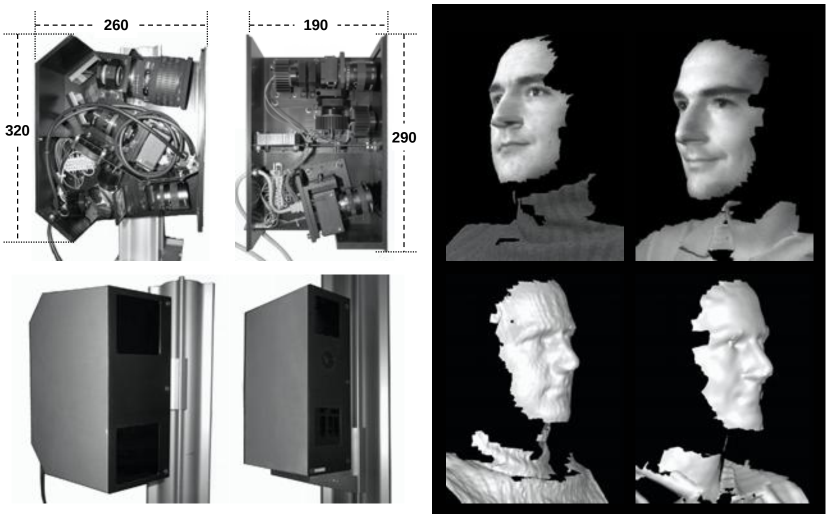


Fig. 4. Proposed sensor systems and sample data. The left sensor shows the one-shot dual wavelength approach, the right one the dual-shot one wavelength system. Their dimensions are indicated in [mm]. The sample results show a frontal recorded human head. On the left side following the one-shot principle, on the right side recorded with the dual-shot sensor.

whereas dx denotes the camera's effective pixel size, b the baseline between camera and projection unit, f the focal length of the camera, z the actual distance in camera coordinates, z_P the shift of the projection unit relative to the camera and θ the triangulation angle. With these data, an error prediction for each distance is possible, as it has been done in table 2. In this case, the near infrared sensor of the second approach is compared to a color coded light system that is working in the visible light spectrum with a LCD video projector as its projection unit. Both cameras have identical intrinsic parameters, but differ in the fact that one is using a Bayer pattern for capturing color data in the visible light. The one-shot approach hasn't been taken into account because of its different intrinsic parameters and its reconstruction errors caused by the interference slides. The predicted depth error has been calculated according to formula 2. The recorded object was a calibration board with a very flat surface that has been taken as $z_W = 0$ plane. To get comparable data from the measurements, the standard z-deviation to a fitted plane in space has been calculated using a least-squares method.

Corresponding data is presented in figure 5, whereas two more advantages of the new projection and recording technique become visible. Firstly the fact that the infrared projector does not have a discrete grid like a video projector. This yields in a clearly reduction of Moiré effects, which often disturb the depthmap when using LCD or similar projectors. Further the recording of the image does not need a color reconstruction step, e.g. by using a Bayer pattern,

Table 2. Measured and predicted statistical parameters of the depth-deviation and distribution in a depthmap of a planar object

type	mean dist. [mm]	mean value ($z_W = 0$ plane)	$\sigma(\delta z)$ (measured)	$\sigma(\delta z)$ predicted
NIR	625	0.08	0.14	0.204
NIR	665	0.07	0.14	0.231
vis. LCD	745	0.08	0.16	0.219
vis. LCD	795	-0.02	0.17	0.250

as in the visible light. Thus no artifacts caused by color interpolation appear on the reconstructed depthmap, which provides a quite flat surface in its model view.

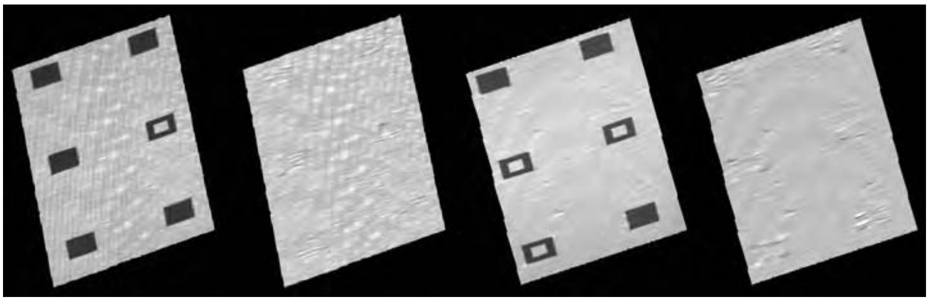


Fig. 5. Sample data, taken from a calibration board. On the left side recorded with a visible color coded light system, on the right side with a near infrared sensor. The first and third image show corresponding texture information, the second and fourth only the computed depth data. The depth artifacts caused by the calibration marks have to be eliminated during further development.

6.2 Robustness Against Ambient Light

As already mentioned in chapters 3 and 4, a bandpass filter is placed in the optical path of the camera to cut off unwanted and disturbing ambient light.

To measure the efficiency of the filter and to get a comparison to systems working with visible light, several tests have been applied on the new sensor systems.

Table 3 shows the results of a contrast measurement in the direct projection path. The illumination power of a fully (L_{max}) and a not enlightened (L_{min}) stripe of the coded pattern has been taken to calculate the contrast ratio according to the following formula.

$$Contrast\ ratio = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{3}$$

Table 3. Achieved contrast values. Contrast of the near infrared dual-shot one wavelength system compared to a visible light dual-shot system, built on a standard LED video projector, under different ambient light conditions.

	illuminance	NIR	NIR, filtered	LED projector
no light	0 lx	0.889	0.819	0.769
neon light	150 lx	0.564	0.790	0.618
halogen lamp	400 lx	0.136	0.400	0.183
daylight	1200 lx	0.044	0.124	0.040

Table 4. Rough classification of sensors and approaches for 3d measurements

method	accuracy	speed	system costs	face scanning
time-of-flight	-	+++	0	0
laser scanning	+++	- -	- -	-
passive stereo	0	++	++	+
visual CCT	++	++	++	0
NIR CCT, 1-shot	+	++	+	+++
NIR CCT, 2-shot	++	+	+	++

To prove the robustness against ambient light, the measurement has been disturbed by a common neon room light, a halogen lamp and daylight. Although intensity or power measurements of light are aligned on human light perception and don't regard the infrared spectrum, the illuminance on the measurement object is also indicated to get an impression of the ambient brightness. To further prove the pre-eminence against a visible light system, a comparable sensor has been built with a standard off-the-shelf video projector. This projector (Toshiba F1) uses a high power LED as its light source which spectral power is nearly equal to the one of the infrared LEDs.

On both systems the contrast can be taken as an indicator for the image and measurement quality, because the applied 3D-algorithm is edge-based. The comparison tests showed that with the new infrared sensors the contrast ratio doubles at optimal conditions, at scenarios with ambient light even an increase of a factor of 3 could be reached.

If we summarize all technical data, experimental results and measurements, the new sensor systems are positioned between current state-of-the art products and approaches as shown in table 4. As their technical parameters can be measured and calculated, their suitability for face scanning applications is based on the measurement accuracy and the positive acceptance of the user. Because infrared light can not be perceived by humans, the proposed sensors are perfectly suited for such applications.

7 Summary

In this paper we have presented two new 3D sensor systems based on optical triangulation and working in the infrared band with coded light. The sensors provide high resolution 3D data at nearly video frame rate. In addition the proposed systems offer a high robustness against ambient light. The sensor design allows a very cheap and easy implementation due to its off-the-self components. This qualifies the use as well for security as for entertainment applications. It is suited for static and moving objects without any user inconvenience, since the projected pattern is invisible for humans.

Acknowledgement. This work has been supported by the EU FP6 integrated project PASION "Psychologically Augmented Social Interaction Over Networks" (IST-2006-027654).

References

1. Jung, B., Kopp, S.: FlurMax: An interactive virtual agent for entertaining visitors in a hallway. In: IVA 2003, pp. 23–26 (2003)
2. Phillips, P.J., Scruggs, W.T., O'Toole, A.J., Flynn, P.J., Bowyer, K.W., Schott, C. L., Sharpe, M.: FRVT 2006 and ICE 2006 large-scale results. Technical report, NISTIR 7408 (2007)
3. Chang, N.L.: Creating interactive 3-D media with projector-camera systems. In: Proceedings of SPIE, vol. 5308, pp. 850–861 (2004)
4. Forster, F.: A high-resolution and high accuracy real-time 3D sensor based on structured light. In: 3DPVT 2006, pp. 208–215 (2006)
5. Hall-Holt, O., Rusinkiewicz, S.: Stripe boundary codes for real-time structured-light range scanning of moving objects. In: ICCV 2001. Eigth International Conference on Computer Vision, pp. 359–366 (2001)
6. Iddan, G.J., Yahav, G.: 3D imaging in the studio. In: Proceedings of SPIE, vol. 4298, pp. 48–55 (2001)
7. Song, H., Lee, S., Kim, J., Sohn, K.: Three-dimensional sensor based face recognition. Appl. Opt. 44, 677–687 (2005)
8. Pagés, J., Salvi, J., García, R., Matabosch, C.: Overview of coded light projection techniques for automatic 3D profiling. In: ICRA 2003, pp. 133–138 (2003)

A New Statistical Model Combining Shape and Spherical Harmonics Illumination for Face Reconstruction

Abdelrehim Ahmed and Aly Farag

Computer Vision and Image Processing Laboratory (CVIP)
University of Louisville, Louisville, KY, 40292

Abstract. This paper develops a new statistical model that can be used to recover the 3D face shape from a single image taken under arbitrary lighting conditions. The proposed statistical model combines shape and image intensity information. The latter is represented by a spherical harmonics (SH) space. Given a training data set of aligned height maps of faces and their corresponding albedos, The input image is projected into the space spanned by the SH basis images of each member in the data set. The combined statistical model is obtained by performing the PCA on both the SH projections and the height maps. Finally, the face is reconstructed by fitting the model to the input intensity image. In addition to the shape recovery, the proposed model can be used to recover the face albedo and to estimate the light direction. Experiments have been conducted to evaluate the performance of the proposed approach.

1 Introduction

3D human face reconstruction is an active area of research in both computer vision and computer graphics with applications to 3D face recognition [1,2] , graphic animation [3], medical field [4] etc. Many approaches have been adopted to recover the 3D information of a face including passive methods e.g., geometric stereo [5], space carving [6], shape from shading [7] and active methods e.g., structured light [8] and 3D laser scanners. Active methods give precise reconstruction, however their use is limited due to the high cost of the 3D scanners or the infeasibility of projecting light pattern onto the subject face in daily life situations. One of the most promising paradigms that is used to overcome many of problems of 3D face reconstruction is statistical-based techniques. Based on strong prior knowledge of the statistical variation of shape and albedo of the human face, Blanz and Vetter [9] proposed an impressive appearance-based approach. A single intensity image is used to reconstruct the face that is assumed to have Phong reflectance [9]. Dimitrijevic et al. [10] proposed a model-based structure-from-motion approach for faces reconstruction from video sequences. An optimization technique is used to find the shape model parameters given pairwise image correspondences as input [10]. Quite accurate results have been obtained by Smith and Hancock [11] by embedding a statistical model of the face shape within the framework of shape from shading (SFS). Combining a statistical global constraint and local irradiance constraint helps in solving the well-known SFS problems e.g., the convex-concave ambiguity and albedo variations [11].

In their Active Appearance Model (AAM), Cootes et al. [12] presented a simple method of interpreting images. In AAM a coupled statistical model is generated to model the 2D shape information and the image intensity appearance of the object of interest. Inspired by the work in [12], Castelan et al. [13] developed a coupled statistical model to recover the 3D shape of the human face from intensity images. The approach can handle faces illuminated by a frontal light source. The statistical model simultaneously models the 3D shape information and the image intensity. The *eigenmodes* of the gray-level intensity images and their corresponding 3D shapes of a large set of aligned faces are obtained by the PCA. Finally, the coupled model is obtained by concatenating the eigenmodes of both the intensity and the shape followed by applying the PCA on the result. To recover the 3D shape of a new face, an optimization method is used to find the couple model parameters that fit the input intensity image [13].

Changing of the face appearance due to the unconstrained illumination is one of the major problems in both the face reconstruction and face recognition algorithms. Developing a face reconstruction algorithm that is robust to the variation in the illumination conditions is a very challenging task. In this paper, we aim at developing a statistical model that is capable of recovering the 3D shape of human faces from images under arbitrary unknown lighting conditions. To achieve this goal, we integrate the spherical harmonics representation of illumination [14,15,16,17] in the statistical model proposed in [13]. As another contribution, we explain how to use the proposed model to recover the gray-level albedo of the face. Using the recovered 3D shape and albedo, an accurate estimation for the light direction can be obtained.

2 Spherical Harmonics Illumination Representation

The Spherical Harmonics (SH) representation is a mathematical system analogous to the Fourier transform but defined across the surface of a sphere. Basri and Jacobs [14] and Ramamoorthi and Hanrahan [16] have used the SH representation to obtain an interesting result for convex Lambertian objects. They have shown that by expressing the lighting and the surface reflectance as functions in the SH space, the image irradiance equation can be described as a convolution process between the light and the surface reflectance. In this convolution the Lambert's reflection acts as a low pass filter with about 99.2 percent of energy in the first nine spherical harmonics. This means that it is sufficient to use only 9D linear subspace to accurately approximate the images of a convex Lambertian object obtained under a wide variety of illumination conditions.

Let p_i denote the ith point of an object that has N_p points. Let ρ_i denote the surface albedo of p_i and ρ denote a $N_p \times 1$ vector containing the values of albedo for all points. Similarly, let n_x, n_y and n_z denote three vectors of the same length as ρ and contain the x, y and z components of the surface normals. Using this notation, the first nine harmonics images (in Cartesian coordinates) of the objects are obtained as follows:

$$\mathfrak{b}_{0,0} = \sqrt{\frac{1}{4\pi}}\,\rho \quad \mathfrak{b}_{1,0} = \sqrt{\frac{3}{4\pi}}\,\rho \circ n_z \quad \mathfrak{b}_{1,1} = \sqrt{\frac{3}{4\pi}}\,\rho \circ n_x \quad \mathfrak{b}_{1,-1} = \sqrt{\frac{3}{4\pi}}\,\rho \circ n_y$$
$$\mathfrak{b}_{2,0} = \sqrt{\frac{5}{16\pi}}\,\rho \circ (2n_z \circ n_z - n_x \circ n_x - n_y \circ n_y) \quad \mathfrak{b}_{2,1} = \sqrt{\frac{15}{4\pi}}\,\rho \circ (n_x \circ n_z)$$
$$\mathfrak{b}_{2,-1} = \sqrt{\frac{15}{4\pi}}\rho \circ (n_y \circ n_z) \quad \mathfrak{b}_{2,2} = \sqrt{\frac{15}{16\pi}}\rho \circ (n_x \circ n_x - n_y \circ n_y) \quad \mathfrak{b}_{2,-2} = \sqrt{\frac{15}{4\pi}}\rho \circ (n_x \circ n_y) \tag{1}$$

where the symbol $\circ$ denotes the Hadamard product (the component-wise product) between two vectors. Any image I of a convex Lambertian object under arbitrary lighting condition can be approximated by a weighted combination, α, of the harmonics basis images as:

$$I \approx B\,\alpha \tag{2}$$

where B is an $N_p \times N_r$ matrix denotes the basis images and N_r is the number of basis images used. Every column vector of the matrix B contains one harmonic image $\mathfrak{b}_{nm}$. QR decomposition can be applied on B to obtain an orthonormal basis. The decomposition computes two matrices, Q an $N_p \times N_r$ matrix with orthonormal columns, and R, an $N_r \times N_r$ upper triangular matrix such that:

$$QR = B \tag{3}$$

Then Q is an orthonormal basis for B, accordingly the projection of any image I into the space spanned by B can be expressed by [15]:

$$I_h = QQ^T\,I \tag{4}$$

3 Statistical Models for Face 3D Shape and Albedo

In this section we describe how to construct statistical models for faces 3D shape and gray-level albedo. We obtain these two models by following the approach presented by Castelan et al. [13]. Given a training data set of aligned 3D faces and their corresponding albedo, the models are obtained by applying the PCA as described in the following subsections.

3.1 Albedo Model

Let the $N_p \times 1$ vector a_k denote the k^{th} image of the training data set, where N_p is the number of image pixels (equal to the number of points in the object) . The centered data matrix A, is constructed by subtracting the average image $\bar{a}$ from all the images $A = [(a_1 - \bar{a}) \quad (a_2 - \bar{a}) \quad \ldots (a_{N_d} - \bar{a})]$, where N_d is the number of examples in the data set. The covariance matrix of the data is $C = AA^T$. The large size of the matrix C makes it difficult to compute its eigenvalues and eigenvector directly. Instead, the numerically efficient snap-shot method of Sirovich [18] can be used [13]. Using the eigenvectors, u_i, any example can then be approximated using:

$$\widetilde{a} = \bar{a} + P_a b_a, \tag{5}$$

where P_a is a set of N_a orthogonal modes of variation, $P_a = [u_1 \quad u_2 \quad \ldots u_{N_a}]$ and b_a is a set of albedo parameters.

3.2 3D Shape Model

In this model, the face surface is represented by the Cartesian coordinates (x, y, z). The height map $z(x, y)$ is used to represent the 3D shape. By following the procedure described in the Section 3.1, a linear shape model is obtained:

$$\widetilde{s} = \bar{s} + P_s b_s, \tag{6}$$

where $\bar{s}$ is the mean height. P_s is a set of N_s orthogonal modes of variation $P_s = [v_1 \quad v_2 \quad \ldots v_{N_s}]$, and v_i is the i^{th} eigenvector. b_s is a set of shape parameters.

4 Intensity Model Using Spherical Harmonics Images

As mentioned in Section 2, any face image taken under arbitrary illumination can be approximated accurately by nine spherical harmonics images (with the assumptions of convexity and Lambertian reflectance). Here, we describe how to make use of this result to build a statistical model, that is robust to variation in lighting conditions. From now on, we call this model the SHP (Spherical Harmonics Projection) model . Unlike the two models in Section 3.1 and Section 3.2, this model depends on the input intensity image, for which we want to reconstruct the face. More precisely, the SHP model is constructed from the projection of the input image into the spherical harmonics subspace of each example in the training data set.

Let I denote the input image, then the projection of I into the spherical harmonics subspace of the k^{th} example is found by:

$$h_k = Q_k Q_k^T \, I, \tag{7}$$

where Q_k is the orthonormal basis for the harmonics images matrix B_k, obtained for the k^{th} example (see Section 2). The centered data matrix, H, is constructed from h_k

$$H = [(h_1 - \bar{h}) \quad (h_2 - \bar{h}) \quad \ldots (h_{N_d} - \bar{h})] \tag{8}$$

where $\bar{h}$ is the average image. By following the procedure described Section 3.1, a linear model is obtained:

$$\widetilde{h} = \bar{h} + P_h b_h, \tag{9}$$

where P_h is a set of orthogonal modes of variation, $P_h = [w_1 \quad w_2 \quad \ldots w_{N_h}]$, and w_i is the i^{th} eigenvector. b_h is a set of SHP model parameters.

5 Combining Shape, Albedo and SHP Models

As explained in Section 3 and 4, the face 3D shape and image appearance of any example can be described by the vectors b_a, b_s and b_h. In this section, we describe how to combine these vectors into a single model that can be used to recover the face 3D shape given a single image obtained under arbitrary lighting. For the k^{th} training sample, we form a concatenated parameter vector from the three vectors b_a, b_s and b_h as follows:

$$b_{c_k} = \begin{pmatrix} W_h \ b_{h_k} \\ W_a \ b_{a_k} \\ b_{s_k} \end{pmatrix} = \begin{pmatrix} W_h \ (P_h^T(h_k - \bar{h})) \\ W_a \ (P_a^T(a_k - \bar{a})) \\ (P_s^T(s_k - \bar{s})) \end{pmatrix} \tag{10}$$

where W_h and W_a are two diagonal matrices of weights for the SHP and the albedo models. These weights compensate for the difference in units between the shape and the intensity. Similar to [12,13], we set $W_h = r_h\mathbf{I}$ where r_h^2 is the ratio of the total shape variance to the total SHP variance and $\mathbf{I}$ is the identity matrix. Similarly, $W_a = r_a\mathbf{I}$ where r_a^2 is the ratio of the total shape variance to the total albedo variance. All the concatenated vectors b_{c_k} are arranged to form one matrix $[b_{c_1} \quad b_{c_2} \ldots]$, then the PCA is applied on the result to give one model that links the variations of the three models

$$b = F\ c = \begin{pmatrix} F_h \\ F_a \\ F_s \end{pmatrix} c \tag{11}$$

where F are the eigenvectors and c is a vector of model parameters controlling the SHP, albedo and shape of the model. The matrices F_h, F_a and F_s represent the eigenvectors corresponding to the SHP, albedo and shape subspaces of the model. Since the combined model is linear, the three models can be expressed explicitly as functions of c

$$\widetilde{h} = \bar{h} + P_h W_h^{-1} F_h\ c\ , \tag{12}$$

$$\widetilde{a} = \bar{a} + P_a W_a^{-1} F_a\ c\ , \tag{13}$$

$$\widetilde{s} = \bar{s} + P_s F_s\ c\ . \tag{14}$$

5.1 Estimating the 3D Shape for a New Example

Given a new image, I, the 3D shape can be recovered by first finding the SHP model parameters vector b_h

$$b_h = P_h(I - \bar{h}), \tag{15}$$

Since b_h corresponds to $W_h^{-1}F_h c$ in the combined model, the parameters vector c can be estimated by minimizing the error between them

$$c = arg\ \min_c (b_h - W_h^{-1}F_h\ c)^T(b_h - W_h^{-1}F_h\ c) \tag{16}$$

consequently, the 3D shape can be recovered

$$\hat{s} = \bar{s} + P_s F_s\ c\ . \tag{17}$$

Using a Matlab implementation on a PC workstation with Pentium4 3.00GHz processor and 2 GB RAM, it takes about 15 seconds to recover the shape of each example.

5.2 Estimating the Albedo and the Light Direction

In addition to the 3D shape reconstruction, the model can be used to estimate the face albedo and the light direction. Using the value of the vector c computed from Eq. 16, the estimated face albedo is obtained

$$\hat{a} = \bar{a} + P_a W_a^{-1} F_a \; c \; . \tag{18}$$

The image irradiance equation at pixel i can be expressed as

$$I_i = \hat{a}_i \frac{\hat{n}_i \cdot L}{\|\hat{n}_i\|}, \tag{19}$$

where $L = (l_x, l_y, l_z)$ is a unit vector denoting the light direction and $\hat{n}_i = (-\frac{\partial \hat{s}_i}{\partial x}, -\frac{\partial \hat{s}_i}{\partial y}, 1)$. Accordingly, the estimation of the light direction can be formulated as a minimization problem

$$\min_{L \in \mathbb{R}^3} \sum_{i=1}^{N_p} \left| I_i - \hat{a}_i \frac{\hat{n}_i \cdot L}{\|\hat{n}_i\|} \right| . \tag{20}$$

6 Experimental Results

This section shows several experiments conducted to evaluate the performance of the proposed model in recovering the face 3D shape. In these experiments, we used the Chinese Face Database [19] to build the face models. This database contains a large collection of Chinese 3D faces acquired using a high resolution Cyberware 3D Laser Scanner [20]. Figure 1 shows two samples from the database, one for female and one for male. In all the experiments, we used 100 examples from the database to build the face models.

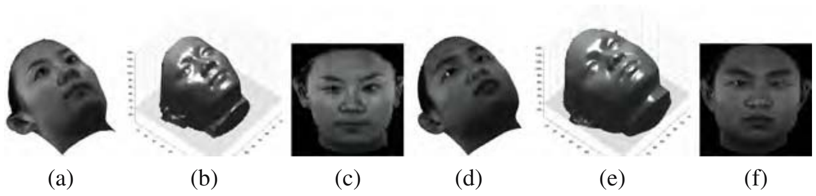


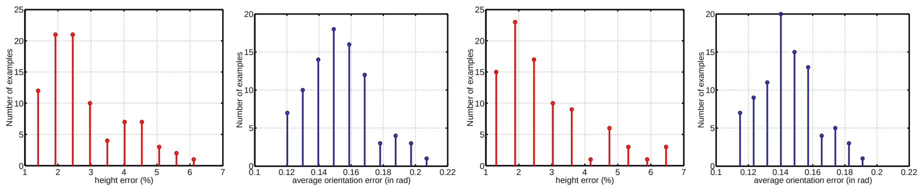
Fig. 1. Two samples from the BJUT database. The faces (3D+ color albedo) are shown in columns (a,d), while the height maps and the gray level albedo images are shown in columns (b,e) and (c,f) respectively.

Experiment#1. To quantify the reconstruction accuracy, we recover the 3D shape for 88 out-of-training-sample examples and for each example we compute the following error measures *Height Error* the recovered height map is compared with the ground truth height and mean absolute error is computed. *Surface Orientation Error* the directions of the recovered normal vectors are compared with the directions of normal vectors in the ground truth data. The average of the difference angle is computed as follows

$$\bar{\theta_k} = \frac{1}{N_p}\sum_{i=1}^{N_p} \arccos(\frac{\hat{n}_i \cdot n_i}{\|\hat{n}_i\|\|n_i\|}), \tag{21}$$

where n_i is the normal vector at point i of the ground truth data and $\hat{n}_i$ is the normal vector at the same point in the recovered surface.

The histogram of the height mean absolute error is plotted in Fig 2(a). As the figure illustrates, the maximum height error is less than 6.5%. Meanwhile, more than 72% of the examples have very small error (less than 3%). Similarly, the orientation error does not exceed 0.21 rad as shown on Fig 2(b). In order to investigate the robustness of the



(a) (b) (c) (d)

Fig. 2. Error histograms: (a,c) are the histograms of the height error for images under frontal lighting and $[-1,\ 0,\ 1]$ lighting respectively. (b,d) are the corresponding histograms of the orientation error.

proposed approach against the variations in the illumination conditions, we synthesized new images from the database subjects illuminated by a single light source coming from the direction $[-1, 0, 1]$. The approach is applied on the synthetic images and the histograms of the height and the orientation errors are computed (see Fig 2(c,d)). As it is clear from the four histograms in Fig. 2, the proposed approach succeeds to recover the face 3D shape and achieves high accuracy in terms of surface height and surface orientation.

Figure 3 shows the reconstruction results of four out-of-training-sample examples illuminated by different lighting conditions. In the first four columns of the figure, the faces are illuminated from the direction $[0, 0, 1]$ while the faces in the last four columns are illuminated from the direction $[-1, 0, 1]$. The input images are shown in the first row of the figure, while the recovered shape obtained from the combined model is shown in the third row. The second row shows the recovered best-fit SHP image. For the sake of visual comparison, the profiles of the recovered and the ground truth surfaces are plotted in the fourth row. As the figure illustrates, for all cases, the recovered SHP images are very close to the input images which consequently leads to good accuracy in the surface recovery. These results show the major potential of the proposed approach which resides in the using of the spherical harmonics illumination representation which can naturally handle the variation in the lighting conditions.

The last two rows of Fig. 3 show the reconstruction results obtained from the coupled model presented in [13]. As expected, For the frontal lighting case (columns a,b,c and d in Fig. 3), the intensity images and the surfaces are recovered with high accuracy. Also, the difference between these results and the results of the proposed model is very small since the intensity information and the SHP information are almost the same. In contrary, the coupled model in [13] fails to recover the correct shape for the faces illuminated from the direction $[-1, 0, 1]$ as shown in columns (e,f,g and h) of the same figure. For quantitative comparison, Table. 1 gives the values of the height and the orientation error.

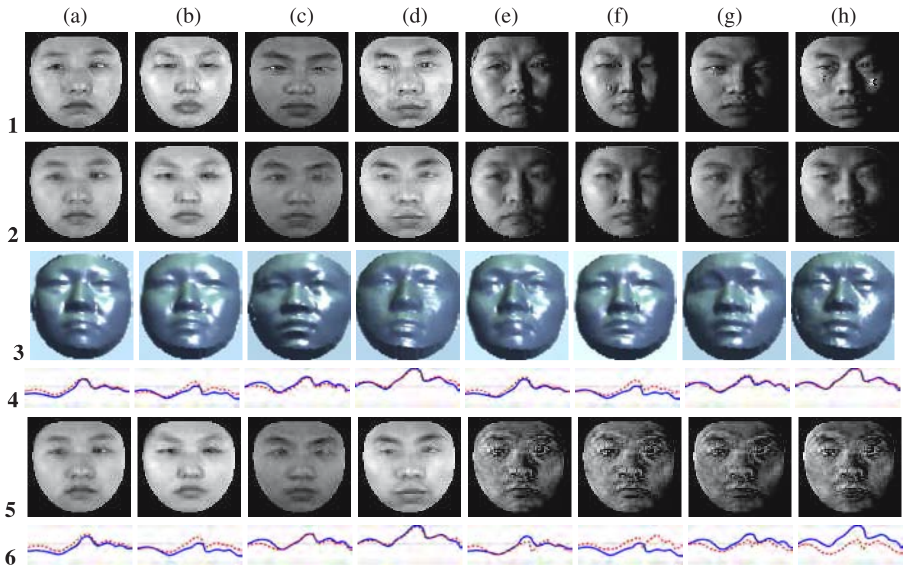


Fig. 3. The reconstruction results for four examples illuminated by different lighting conditions: faces in columns (a,b,c,d) are illuminated by frontal light while faces in columns (e,f,g,h) are illuminated by light coming from the direction $[-1,\ 0,\ 1]$. The input images are shown in the 1^{st} row. The recovered SHP images and recovered shapes are shown in the 2^{nd} and 3^{rd} rows respectively. The 4^{th} row shows the profiles of the ground truth shapes in solid blue and the recovered shapes in dashed red. The recovered intensity and the profiles of the recovered shapes obtained using the model [13] are shown in the 5^{th} and the 6^{th} rows.

Experiment#2. In this experiment we applied the proposed approach to recover the 3D shape of six images form the Yale Face Database B [21]. Three of these images were taken under frontal lighting while the remaining were taken under various illuminations. The input images are shown in the first row of Fig. 4 while the recovered best-fit SHP images and the recovered shapes are shown in the second and the third rows respectively. Through the visual comparison, we can see that the difference in the appearance between the input images and recovered SHP images is acceptable which leads to relatively good shape reconstruction.

Table 1. The error measures for the results shown in Fig 3

	methods	a	b	c	d	e	f	g	h
height error (%)	[13]	1.9	4.0	2.0	1.4	2.6	4.3	3.6	4.4
	proposed	1.6	3.0	2.0	1.8	1.5	3.7	1.8	1.6
orientation error (in rad)	[13]	0.15	0.14	0.16	0.15	0.33	0.32	0.30	0.34
	proposed	0.15	0.14	0.18	0.15	0.15	0.14	0.16	0.16

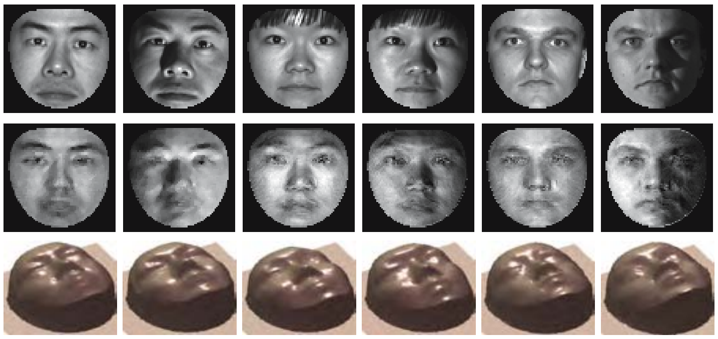


Fig. 4. The reconstruction results for six images from Yale Database B. The 1^{st} row shows the input images taken under different illumination conditions. The best-fit SHP images are shown in the 2^{nd} row. The recovered shapes are shown in the 3^{rd} row.

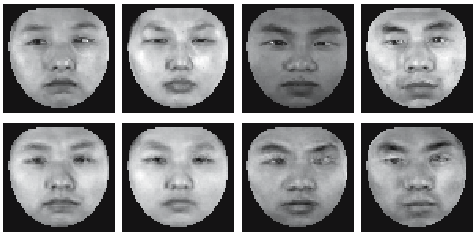


Fig. 5. Gray-level albedo recovery. The ground truth albedo images are shown in the first row. The recovered albedo images are shown in the second row.

Table 2. The values of the estimated light directions

	exact direction	estimated direction	difference angle (in rad)
(a)	[0.5 0.5 1.0]	[0.41 0.33 1.0]	0.14
(b)	[-1.0 0.0 1.0]	[-0.96 -0.01 1.0]	0.02
(c)	[0.0 -1.0 1.0]	[-0.08 -0.96 1.0]	0.06
(d)	[-1.0 0.0 1.0]	[-0.92 0.12 1.0]	0.09

Experiment#3. In this experiment we use Eq. 18 to recover the albedo images for four examples illuminated with different lighting (see Table 2). We show the ground truth albedo and the recovered albedos images in Fig. 5. Also, we used the results to estimate the light direction as explained in Section 5.2. Table 2 gives the values of the estimated light directions for the four examples.

The accuracy of the recovered albedos and the small difference between the estimated light direction and the exact direction, emphasize on the high precision the whole recovery process.

7 Conclusion

A new statistical model is presented that combines 3D shape and intensity appearance of human faces. Not only can the model be used to recover the face shape, but also, it can recover the gray-level albedo and it can estimate the light direction. The model makes use of the powerful spherical harmonics illumination representation to handle different variations in the lighting conditions. The model has been reconstructed from a large training set of human 3D faces and their albedo images. The performance of the proposed model in 3D shape recovery is evaluated using images taken under different illumination conditions and it achieves promising results. It takes about 15 second to recover the shape and the albedo from a single image.

References

1. Blanz, V., Vetter, T.: Face recognition based on fitting a 3d morphable model. IEEE Trans. Pattern Anal. Mach. Intell. 25, 1063–1074 (2003)
2. Zhang, L., Wang, S., Samaras, D.: Face synthesis and recognition from a single image under arbitrary unknown lighting using a spherical harmonic basis morphable model. In: CVPR 2005. International Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA (2005)
3. Lee, W.-S., Magnenat-Thalmann, N.: Fast head modeling for animation. Image and Vision Computing 18, 355–364 (2000)
4. Basso, C., Vetter, T.: Statistically motivated 3d faces reconstruction. In: The 2nd International Conference on Reconstruction of Soft Facial Parts, Remagen, Germany (2005)
5. Devernay, F., Faugeras, O.: Computing differential properties of 3-d shapes from stereoscopic images without 3-d models. In: International Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA (1994)
6. Zeng, G., Paris, S., Lhuillier, M., Quan, L.: Study of volumetric methods for face reconstruction. In: Proceedings of IEEE Intelligent Automation Conference, IEEE Computer Society Press, Los Alamitos (2003)
7. Zhao, W.Y., Chellappa, R.: Symmetric shape-from-shading using self-ratio image. Int. J. Comput. Vision 45, 55–75 (2001)
8. Ypsilos, I.A., Hilton, A., Rowe, S.: Video-rate capture of dynamic face shape and appearance. In: FGR 2004. International Conference on Automatic Face and Gesture Recognition, Seoul, Korea (2004)
9. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: SIGGRAPH 1999. Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 187–194 (1999)
10. Dimitrijevic, M., Ilic, S., Fua, P.: Accurate face models from uncalibrated and ill-lit video sequences. In: CVPR 2004. International Conference on Computer Vision and Pattern Recognition, Washington, DC, USA (2004)
11. Smith, W.A.P., Hancock, E.R.: Recovering facial shape using a statistical model of surface normal direction. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1914–1930 (2006)

12. Cootes, T., Edwards, G., Taylor, C.: Active appearance models. In: Burkhardt, H., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1407, pp. 484–498. Springer, Heidelberg (1998)
13. Castelan, M., Smith, W.A.P., Hancock, E.R.: A coupled statistical model for face shape recovery from brightness images. IEEE Trans. Image Processing 16, 1139–1151 (2007)
14. Basri, R., Jacobs, D.W.: Lambertian reflectances and linear subspaces. In: International Conference on Computer Vision (2001)
15. Basri, R., Jacobs, D.W.: Lambertian reflectance and linear subspaces. IEEE Trans. Pattern Anal. Mach. Intell. 25, 218–233 (2003)
16. Ramamoorthi, R., Hanrahan, P.: On the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. Journal of the Optical Society of America (JOSA A) 18, 2448–2459 (2001)
17. Ramamoorthi, R., Hanrahan, P.: A signal-processing framework for inverse rendering. In: SIGGRAPH 2001. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 117–128 (2001)
18. Sirovich, L.: Turbulence and the dynamics of coherent structures. Quarterly of Applied Mathematics XLV, 561–590 (1987)
19. The BJUT-3d large-scale chinese face database, `http://www.bjut.edu.cn/sci/multimedia/mul-lab/3dface/face_database.htm`
20. Cyberware, inc., monterey, ca. `http://www.cyberware.com/`
21. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Trans. Pattern Anal. Mach. Intelligence 23, 643–660 (2001)

SketchSurfaces: Sketch-Line Initialized Deformable Surfaces for Efficient and Controllable Interactive 3D Medical Image Segmentation

Meisam Aliroteh and Tim McInerney

Depts. of Computer Science and Electrical Engineering, Ryerson University, Toronto, ON, Canada, M5B 2K3

Abstract. We present an intuitive, fast and accurate interactive segmentation method for extracting and visualizing a large range of objects from 3D medical images. Our method combines a general deformable subdivision-surface model with a novel sketch-line user initialization process. The model is simply and precisely initialized with a few quick sketch lines drawn across the width of the target object on several key slices of the volume image. The smooth surface constructed using these lines is extremely close to the shape of the object boundary, making the model's task of snapping to this boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. Our subdivision surface model provides a foundation for precise user steering/editing capabilities and simple, intuitive user interactions are seamlessly integrated with advanced visualization capabilities. We use our model to segment objects from several 3D medical images to demonstrate its effectiveness.

1 Introduction

Segmentation of 3D and 4D images remains one of the major challenges in medical image visualization and analysis. Currently, semiautomatic methods are potentially the most effective approach since they combine the efficiency aspects of automatic segmentation with human expert guaranteed robustness. However, despite the large number of 3D semiautomatic segmentation methods developed over the past decade, no one technique has yet been universally adopted in clinical practice; as a result, manual delineation or simple pixel-based tools are still heavily used. It is our contention that the lack of adoption of these techniques is in part related to the incomplete integration of simple, intuitive, and consistent interaction capabilities, with the necessary precision and power, into the segmentation work-flow.

Semi-automatic 3D segmentation techniques can fail in noisy images and this failure often results in time-consuming and tedious user intervention. With many of the current techniques [1], [2], [3], [5], [6], once the algorithm is initiated the ability to steer it is limited and/or the ability to edit it is restricted to a separate post processing phase, often with a separate tool-set and/or user actions. That is, while the mathematical formulations and numerical algorithms of these methods enable simple initialization and complex shape extraction, the cost may be incurred on the "back-end" of the segmentation process. Other techniques [7], [8] do provide interactive steering capabilities but editing facilities are not well integrated, the required tracing actions can be tedious, and the 2D nature of the underlying algorithm often requires considerable user concentration for a 3D data set.

The deficiencies of user control functionality, simplicity of use, and the lack of interaction precision and intuitiveness of current methods have led us to explore an alternative research direction in 2D [9], creating interactive active contour models suitable for segmenting medical images which exhibit a significant amount of noise. In this paper, we extend this idea to 3D segmentation. More specifically, our goals are to provide a highly accurate initialization of an active surface model using fast, simple, user actions that require little concentration. After initialization, it should be visually apparent to the user what the resulting segmentation will look like. In addition, we seek a robust, noise-insensitive model that provides the foundation for precise, efficient, and intuitive steering and editing capabilities, and a method that requires little or no parameter tweaking and/or mode changes. Most importantly, all the interaction capabilities must be seamlessly "woven" together providing simple, consistent interactions and complete user control throughout the segmentation process.

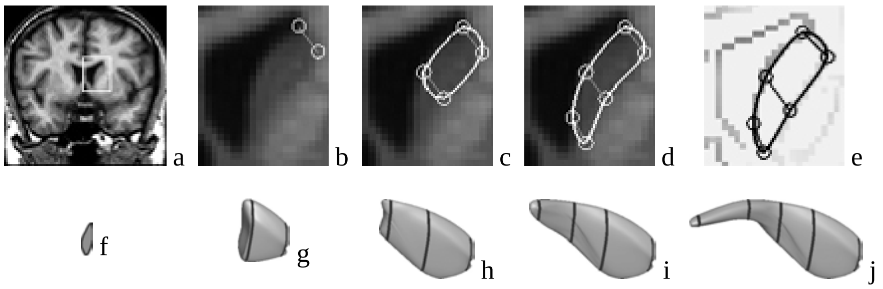


Fig. 1. SketchSurfaces segmentation of the right caudate nucleus (CN) from an MR volume image. The user positions and orients an image slice plane (a), and then sketches a few lines across the width of the CN, creating a contour that approximates the CN boundary (b-d). The user may also use an edge detected view (e). The user then proceeds to another slice plane and repeats this sketching process. After each sketch process, the previous contours are automatically connected and converted into a subdivision surface (f – j). The dark curves in f – j highlight the sketch plane locations.

To achieve our goals, we present *SketchSurfaces* - a highly controllable active surface model with a simple formulation that is coupled with a novel sketch-based 3D initialization method (Fig. 1). SketchSurfaces offer easy, consistent transitions between initializing, fitting, steering, editing, visualizing, and zooming. The user is able to stop the segmentation at any time, examine the current initial surface model or partial segmentation result, zoom in and out, and make precise corrections before continuing, all with simple mouse or stylus actions. In the remainder of this paper, we describe SketchSurfaces and its interaction model in more detail. We apply it to several MRI brain data sets to demonstrate its effectiveness and make a brief comparison of its efficiency and accuracy with other techniques.

1.1 Background

Over the past decade or more, several (model-based) 3D interactive segmentation techniques have been developed. For many of the methods, the research focus was on the model itself and the interaction facilities were added later. One popular deformable surface model technique is commonly known as a deformable "balloon" model [1], [2], [3], [5], [6]. These models can "inflate" and take on the shape of the target object and some are also able to dynamically change their topology [1], [2], [5]. The inflation forces significantly increase their shape capture range and they can be simply initialized using, for example, one mouse click to create a small spherical seed model inside the target object. Balloon models work best when the image feature map is relatively clean and homogeneous. However, clinical images are noisy, contain many uninteresting edges and regions of low contrast, contain gaps in the object boundary, or exhibit a complex texture. Hence, these more automatic techniques may not generate the expected result - the added automation does not come without a cost. Some interactive control (steering capability) over the model is lost and the gaps in the object boundaries may allow the model to leak through, requiring user intervention. Another common problem is the balloon may not completely inflate into all regions of the object and mechanisms must be provided so that the user can force the model into these regions. Finally, editing in these techniques is typically performed in a post processing phase, often using a separate tool-set. More recently, researchers have developed semiautomatic volume/region painting methods [10], [11], [12], [13]. Here the user gives loose hints to the algorithm by painting rough strokes or seeds on the object and several strokes on the background. An optimization algorithm uses these inputs hints to extract the actual object boundary. In noisy images, these “Graph Cut” techniques can share similar problems with the balloon models and the user needs to provide more foreground/background strokes or edit with a separate tool-set. On the other hand, both the balloon and Graph Cut methods may be used to segment very complex-shaped objects, such as the brain cortical surface. SketchSurfaces would currently require too much user input to efficiently extract such objects.

Another popular technique is the extension of LiveWire to 3D [7], [8]. "LiveWire" or "Intelligent Scissors" [14], [15], is an interactive 2D boundary tracing tool which provides the user with the ability to steer the segmentation. Although faster and more reproducible than manual tracing, these techniques can still demand a large amount of concentration from the user, especially in noisy 3D images. There is never a perfect match between the features used by the LiveWire algorithm and the desired object boundary. As a result, the user often must control the mouse carefully. If a mistake is made, the user’s only option is to "backtrack" and try again. In addition, tracing around the entire object using a mouse can be fatiguing. Finally, 3D LiveWire is fundamentally an extended 2D technique – when segmenting a 3D image, the 2D contour-based nature of the algorithm forces the user to pay careful attention to how it is applied and to mentally reconstruct the 3D shape of the object.

2 SketchSurfaces

Our method uses an active surface model and is 3D by nature. It is constructed using a subdivision surface [16], providing the robustness against noise of a finite-element based model, the broadest possible shape coverage, a foundation for simple and precise editing, inherent smoothness, and a natural hierarchical organization.

We have carefully integrated a novel sketch line initialization process that allows the subdivision surface to be quickly initialized with a few rough sketch lines drawn across the width of the target object in several key image slices. The result is an initial model that is extremely close in shape to the target object, making the deformable surface model's task of snapping to the object boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. It also minimizes the effect of the model's elasticity parameters. Furthermore, it allows the model to primarily rely on image edge information and (optionally) expected edge transitions (e.g. bright to dark) so that it is generally applicable to many common imaging modalities without the need for modality-specific parameters. Finally, no tedious tracing actions are needed - only simple short sketch lines on a few slices are required. Steering and editing the model can take place at any time during the segmentation process and anywhere on the model. In other words, we have attempted to design the method with the 3D user interaction model as our focus. In the following sections, we describe the construction of our subdivision surface model and our interaction model.

2.1 Subdivision Surfaces

The success of interactive shape-model based segmentation techniques, such as active contours/surfaces, is still heavily dependent upon the generality, controllability, and simplicity of the underlying shape representation. We propose the use of subdivision surfaces (and curves in 2D) [16] as a very general shape representation for deformable models. The underlying idea behind subdivision methods [16] is very simple - using geometric algorithms to progressively subdivide a control polygon (or control mesh in 3D). Repeated subdivision leads to a hierarchy of increasingly refined models which approach a smooth *limit curve* (*limit surface* in 3D). The control mesh vertices (the control points) provide an intuitive and precise mechanism to control the shape of the limit surface. As the control points are repositioned, a new limit surface is computed in real time, allowing the user to precisely deform it (Fig. 2).

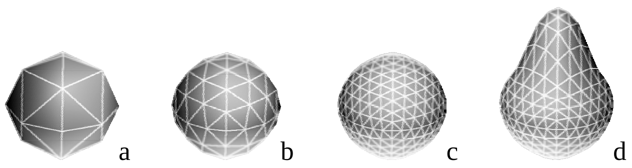


Fig. 2. Example of a subdivision surface and the subdivision process: (a) control mesh, (b) one level of subdivision, (c) two levels, and (d) modified surface after moving a control point

2.2 Constructing a Deformable Subdivision-Surface Model

To construct a deformable surface model using a subdivision surface, we use the vertices of the coarsest level of the surface (i.e. the control points) as the degrees of freedom (d.o.f.), and the finest level vertices as "sensors". The idea is to use a sufficient number of control points and a sufficient level of subdivision such that there is roughly one sensor point for each boundary voxel of the target object, making maximal use of all available image information. External image forces are computed at these sensor points and then (using the weights of the subdivision mask of the points) distributed among the control points (i.e. a control point that is closer to a particular sensor point will receive proportionately more of the computed force than a control point further away). The deformable subdivision surface model is then formulated numerically using the simple explicit scheme described in detail in [5] and the positions of the control points are updated. New external forces are calculated and the process is repeated until an accurate solution is reached.

Our subdivision surface shares similarities with the model developed in [4]. However, their model was developed as a standard deformable balloon and uses a complex finite-element formulation. We use a simple finite difference formulation (although with a newer, more stable numerical update procedure). In addition, rather than the traditional image gradient magnitude external forces, we use spring forces between a sensor point and an edge voxel as our external forces. For each sensor point on the model, a search along its normal is carried out for a small, user-specified distance (typically only two or three voxels). If a matching edge voxel is found, a spring force is applied to attract the sensor point to it. If no matching edge is found (due to boundary gaps for example), this sensor point does not contribute to the image forces. This type of external force produces very predictable and stable behavior, with no "leaking" issues commonly associated with balloon inflation forces.

3 SketchSurfaces Interaction

SketchSurfaces presents two windows to the user. One window is used as a contextual view and consists of a 3D image slice plane that can be oriented as the user chooses (Fig. 1a). The user can intuitively push or pull on this plane to navigate through the volume and the display of the 3D deformable surface model is controlled with a key. The other window presents the same slice plane in 2D (i.e. looking down at the plane in the negative direction of its normal) and allows the user to focus on a slice of the target object with the cross-section of the surface model overlaid (Fig. 1b). Without this 2D view, the user will encounter situations where they would have to tediously rotate the entire scene in the 3D window to bring the content of the 3D slice plane into view. The user can initialize, fit, edit, steer, or zoom in on the model in either window.

3.1 Initialization

A critical and often overlooked phase of the interactive shape-model based segmentation process is the initialization of the model. Explicit, parametric deformable models such as SketchSurfaces typically minimize energy using local optimization techniques. Consequently, efficient convergence to the desired segmentation result is heavily dependent upon careful initial model placement.

As discussed in Section 1.1, the most common method of initializing a deformable surface model is to create a small spherical model from a user-defined seed point. Inflation forces are then used to drive the model towards the object boundary. A potential problem with this approach is the model needs to inflate a considerable distance to reach the object boundary so it is not immediately visually apparent whether the segmentation will succeed. This approach creates model steering issues as well as a separation between the inflation phase of the segmentation and the editing phase.

User fatigue is an important consideration in any interactive design and analysis task. For this reason, sketching actions are being actively researched for many of these tasks in an effort to reduce fatigue and user concentration and to "amplify" user input actions, thereby maximizing productivity. In the context of interactive segmentation, the idea behind sketching is to allow the user to provide an accurate initialization for the surface model - which minimizes subsequent steering and editing - with a low degree of concentration and simple mouse movements. Our sketch-line initialization process [9] is a simple but effective technique that realizes this idea (Fig. 3). It does not require tedious tracing actions and can be performed with minimal concentration; short sketch lines are quick, easy and comfortable to draw.

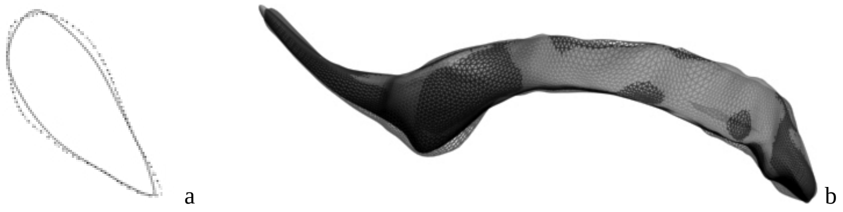


Fig. 3. Result of the sketch line initialization process for the right brain ventricle. By quickly sketching a few lines in several slices of an MR volume image, a highly-accurate initial surface is created. (a) cross-sectional contour of the initial model (solid curve) shown with a cross-section of the manually segmented ventricle (dotted curve). (b) initial surface model (light semi-transparent) shown with the manually segmented ventricle surface (dark wireframe).

The user begins the initialization process by first positioning the image slice plane at the far end of the target object and then sketching a few short lines across the width of the object (Fig. 1, b-d) on this plane (which we term a *sketch plane*). Drawing these lines roughly perpendicular to the object boundary will result in an initial surface model that is approximately locally aligned with the boundary. The user then pulls the image slice plane forward and again sketches a few lines across the object. A subdivision surface model is progressively constructed in real-time from these sketch lines using the quick-hull algorithm and the subdivision process (Fig. 1, f – j), and the cross-section of the model is instantly displayed on the image slice. In this way, the user is given immediate visual feedback of the surface model construction and can observe the initialization accuracy. The user repeats this pulling and sketching process several times until a sketch plane is close to the near-end of the object.

The user can view the 3D surface model as it is constructed and make corrections if desired by repositioning control points and/or by adding new sketch lines. Additional sketch planes can also be inserted. The number of sketch planes required in order to generate an accurate initial surface model is dependent upon the shape of the target object. For example, in our experiments with the caudate nucleus, we typically used five sketch planes, with one close to each end of the caudate nucleus and three evenly distributed in between. Furthermore, the number of sketch lines required on each sketch plane is dependent on the complexity of the shape of target object cross-section (usually only two or three sketch lines are required for the caudate nucleus).

The user typically performs a few trials, with a different number and position of sketch planes, until an accurate initial surface model is obtained. We have found that it is best to begin with two/three sketch planes (one close to the each end of the target object and optionally, one in the center), examine the initial surface, fit the model and examine the segmentation result, and then add intermediate sketch planes until the desired accuracy is obtained. This pattern of sketch planes and sketch lines can then be used when segmenting the target object in all subsequent volumes. Note that the pattern of sketch planes is only a rough guide – the algorithm is fairly robust to small differences in the number/positions of the sketch planes. For large differences accuracy is impacted, resulting in increased user editing.

3.2 Fitting, Steering, and Editing

Editing a segmentation result typically implies making corrections to the segmentation after the algorithm has run to completion. Steering, on the other hand, implies guiding the segmentation process toward the correct result while the algorithm is running. The lines between fitting, steering, and editing are blurred when using SketchSurfaces. Once the model has been initialized, the user presses a button to begin the fitting step and the surface quickly *snaps* to the object boundary (typically within two or three seconds). We terminate the fitting step after a (user-adjustable) number of iterations. Because the initialization process creates a surface model very close to its final position, one fitting step is often sufficient to generate an accurate segmentation. Nevertheless, the user may repeat the fitting step as often as desired.

Before, between, or after these fitting steps, SketchSurfaces can be precisely and intuitively controlled using various geometric editing actions. The initialization, fitting, editing, steering, and zooming are all performed within a single process, using only simple actions. The user is not forced to switch modes or select action items from a menu/panel. Examples of these editing actions are listed below:

1. *Before fitting*: The user can edit the initial surface by modifying the sketch plane contours. Actions include adding new sketch planes, repositioning the existing control points and sketch lines, and adding new control points by drawing new sketch lines or by breaking an existing sketch line into two pieces.
2. *After/between fitting steps*: Simple, precise control over the surface model position and shape is performed in 2D on an (2D/3D) image slice plane. Constraining user interactions to 2D significantly simplifies them, improves accuracy and efficiency, maintains user familiarity, and does not require a special input device.

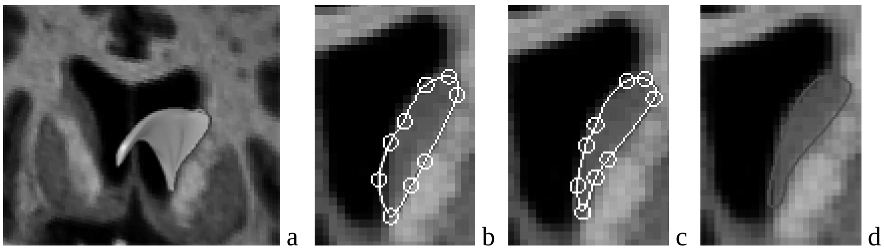


Fig. 4. Editing in SketchSurfaces. (a) the user manipulates the 3D image slice plane and generates a cross-section of the surface model. (b) a subdivision curve is automatically constructed from this cross-section. (c) the user precisely reshapes the curve by dragging curve control points. (d) constraint forces are generated and the surface model is refitted such that the new cross-section matches the subdivision curve.

The user manipulates the image slice plane in either the 3D or 2D window and cuts the surface model in any desired orientation to generate a 2D subdivision curve (Fig. 4b). The control points of this curve are automatically created from the cross-section of the surface model's control mesh and the resulting curve shape closely matches the shape of the subdivision surface cross-section. Because the initialization and fitting of the surface model provides an accurate segmentation, the control points of the subdivision curve are often in a "good" position and fine-tuning can be performed simply by nudging these points, causing a section of the curve to be dragged into the desired position (Fig 4c). We then construct “soft” constraint forces between the sensor points on the surface cross-section and corresponding points along the newly deformed subdivision curve. A quick refitting (snapping) of the surface model is performed to pull it towards the subdivision curve via the soft constraints (Fig. 4d). This process allows any number of arbitrarily oriented constraint forces to be added to the surface model and also takes advantage of the powerful editing capabilities of subdivision curves, avoiding imprecise and tedious editing using manual tracing. Curve control points can be fluidly dragged and precisely positioned and new control points can be added.

3. *Snipping*: Occasionally the initialization process may create a surface region that protrudes past the end of an object. We provide the ability to snip away this protrusion by simply positioning the 3D image slice plane such that the protrusion is on one side. The user simply presses a key to cutout this excess portion.
4. *Pinning/Unpinning*: The user can pin/unpin any control point of the surface model by simply double-clicking on a point or an edge joining two control points. A pinned control point will remain frozen regardless of any forces that are acting on it. This is a useful feature for segmenting objects in regions with no (or very weak) edges and can be performed at any time (typically during sketch line initialization). The user takes more care to position and pin a point in its "final" position.
5. *Local subdivision*: New control points and sensor points can be added to the subdivision surface model in a *local* region where editing is occurring. Currently, this is explicitly initiated by the user.

3.3 SketchSurfaces Parameters

The goal of SketchSurfaces is the creation of a segmentation tool that requires no parameter tweaking or mode changing. Our initialization process and subdivision surface model goes a long way towards achieving this goal - default parameter settings are often adequate. Nevertheless, the user is able to set several parameters:

1. *Edge intensity control*: Maximum and minimum edge intensity threshold can be set and the model will search for edge intensities within this range. Canny edges are used as they are visually simpler to "see".
2. *Automatic insertion of internal control points*: The number of control points automatically added between the user-defined control points can be controlled. The default value is set to one and we have used this value in all of our experiments.
3. *Image edge search range*: Edge voxel search range can be controlled in both the positive and negative sensor point normal direction.
4. *Edge transition control*: This parameter can be used to activate a search for edges with either a bright-to-dark transition or a dark-to-bright transition, allowing the model to ignore edges with incorrect transitions thereby improving robustness.

4 Experimental Results and Discussion

We have successfully tested our approach on five MR brain data sets obtained from The Internet Brain Segmentation Repository (http://www.cma.mgh.harvard.edu/ibsr/). The data sets were interpolated to obtain cubical voxels, with interpolated dimensions of 256 x 256 x 205 and 256 x 256 x 192. We have tested our application on a PC with a CPU clock speed of 3.2 Ghz. We segmented three different objects and used the hand-segmented volumes available from the repository to calculate the accuracy of our model. The accuracy was measured using 2-sided Hausdorff distancing, which is a more strict measure of accuracy than measures such as volume overlap.

Table 1 and Figure 5 illustrate the results obtained from segmenting the right caudate nucleus from the five data sets. The same parameters have been used for all of the segmentations and the same number of sketch planes was used to construct the initial surface model. Table 1 also demonstrates a significant increase in the speed of the segmentation when compared to [1] and [8]. On average, we performed the initialization and fitting in about 36 seconds plus an additional 29 seconds to edit a few slices, resulting in a total segmentation time of about 65 seconds and sub-voxel accuracy. This can be roughly compared to 2.5 minutes for a well-known balloon method [1] and 3.5 minutes for a 3D LiveWire technique [8].

Table 2 and Figure 6 illustrate the results obtained from segmenting the right ventricle from the five data sets. Once again, the same parameters have been used for all segmentations. Moreover, with the exception of experiment 4, approximately the same numbers of sketch planes and control points have been used. Due to a significant difference in the shape of the ventricle in the data set for experiment 4, an extra sketch plane was inserted. On average, we performed the initialization and fitting in 41 seconds plus an additional 9 seconds to edit one slice, resulting in a total segmentation time of about 50 seconds with sub-voxel accuracy. Again, this result can be roughly compared to 7.5 mins. for a balloon [1] and 5.5 mins. for 3D LiveWire [8].

Table 1. Results of segmenting the right caudate nucleus from five data sets

Exp	# of Sketch Planes	# of Control / Sensor Points	Seg. Time (sec)	Editing Time (sec)	Hausdorff dist. stat. between the extracted and expert seg. surfaces		
					Avg. Dist.	Max Dist.	# of (% of) sensor points with sub-voxel accuracy
1	5	79 / 1219	33	20	0.308	1.81	1172 (96.1%)
2	5	87 / 1343	35	42	0.481	2.25	1224 (91.1%)
3	5	92 / 1412	37	24	0.444	2.11	1294 (91.6%)
4	5	87 / 1347	36	36	0.422	1.93	1257 (93.3%)
5	5	84 / 1284	37	24	0.373	1.36	1243 (96.8%)

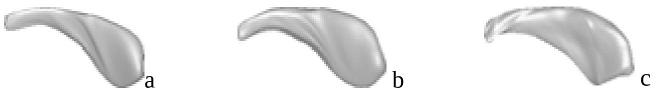


Fig. 5. Example segmentation result for the right caudate nucleus from MR volume image. (a,b) initial and fitted subdivision surface and (c) hand-segmented surface.

Table 2. Results of segmenting the right ventricle from five data sets

Exp	# of Sketch Planes	# of Control / Sensor Points	Seg. Time (sec)	Editing Time (sec)	Hausdorff dist. stat. between the extracted and expert seg. Surfaces		
					Avg. Dist.	Max Dist.	# of (% of) sensor points with sub-voxel accuracy
1	7	132 / 2052	40	14	0.460	2.07	1905 (92.8%)
2	8	153 / 2373	39	8	0.393	1.83	2290 (96.5%)
3	8	150 / 2310	41	7	0.412	2.43	2160 (93.5%)
4	9	169 / 2629	43	12	0.336	2.15	2500 (95.1%)
5	8	153 / 2373	42	5	0.357	1.62	2297 (96.8%)

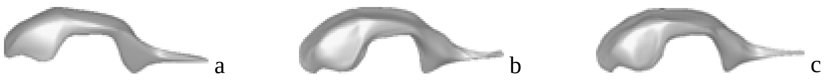


Fig. 6. Example segmentation result for the right brain ventricle. (a, b) initial and fitted subdivision surface model and (c) the hand-segmented ventricle surface.

Finally, we also segmented the right putamen from one volume image. The shape of the putamen is more complex than that of the caudate, requiring 6 sketch planes instead of 5 to construct the initial surface model and one or two more sketch lines per sketch plane were required to create accurate contours. In addition, the putamen is a very noisy object containing large gaps in the edge detected image and the user must interpret the location of the boundary. These factors demand precise segmentation control. From our experiments, we observed that steering the model by pinning control points in these regions requires only a very slight increase in effort but realizes considerable gains in efficiency. SketchSurfaces initialization and fitting were performed in 65 seconds for the putamen plus an additional 8 seconds to edit one slice, resulting in a segmentation time of about 73 seconds (avg. distance 0.484 voxels, max. dist. 2.07 voxels, 90.3% of sensor points with sub-voxel accuracy).

5 Conclusion

Optimizing the performance of 3D semiautomatic medical image segmentation methods for noisy volume images requires a minimal number of fast, simple and fatigue-free interactions, intuitive, flexible, and precise user steering and editing capabilities. Furthermore, all of these capabilities must be seamlessly integrated into the segmentation work-flow so that they are available at any time and presented to the user using a consistent interface. We believe the combination of sketching input lines across an object, a powerful subdivision-surface based deformable model, with subdivision curve editing flexibly derived from arbitrarily oriented cross-sections of this model, results in a tool that meets these requirements and is effective for many segmentation tasks that may not be processed as efficiently with other techniques. Currently our method is optimized for simple to moderately complex-shaped objects. We are extending the method so that objects with significant protrusions and/ holes may be segmented. For very complicated objects (e.g. brain cortical surface) our method may require too much user input and we are exploring the idea of merging SketchSurfaces with a balloon model. We are also investigating techniques to automate sketch plane placement, allowing even more efficient and simple volume image segmentation.

References

1. Yushkevich, P., Piven, J., Cody Hazlett, H., Gimpel Smith, R., Ho, S., Gee, J., Gerig, G.: User-guided 3D active contour segmentation of anatomical structures: significantly improved efficiency and reliability. NeuroImage 31(3), 1116–1128 (2006)
2. Pons, J.P., Boissonnat, J.D.: Delaunay deformable models: Topology-adaptive meshes based on the restricted Delaunay triangulation. In: IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, USA (June 2007)
3. Montagnat, J., Delingette, H.: Spatial and temporal shape constrained deformable surfaces for 3D and 4D medical image segmentation, Technical report RR-4078, INRIA (2000)
4. Mandal, C., Qin, H., Vemuri, B.C.: Dynamic Modeling of Butterfly Subdivision Surfaces. IEEE Trans. on Visualization and Computer Graphics 6(3), 265–287 (2000)
5. McInerney, T., Terzopoulos, D.: Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation. IEEE Trans. on Med. Imaging 18(10), 840–850 (1999)
6. Bredno, J., Lehmann, T.M., Spitzer, K.: A General Discrete Contour Model in Two, Three, and Four Dimensions for Topology-Adaptive Multichannel Segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence 25(5), 550–563 (2003)

7. Falcão, A.X., Udupa, J.K.: A 3D Generalization of User-Steered Live Wire Segmentation. Medical Image Analysis 4, 389–402 (2000)
8. Poon, K., Hamarneh, G., Abugharbieh, R.: Segmentation of Complex Objects with Non-Spherical Topologies from Volumetric Medical Images using 3D Livewire. SPIE Medical Imaging 6512(31), 1–10 (2007)
9. McInerney, T., Akhavan Sharif, M.R.: Sketch Initialized Snakes for Rapid, Accurate, and Repeatable Interactive Medical Image Segmentation. In: 3rd Int. Symposium on Biomedical Imaging, Arlington, Virginia, pp. 398–401 (April 2006)
10. Boykov, Y.Y., Funka-Lea, G.: Graph Cuts and Efficient N-D Image Segmentation. International Journal of Computer Vision 70(2), 109–131 (2006)
11. Tzeng, F.Y., Lum, E.B., Ma, K.L.: An Intelligent System Approach to Higher-Dimensional Classification of Volume Data. IEEE Trans. On Visualization and Computer Graphics 11(3), 273–284 (2005)
12. Falcão, A.X., Bergo, F.P.G.: Interactive Volume Segmentation with Differential Image Foresting Transforms. IEEE Trans. on Medical Imaging 23(9), 1100–1108 (2004)
13. Owada, S., Nielsen, F., Igarashi, T.: Volume Catcher. In: Proceedings of the 2005 symposium on Interactive 3D graphics and games, pp. 111–116 (2005)
14. Mortensen, E.N., Barrett, W.A.: Interactive segmentation with intelligent scissors. Graphical Models and Image Processing 60(5), 349–384 (1998)
15. Falcão, A.X., Udupa, J.K., Samarasekera, S., Sharma, S., Hirsch, B.E., de Lotufo, A.R.: User-steered image segmentation paradigms: Live wire and live lane. Graphical Models and Image Processing 60, 233–260 (1998)
16. Zorin, D., Schröder, P., Sweldens, W.: Interpolating Subdivision for Meshes with Arbitrary Topology. In: SIGGRAPH 1996. Computer Graphics Proceedings, pp. 189–192 (1996)

Visualization of Resource Allocation in Large-Scale Mobile Ad Hoc Networks

Alex Fridman, Dan Hennessey, David Breen, Steven Weber, and Moshe Kam

Drexel University
3141 Chestnut Street
Philadelphia, PA 19104
{af59@,dph29@,david@cs.,sweber@ece.,kam@ece.}drexel.edu

Abstract. Resource allocation in ad hoc communication networks is a field of high complexity because of both *i*) the distributed nature of the interactions between the nodes, and *ii*) the large set of control variables for even the most primitive networks. Visual representation of this information across physical space and across layers of the network can greatly benefit the understanding of efficient allocation policies in these complex systems. Yet, very few software packages have been developed to address specifically this task, especially for large scale networks. We develop such a software system, and demonstrate some of its capabilities. The system illustrates that multi-layered visualization of the network state can be an effective tool at making network design decisions even in the face of uncertainty and for a large number of network interactions.

1 Introduction

A mobile ad hoc network is generally unstructured and highly dynamic, which makes the process of its visualization particularly important and challenging. In large-scale networks, the amount of resources allocated in a localized fashion is immense and requires graph drawing tools that optimize the trade-off between information density and visual clarity. We propose an abstraction of mobile ad hoc networks that achieves a reasonable balance between these two objectives. The four layers we define are: the mobility layer, the physical (PHY) layer, the medium access control (MAC) layer, and the transport layer. The mobility layer models the movement kinematics of the nodes as well as their movement plans. The PHY layer models the lowest level of the communication medium which includes simultaneous transmissions, radiation patterns, interference, signal quality, and capacity. The MAC layer models the temporal scheduling of node transmission and the aggregate channel state produced by a sequence of physical states. Finally, the transport layer models the flow of information on the network given the aggregate state computed at the MAC layer.

In this paper we *i*) present a multi-layer mobile ad hoc network model, and *ii*) discuss how each of the layers of that model can be visualized to present the essential information about the allocation of network resources. Although there is both significant flexibility in parameter selection and significant model complexity, the visualization model we propose is sufficiently general to allow useful visual presentation of the impact of parameter selection on network performance.

2 Related Work

The technical literature on the visualization of ad hoc networks is significantly limited. To the best of our knowledge, existing work is restricted to small-scale examples that focus more on the attainment of the data that is visualized and less of the clarity, quality, and richness of information contained in the visual output [1]. Connectivity is a critical metric for performance of ad hoc networks, and is easy to visualize, even as the size of the network scales up. Therefore, most larger scale visual information processing efforts in the networking and visualization literature are focused on connectivity, such as in [2,3,4,5]. On the other hand, tools that allow for the visualization of other aspects of the communication medium can only handle small scale networks, both in terms of running-time complexity and visual complexity; [6,7,8] are popular examples of such tools.

One of the most active areas of research in visualization of networks approaches the problem from the perspective of graph theory. Problems addressed in these efforts are concerned with drawing graphs while minimizing the number of intersecting edges, pruning subgraphs based on predefined priorities, and generally maximizing the amount of visually conveyed information without sacrificing clarity and visual appeal. The popular GraphViz [9] tool applies a wide selection of cutting-edge research from this graph theoretic approach. While many graph simplification and layout algorithms are used in our tool, the focus and novelty of the visualization methods we present are in the effective visual representation of network resources commonly allocated in ad hoc networks.

3 Mobility Layer

3.1 Positions and Obstacles

The mobility layer contains the lowest level of information about the communication network. It specifies the current position, velocity, and acceleration of each node $i = 1, 2, ..., n$ in the network at each snapshot in time. Figure 1(a) shows a snapshot of the node and obstacle positions for a large ad hoc network. Each small circle in the figure designates a node, and each large shaded polygon designates an obstacle. Each of the obstacles define the closed sections of the arena that can not be traversed by any of the nodes in the network. This visual information, in its animated form, can be used to monitor in real-time the changes in the network topology.

Figure 1(b) shows the aggregation ability of our visualization engine. A variety of spatial metrics can be gathered and plotted as a contour map, or in 3D with the z-axis containing the value of the metric. In this case, we use a contour map to show the visitation frequency of the nodes to each part of the network arena.

Light and dark areas reflect frequent and infrequent visitation, respectively. The black areas correspond to the obstacle locations. For simplicity, we used a random walk mobility model. This aggregate visual information can be used to study the movement patterns of a chosen mobility model.

A large-scale network of $10,000$ nodes is shown in Figure 1(c). Specifically, the figure displays four snapshots of this network in time. The first snapshot is at the start of a simulation where the node topology is uniformly distributed over the (obstacle-free) arena. The next three snapshots capture the changing mobility graph state as the simulation progresses. The mobility model here is "follow the leader", where ten of the nodes are chosen as leaders, and the rest of the nodes are followers. A follower constantly switches between the state of following a leader and the state of independent movement. The leaders and the followers without a leader perform a random walk. The animated version of what is shown in Figure 1(c) clearly shows that increasing the "following probability" increases the rate of convergence towards a state where small clusters of nodes move around the arena with minimal spreading. If the "following probability" is low enough, the topology dynamics are indistinguishable from one where all nodes perform a random walk.

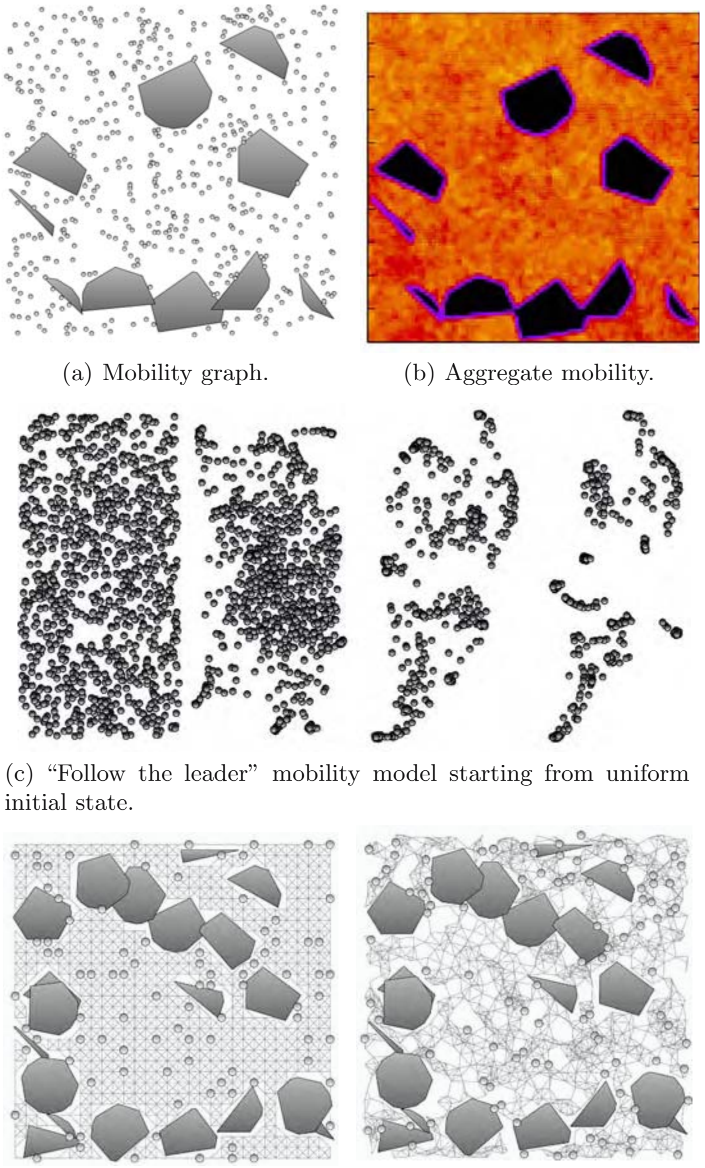
3.2 Kinematics Graph

The kinematics graph $G_K = (V_K, E_K)$ specifies traversability information imposed by the presence of obstacles, boundaries, and kinematic restrictions in an arena. The vertices in V_K each denote a spatial location that can be occupied by a network node. The edges in E_K indicate feasible movements in the arena. Specifically, $\Lambda(i) = \{j : (i, j) \in E_K\}$ is the set of locations directly accessible from location i. Each step has an associated travel time of one time unit.

For the problem of motion planning, the kinematics graph most often takes the structured form of a lattice [10] or the unstructured form of a graph with distance-limited edges [11] generated by randomly sprinkling points V_K in the arena. The visualization of both forms is shown in Figure 1(d) and Figure 1(e), respectively. This visual information is effective for studying the chosen domain of the path planning problem for a discretized space. For example, the figures indicate that the lattice can capture the intricacies of the motion constraints in the physical space with fewer points than required by the unstructured graph, and is therefore a better choice for space discretization.

4 Physical and Medium Access Control Layers

We use the abstraction of the physical (PHY) and medium access control (MAC) layers defined in [12]. Medium access is controlled by defining a transmission schedule, consisting of a sequence of time slots. The schedule specifies a set of concurrent transmissions to occur in each time slot. The concurrent transmissions have the properties that *i*) all transmissions are successful, *ii*) adding an additional transmitter to any time slot would cause one or more transmissions


(a) Mobility graph.

(b) Aggregate mobility.

(c) "Follow the leader" mobility model starting from uniform initial state.

(d) Movement graph (structured).

(e) Movement graph (unstructured).

Fig. 1. Mobility visualization

to fail, and *iii*) the number of time slots is sufficient to ensure that each node is allowed to transmit in at least one time slot per round. The random packing procedure in [12] is employed to identify a random schedule satisfying these properties. The left part of Figure 2 shows the set of concurrent transmissions

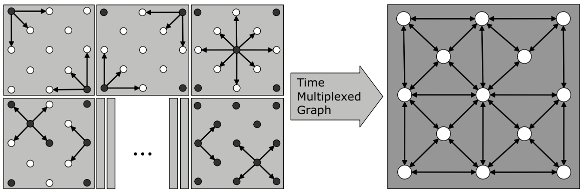


Fig. 2. The process of forming a capacitated graph from a set of schedules

at some of the time slots; the right part of the figure shows the *flow graph* $G_F = (V_F, E_F)$ formed by multiplexing the concurrent transmissions together.

The result of the packing procedure is a collection of graphs

$$\mathbf{G}_P = \{G_P^1, \ldots, G_P^m\},\ G_P^z = (V_P^z, E_P^z),\ z = 1, \ldots, m, \tag{1}$$

where the graphs indicates concurrent transmitter receiver pairs at each time slot in the schedule. Granting preference to nodes that have not yet had the opportunity to transmit in a previous time slot ensures $m \leq n$.

The parameters of the PHY layer are *i*) the fixed transmission power P_i on each node i, and *ii*) the minimum acceptable signal-to-interference-plus-noise (SINR) ratio, γ. Given these values, the random packing procedure adds transmitters to a time slot incrementally such that the SINR requirement, γ, is not violated for any of the links already selected for the time slot. The SINR from transmitter i to receiver j in time slot z is given by:

$$\mathrm{SINR}_{ij}(z) = \frac{P_i A_{ij}}{\sum_{k \in V_P^z \setminus i} P_k A_{kj} + \eta},\ (i,j) \in E_P^z,\ z = 1, \ldots, m. \tag{2}$$

Here, η is the channel noise power. The channel attenuation between transmitter i and receiver j is:

$$A_{ij} = \left(\frac{d_{\mathrm{ref}}}{d_{ij}}\right)^{\alpha}, \tag{3}$$

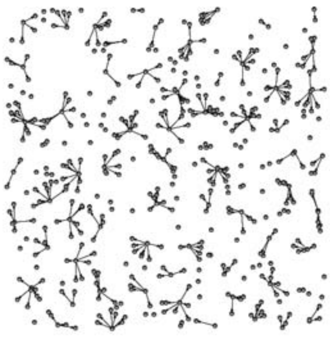
where d_{ij} is the distance between nodes i and j, d_{ref} is a channel reference distance, and α is the attenuation exponent. All edges (i,j) in each graph G_P^z satisfy the requirement that $\mathrm{SINR}_{ij}(z) > \gamma$. The capacity on each edge (i,j) at each time slot, $C_{ij}(z)$, may be assigned to be unity, or may be selected using the Shannon capacity:

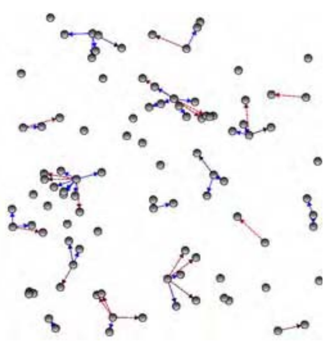
$$C_{ij}(z) = B_{ij} \log_2 \left(1 + \mathrm{SINR}_{ij}(z)\right),\ (i,j) \in E_P^z,\ z = 1, \ldots, m. \tag{4}$$

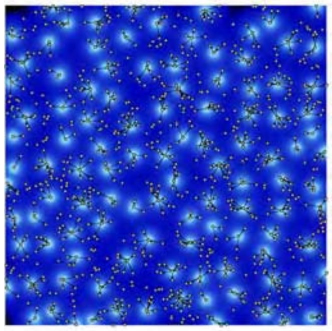
The multiplexing procedure is additive for capacity, that is, the capacity of an edge (i,j) in the flow graph G_F is computed from $\mathbf{G}_P$ by:

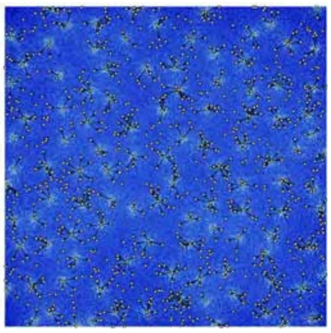
$$C_{ij} = \sum_{z=1}^{m} C_{ij}(z). \tag{5}$$

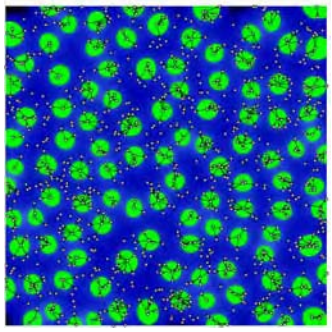
The result of the multiplexing procedure is a capacitated flow graph $G_F = (V_F, E_F)$ that has an edge $(i,j) \in E_F$ with capacity C_{ij} if and only if $(i,j) \in E_P^z$ for one or more $z = 1, \ldots, m$.


(a) Physical graph.


(b) Channel feasibility.


(c) Radiation pattern.


(d) Radiation pattern with uncertainty.


(e) SINR feasibility.

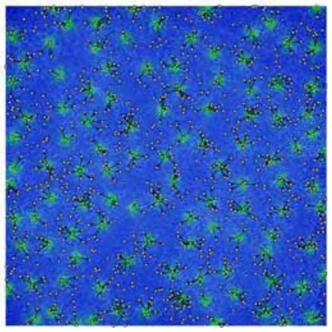

(f) SINR feasibility with uncertainty.

Fig. 3. Physical layer visualization

The visualization of a physical layer snapshot is done by drawing arrows outgoing from each active transmitter to each of its desired receivers. An example is shown in Figure 3(a). The capacity of each edge, $C_{ij}(z)$ can effectively be visualized with color, line thickness, and line dash patterns. Figure 3(b) demonstrates the use of color and line dashes to distinguish between feasible and infeasible edges. Specifically, the solid lines indicate feasible edges, and dashed lines indicate infeasible (but nearly feasible) edges. This visualization technique is extremely effective at identifying nodes in the network that need to adjust their power either by increasing (in order to be better heard) or completely shutting itself off (to reduce total interference of that physical state).

Figure 3(c) and Figure 3(d) show the signal radiation patterns from transmitting nodes under two different models of the channel. The former illustrates a channel state that is assumed to be fully known with no fading, shadowing, or noise in the computation of the attenuation. The latter illustrates a channel state that is highly unstable, modeling uncertainty in the attenuation exponent, shadowing, fading, node positions, and noise. These two visual presentations of the physical state help demonstrate that it is much more difficult to make resource allocation decisions under uncertainty.

Figure 3(e) and Figure 3(f) show bright green regions where the SINR is acceptably high for a feasible channel to exist. That is, if a transmitter were to move into one of those regions, it would be able to transmit a signal to the receiving node associated with that region. Once again, the presence of uncertainty makes it difficult to design node movement and scheduling such that the receiver nodes can remain in the feasible regions of the transmitter with whom they seek to communicate.

The formation of a multiplexed MAC graph from a set of physical layer graphs is a powerful aggregate abstraction of resource allocation. A lot of visual complexity is removed in performing this abstraction while not loosing much of the critical network state information. Figure 4 shows a subset of the physical layer schedules generated by a random packing scheduling algorithm from [12] on the left. On the right is a multiplexed MAC graph which is formed from

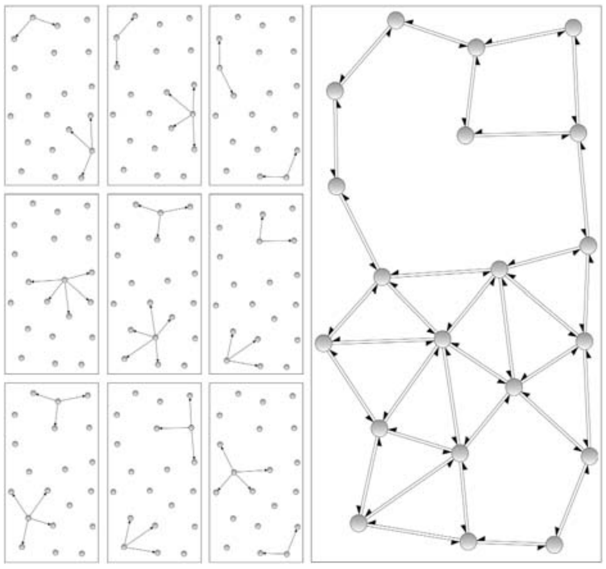


Fig. 4. Multiplexed MAC graph

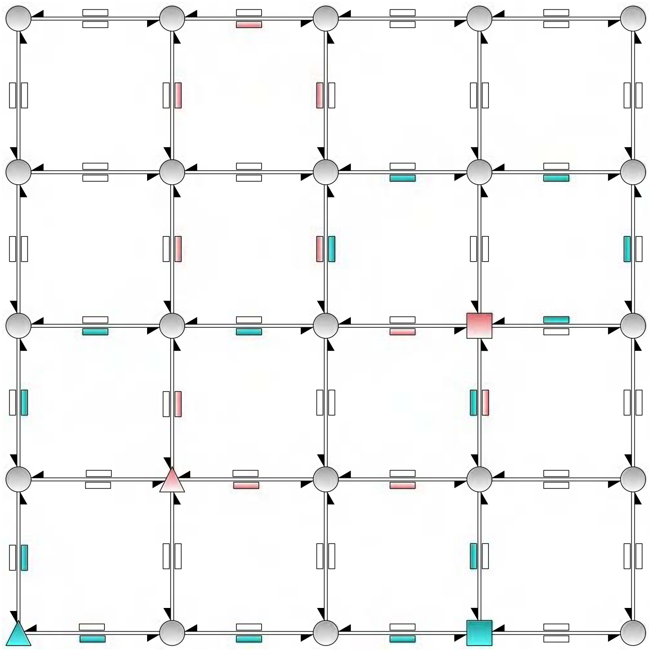


Fig. 5. Flow graph

superimposing all of the physical graphs output by this scheduling method. The graph illustrates global and local connectivity.

5 Network and Transport Layers

Let there be K commodities, each commodity specified by a source node $\sigma_k \in V_F$, and a destination node $\delta_k \in V_F$. A commodity specifies a stream of unique data. A *throughput vector* $\mathbf{f} = \{f_1, \ldots, f_K\}$ is *feasible* if it respects both network capacity constraints and conservation of flow constraints. The objective is to maximize the sum throughput summed over the K commodities. Each commodity may be split across multiple paths; we define a *flow* as the set $\{x_e^k, e \in E_F, k \in [K]\}$, so that x_e^k is the flow for commodity k on edge e. The max multicommodity flow problem is:

$$\begin{aligned} \max_{\mathbf{x}} \quad & F(\mathbf{f}) = \sum_{k=1}^{K} f_k \\ \text{s.t.} \quad & \sum_{i \in V_M} x_{i,\delta_k}^k = f_k, \ k \in [K] \\ & \sum_{k \in [K]} x_{ij}^k \leq c_{ij}, \ i \in V_F, j \in V_F \\ & \sum_{j \in V_M} x_{ji}^k = \sum_{j \in V_M} x_{ij}^k, \ k \in [K], i \in V_F \end{aligned} \tag{6}$$

The first line states the objective is to maximize the sum commodity throughput. The first constraint defines the throughput for each commodity as the sum of the flow over all edges which terminate in commodity k's terminal node δ_k. The second constraint is the capacity constraint; the third is the conservation of flow constraint.

Figure 5 presents a visual representation of a two-commodity flow graph. To demonstrate the amount of flow on each edge, a rectangular "container" is drawn to the right of the edge. If the container is white, this means that no data is flowing on that edge. If the container is filled, data is flowing on the edge, and the color of the fill indicates which commodity the data belongs to. The triangle shape is the source node of the data, while the rectangular shape is the node which is the desired destination for the data.

6 Conclusion

The visualization of the mobility graph informs the observer of the evolution of global patterns in the network topology. The visualization of the physical layer displays the interaction between transmitting and receiving nodes. The visualization of the MAC layer gives a set of "capacity pipes" through which information can travel as represented by the flow graph. Its visualization provides an overview of the movement patterns of data. All of this visual information is critical for making network control decisions.

References

1. Dhoutaut, D., Vo, Q., Lassous, I.G.: Global visualization of experiments in ad hoc networks (2003)
2. Bettstetter, C.: On the connectivity of ad hoc networks. COMPJ: The Computer Journal 47 (2004)
3. van Ham, F., van Wijk, J.J.: Interactive visualization of small world graphs. In: INFOVIS, pp. 199–206. IEEE Computer Society Press, Los Alamitos (2004)
4. Becker, R., Eick, S., Wilks, A.: Visualizing network data. IEEE Transactions on Visualization and Computer Graphics 1, 16–28 (1995)
5. Herman, I., Melançon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 6, 24–43 (2000)
6. Fitzek, F.H.P., Seeling, P., Reisslein, M., Zorzi, M.: ViTAN - visualisation tool for ad hoc networks. IEEE Network 17, 5 (2003)
7. Matthes, M., Biehl, H., Lauer, M., Drobnik, O.: MASSIVE: An emulation environment for mobile ad-hoc networks. In: WONS, pp. 54–59. IEEE Computer Society Press, Los Alamitos (2005)
8. Estrin, D., Handley, M., Heidemann, J., McCanne, S., Xu, Y., Yu, H.: Network visualization with nam, the VINT network animator. Computer 33, 63–68 (2000)
9. Ellson, Gansner, Koutsofios, North, Woodhull: Graphviz – open source graph drawing tools. In: GDRAWING: Conference on Graph Drawing (GD) (2001)

10. Chatila, R.: Mobile robot navigation: space modelling and decisional processes. In: Giralt, F. (ed.) Robotics Research 3, pp. 373–378. MIT Press, Cambridge (1986)
11. Canny, J.: The Complexity of Robot Motion Planning. MIT Press, Cambridge (1988)
12. Wu, Y., Jain, P., Kung, K.: Network planning in wireless ad hoc networks: a cross-layer approach. IEEE Journal on Selected Areas in Communications 23, 136–150 (2005)

A Scalable Aural-Visual Environment for Security Event Monitoring, Analysis, and Response[⋆]

Paul Z. Kolano

NASA Advanced Supercomputing Division, NASA Ames Research Center
M/S 258-6, Moffett Field, CA 94035 U.S.A.
kolano@nas.nasa.gov

Abstract. Intrusion detection systems gather large quantities of host and network information in an attempt to detect and respond to attacks against an organization. The widely varying nature of attacks makes humans essential for analysis, but the sheer volume of data can quickly overwhelm even experienced analysts. Existing approaches utilize visualization to provide rapidly comprehensible representations of the data, but fail to scale to real-world environments due to unrealistic data handling and lack of response facilities. This paper introduces a new tool for security event monitoring, analysis, and response called Savors. Savors provides suitable scalability by utilizing three additional areas of computing. High-end computing brings large amounts of on-demand processing to bear on the problem. Auralization allows both monitoring and analysis to be performed in parallel. Finally, grid computing provides the basis for remote data access and response capabilities with seamless and secure access to organization resources.

1 Introduction

Modern intrusion detection systems (IDS) utilize a collection of sensors to gather data about host and network activity within an organization. This information is then used to identify and analyse current and historical attacks. Sensor data must be correlated [18] to detect single attacks spread out across multiple hosts and networks. Data volume, especially in the network sensor case, can quickly consume even local storage resources, thus data must be significantly reduced before it can be sent to a centralized correlation engine. This creates a distributed information hierarchy where all the data necessary to analyze a given attack may not be present at the same location.

Once an intrusion has been verified, an appropriate response is necessary such as repairing existing damage, blocking traffic to prevent further damage, and adding filters to enhance detection of similar attacks. Taken together, the tasks of analyzing large security data sets, knowing where data resides and how to access it, knowing the network topology and how to access and configure devices within it, etc. can quickly lead to information overload for human analysts. To make intrusion detection more effective, it must be made scalable for human consumption.

[⋆] This work is supported by the NASA Advanced Supercomputing Division under Task Order NNA05AC20T (Contract GS-09F-00282) with Advanced Management Technology Inc.

A variety of tools have been proposed that use visualization techniques to provide greater amounts of information at once in a visual medium more easily grasped by humans. Existing tools, however, do not provide the scalability required for real-world security environments. In particular, they do not provide support for large data sets that may be split across multiple files or distributed across multiple hosts. Beyond basic filtering, they do not provide in-depth analysis capabilities to prune away the vast majority of normal traffic that can overwhelm visual displays. Finally, they do not provide a response capability to take actions based on results uncovered.

This papers presents a new tool called Savors, the **S**calable **A**ural-**V**isual **O**perations **R**oom for **S**ecurity. Savors provides a scalable approach to monitoring, analyzing, and responding to arbitrarily sized and distributed security event data by combining four different areas of computing within five visualization tools, an auralization component, and supporting software. Visualization is used to provide quickly comprehensible graphical representations of the data that take advantage of the human ability to recognize visual patterns and anomalies. High-end computing is used to support computationally intensive anomaly calculations that highlight areas of interest. Auralization is used to increase human interface bandwidth by utilizing sound in parallel with sight. Finally, grid computing is used to seamlessly hook resources together, shielding the analyst from low level details while providing a strong security environment to protect resources.

This paper is organized as follows. Section 2 discusses related work. Section 3 gives an overview of Savors. Sections 4, 5, and 6 describe the monitoring, analysis, and response aspects of Savors. Finally, Section 7 discusses conclusions and future work.

2 Related Work

There are a variety of efforts related to the problem addressed by this paper. RUMINT provides a variety of security visualizations with DVR-like controls, which include binary rainfall displays [4] that map varying numbers of bits to pixels, parallel coordinate and scatter plots [3] between various protocol fields, and ASCII packet information.

IDS Rainstorm [1] shows the occurrence of IDS alarms across IP addresses and time. Multiple axes are used to visualize a large range of IP addresses with support for zooming in on a particular region to show increased detail. The IDS Rainstorm view is capable of utilizing every pixel for information, but is mainly useful for monitoring network state and not for providing additional insight into attacks.

The Visual Firewall [12] shows four simultaneous views of network traffic including parallel coordinate and animated glyph-based scatter plots between local ports and foreign IP addresses, a network throughput graph over time, and a quad-axis diagram mapping local IP address, foreign subnet, IDS alert type, and time. The concurrent display allows the user to consider four different perspectives of the same traffic at once, although only one view may be shown at full size.

The SIFT tool suite [20] consists of two main visualization tools. NVisionIP represents a complete class B network address space using a matrix of subnets and hosts. Users may perform hardcoded queries to see basic numeric characteristics, such as bytes per host, across all hosts. Users may also drill down to the host level to see more detail. VisFlowConnect-IP shows a parallel coordinate plot between internal and external hosts while distinguishing between inbound and outbound traffic.

TNV [9] shows a grid of host activity over time with the middle portion of the grid devoted to a focus area and the outer portions showing the context. The focus area shows greater detail about hosts within a particular period of time including link communication and port activity. The context area shows color-based packet counts of the remaining time period to provide continuity between the focus area and the rest of the data. Surrounding the main grid are a historical summary and a control timeline as well as additional areas with detailed information about a specific host.

VisAlert [6] plots the time, location, and type of IDS alerts using a main circular area with a surrounding set of concentric circles. Event type is represented by regions along each outer ring where time increases with radius. Alert location is plotted from the types of the innermost ring to a graph of network topology in the main area. Persistent alerts and alert volume are indicated using color coding and graph node size.

Peep [8] auralizes network status using natural sounds that remain pleasing to the ear even when played continuously. Peep distinguishes between one-time events, periodic heartbeats, and specific states using representative sounds such as bird chirps, cricket chirps, and water flows, respectively. Peep clients generate sound information from local state such as system logs, which is mixed and played by the Peep server.

CyberSeer [15] incorporates auralization into its visualization component to enhance monitoring performance. The visualization component utilizes spectral analysis techniques, which are based on the observation of periodic behavior in network traffic. The auralization component synthesises sounds in response to changes in data patterns. CyberSeer further proposes an immersive 3D environment to take advantage of human spatial perception that would significantly increase aural and visual bandwidth.

3 Savors Overview

Savors consists of five visualization tools, an auralization component, and supporting software that together provide an integrated environment for monitoring, analyzing, and responding to security event data. Tools are split into client and server components, with the server responsible for locating and processing the requested data and streaming it to the client for display. Data is located using an administrator-defined function that returns the data file path corresponding to any given timestamp. Savors has built-in support for libpcap [13] and CSV data formats, but custom handlers can be easily added. Servers may be invoked by the client either locally, or remotely over SSH, which allows the client to be geographically separated from the data. Data is sent in batches and buffered at the client to maintain consistent display speed. Subsequent batches are transferred asynchronously while the current batch is being displayed.

The client is responsible for providing a visual (or aural) representation of the data along with playback controls and input fields to manipulate its behavior. All Savors visualization tools share the same basic interface. The bottom of each tool consists of buttons to play, pause, rewind, etc. the data stream as well as input fields to select/display the time period and other parameters. The left side of each tool shows the current mapping between colors and protocols. Protocols may be filtered from the display by clicking on the corresponding color box. The central region of each tool shows a specific visualization. All but the Savors Group tool of Section 4.1 have a set of panes below the central region to display details of any packet/flow over which the mouse pointer hovers (for the remainder of the paper, "flow" and "packet" will be used interchangeably).

Though not required, Savors was intended to be run on a small hyperwall-like device [16] with a dedicated display and back-end system for each tool. Tools can be run on the same system, but will not achieve maximum performance due to their greedy nature. Savors was written in Perl/Tk [14], which allowed for a very rapid and portable implementation. All tools were tested on a real 1.3 TB data set with over 10 billion flows captured during a 7.5 month period using three commercial network flow sensors. These sensors produce aggregated packet data in the form of network flows that contain information such as source/destination IP address/port, byte/packet count, etc. The vendor's binary storage format is proprietary and undocumented, but a CSV export tool was available. Though non-optimal for performance, data was kept in compressed CSV form during prototyping for practicality. Note that all IP addresses from this data set displayed in the remainder of the paper have been anonymized by Cryto-PAn [19].

To support analysis over such large data sets, three of the visualization tools are designed to utilize high-end computing (HEC) resources such as clusters and supercomputers. Additionally, a response capability is integrated into all the visualization tools that allows users to instantaneously react to displayed data by blocking traffic, adding new filters, etc. The details of HEC usage, firewall configuration, and sensor updates as well as the underlying capabilities for locating data and controlling access to data and resources are hidden behind the Savors visual interface. This allows users to focus on their specific security tasks without the need to become experts in every aspect of their organization's infrastructure. The following sections describe Savors in detail.

4 Monitoring

To assess the state of the network at any given time, Savors provides high, medium, and low level views of network flow data, which are described in the following sections.

4.1 Savors Group

High level monitoring is supported by the Savors Group tool, which displays an overview of network activity using a squarified treemap [2]. A treemap utilizes all available screen real estate while conveying hierarchical structure as well as relative size and content information. Savors Group uses a three level hierarchy of a set of sites that contain a set of hosts that perform a set of activities. The area of each site/host with respect to the total/site area represents the percentage of total/site bandwidth (in bytes or packets) that the site/host is utilizing. Totals are based on the hosts/sites displayed, thus accuracy may be improved by increasing the number of hosts/sites shown at once. Activities are represented by color, thus the dominant activity of all sites, one site, or one host is indicated by the most dominant color of the corresponding region at any given time.

Figure 1 shows sample output from Savors Group, which indicates that Site 2 is utilizing the most bandwidth, with HTTP being the most prevalent protocol. From a security perspective, there are two regions of note. Site 2 has a host using significant bandwidth for BitTorrent traffic, which is often used for copyrighted content, thus indicating a possible policy violation of downloading or hosting inappropriate material. Site 1 has a host that is receiving both secure web traffic and some unknown traffic, which may be an indicator of a compromised host that is now accepting malicious commands.

4.2 Savors Flow

Medium level monitoring is supported by the Savors Flow tool, which displays host interactions using a parallel coordinate plot of the local and remote IP addresses and ports of each network flow, inspired by RUMINT's similar capability [3]. IP addresses are mapped along the vertical axes by dropping the decimal point and moduloing against the pixel height. Ports are mapped similarly. This scheme has the desirable property that hosts on the same subnet and numerically adjacent ports are visually adjacent.

Figure 2 shows sample output from Savors Flow. Natural nexuses form along the IP address axes where a single host serves many requests from many ports and along the port axes where the same port is used on different hosts for the same service. Port/host scans may be seen when the same remote host accesses many adjacent local ports/hosts. In the figure, two simultaneous host scans from different remote IP addresses may be seen along the top and middle of the local IP address axis (second of five from the left). These scans are still visible despite the considerable detail in the display.

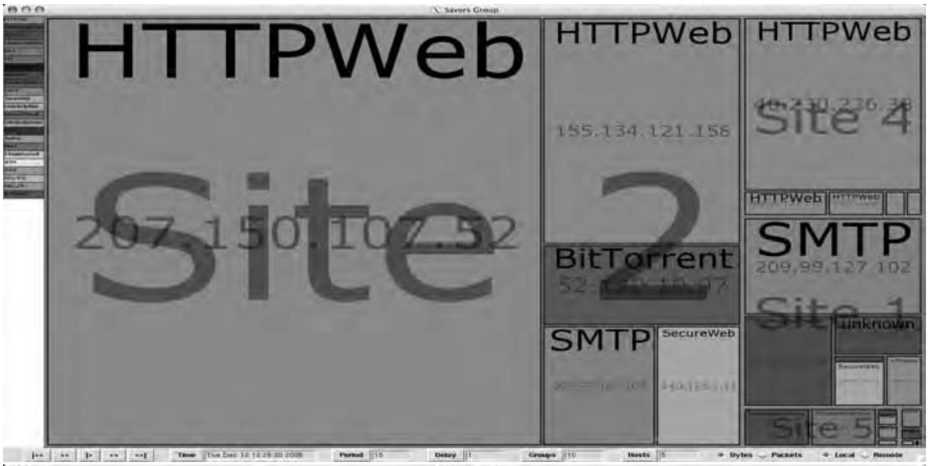
4.3 Savors Content

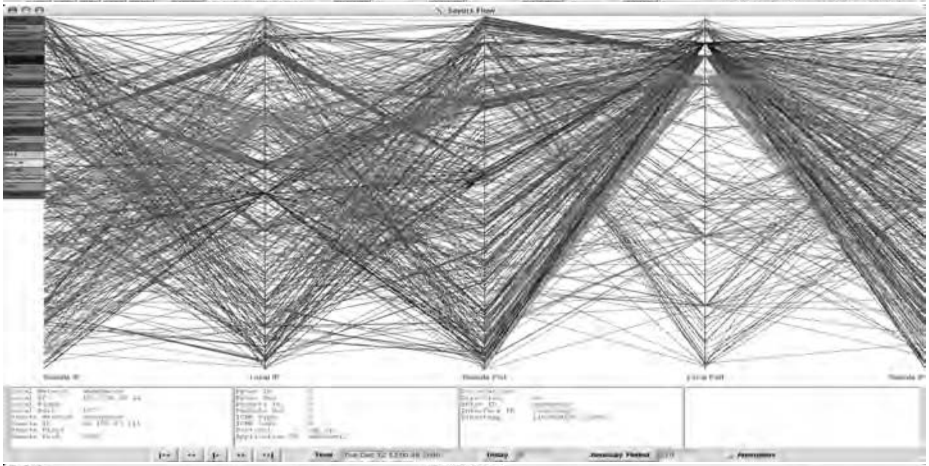
Low level monitoring is supported by the Savors Content tool, which displays pixelized representations of the byte contents of each flow, inspired by RUMINT's similar capability [4]. To the right of each flow is a bandwidth bar, where the length of the bar is proportional to the logarithm of the byte/packet count, with red/green coloring for incoming/outgoing traffic. To the left of each flow is an area for visual flags that are displayed whenever a configurable set of regular expressions is matched in the content. When flow content is not transferred from the sensor to the correlation engine due to bandwidth constraints, the Savors Content server must be invoked on the sensor itself.

Figure 3 shows several features of Savors Content in action. A flag was generated when the content matched the regular expression "USER.*PASS". This indicates an unencrypted protocol such as FTP or POP that is exposing user names and passwords in the clear. Many such flags might indicate a brute force attack against the protocol. The black scan line a quarter of the way down is stopped at a flow that uses much higher bandwidth than those around it. Near the bottom of the pixelization region, there is a burst of HTTP traffic, which may indicate a reconnaissance operation against the organization. The bandwidth bars seem to indicate more inbound than outbound traffic for these requests, however, which does not support this hypothesis, but is itself suspicious for HTTP traffic and worthy of further investigation.

4.4 Savors Sound

A large part of intrusion detection involves analyzing historical events to identify sources of compromise, targets of attack, and the extent of any damage. Analysis





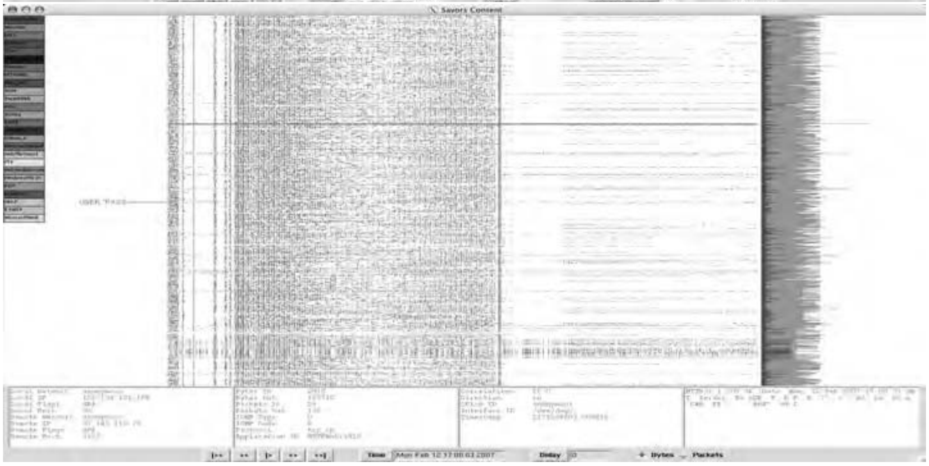


Fig. 1. (Top) Savors Group **Fig. 2.** Savors Flow **Fig. 3.** Savors Content

demands full visual attention, thus impairing the ability to monitor current network activities. To improve monitoring capabilities during analysis and vice versa, Savors takes advantage of the human ability to process sound in parallel with sight. The Savors Sound tool auralizes both current and historical security data allowing the user to aurally monitor current data while visually analyzing historical data and vice versa. All Savors visualization tools can also sonically flag specific events, allowing them to focus attention when needed. All sounds are mixed dynamically by the Enlightened Sound Daemon [5], allowing multiple events from multiple clients to be auralized in parallel.

Savors Sound is based on Peep's three category model [8] of *events*, which are one-time occurrences, *heartbeats,* which are events that occur periodically over time, and *states*, which are the set of attributes that the system possesses at any given point in time. Events and heartbeats are defined based on configurable filters. Each match of a particular event filter triggers the playing of the associated sound. Heartbeat filter matches do not trigger sounds, but instead determine the frequency with which the associated sound should be played. States may be any quantifiable attribute, but only packets and bytes per time unit are currently implemented. Savors Sound can use the Peep sound library, which maps directly to the Peep model with sounds such as bird chirps for events, cricket chirps for heartbeats, and water flow for states.

5 Analysis

The Savors monitoring tools of the previous section are primarily concerned with conveying the state of the network through the display of raw and summarized flow information. As discussed, this information by itself is often enough to suggest systems that are compromised or under attack. The bulk of the displayed state, however, consists of normal activity that does not warrant further investigation. In addition, many indications of compromise and attack are received out-of-band from local system administrators or from individuals belonging to other organizations. Before an appropriate response can be initiated, it must be determined if, when, how, and from which host(s) a compromise has occurred and which other hosts may have been involved. Answering such questions accurately often demands the analysis of large volumes of historical data.

To support advanced analysis operations, Savors provides three HEC-assisted tools. The use of on-demand high-end computing has several advantages over a database model. The de facto standard format for packet trace storage, libpcap [13], and the emerging standard format for network flow storage, IPFIX [17], are both based on flat files. An on-demand HEC approach allows data to be analyzed in its native format exactly when needed, which simplifies transport and storage and frees up high-end resources for other purposes at other times. In contrast, a database solution requires dedicated resources in constant use just to insert continuously generated security data. All computations discussed in the following sections were performed on a 220 processor SGI Altix supercomputer with 2 GB of memory per processor.

5.1 Anomalous Savors Flow

As seen in Figure 2, visualizing all flows can result in a very cluttered display. Even though some details can be extracted, as discussed in Section 4.2, most flows

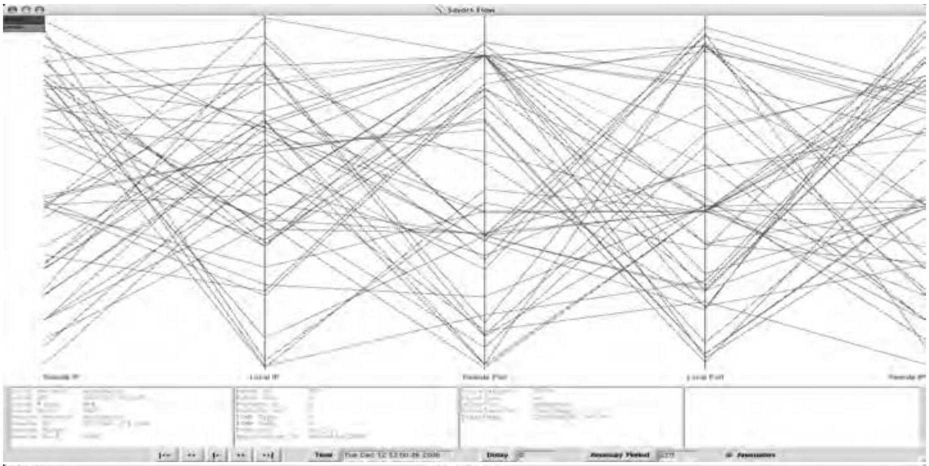
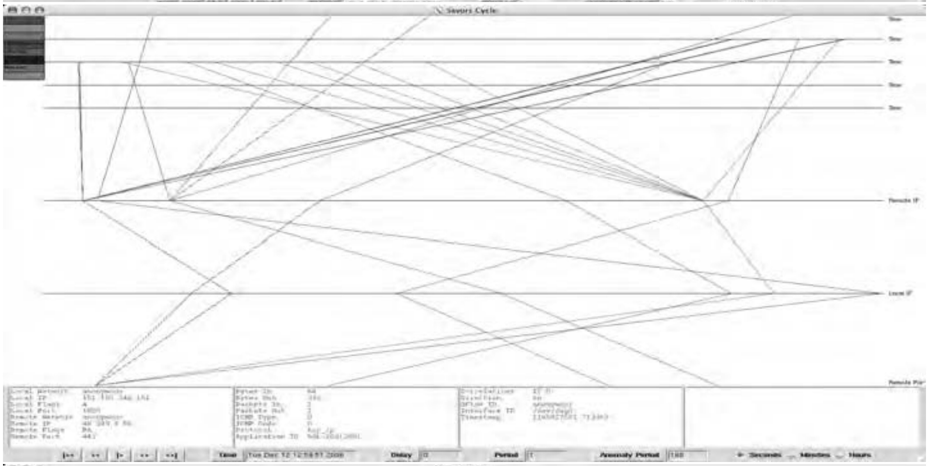
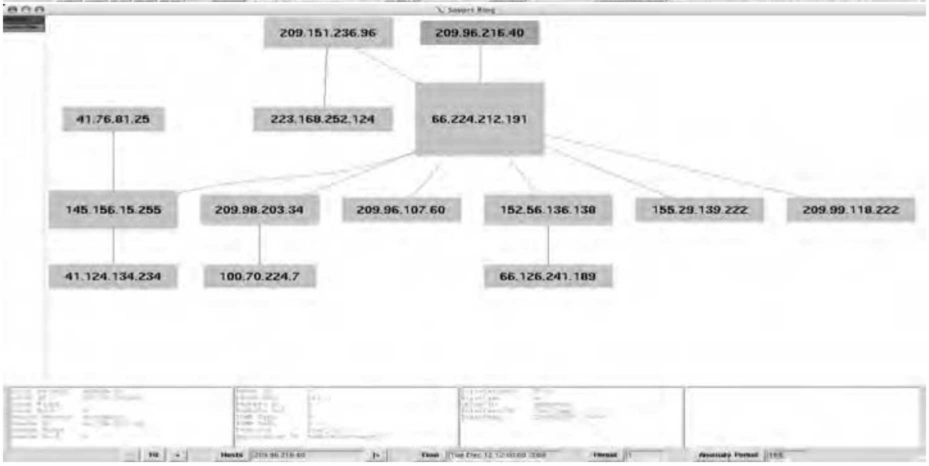




Fig. 4. (Top) Anomalous Savors Flow **Fig. 5.** Savors Cycle **Fig. 6.** Savors Ring

represent normal activity, thus are not of particular interest. The flows deserving attention are those in which internal hosts are either accepting connections on ports they do not normally accept or initiating connections to ports to which they do not normally connect. To find such flows, Savors Flow supports anomaly detection, where flows in the display period are only shown if they are abnormal. Normalcy is derived using either an on-demand HEC computation or a cached instance of such a computation. Given an anomaly period P, an inbound flow not in P is normal if there is a flow in P with the same protocol and destination IP address/port. An outbound flow not in P is normal if there is a flow in P with the same protocol, source IP address, and destination port.

Figure 4 shows the anomalous flows over the same time period as Figure 2. As can be seen, the number of flows has been significantly reduced. This reduction is actually much greater than shown since in Figure 2, many flows have come and gone after the configured screen capacity has been reached, whereas in Figure 4, the display has not yet reached capacity, thus all flows are shown. The figure shows one Gnutella flow and several unknown flows. Of particular note is the inbound Gnutella flow, shown in the details pane, which should not be running within most organizations due to security vulnerabilities and misconfiguration that can lead to disclosure of sensitive information.

5.2 Savors Cycle

Organizations typically place significant restrictions on inbound access to protect internal resources, but have more permissive outbound policies to support access to Internet resources. Attackers use this fact by configuring compromised internal hosts to periodically connect outbound to retrieve further instructions. A human is often best-equipped to recognize such communication due to its widely varying and/or random connection patterns. The Savors Cycle tool supports such analysis by plotting periodic flows in a given time period. Many instances of periodic communication are legitimate such as queries to DNS servers or updates from time servers, thus to give meaningful results, Savors Cycle also supports anomaly detection in the same manner as Savors Flow. After the normal behavior profile is created, the computation counts the number of instances of anomalous connections and plots those past a configurable threshold.

Figure 5 shows sample output from Savors Cycle, with some protocols filtered for readability, which displays a parallel coordinate plot between the local IP address, the remote IP address and port, and the time of each network flow. Each pixel on the time axes may be configured to represent either a second, a minute, or an hour, where time increases with distance from the top of the display. Some connections only appear to have one instance as there is currently no minimum gap requirement, thus a single second may have many of the same connection.

5.3 Savors Ring

Compromised hosts typically become a launching point for additional attacks that take advantage of internal network access. Anomalous association with such hosts often indicates other sources or targets of attack. Given a set of starting hosts, the Savors Ring tool constructs a graph (rendered with GraphViz [7]) of all communication, either direct or indirect, involving those hosts for a given period of time. Because some hosts are common rendezvous points such as DNS servers, which could result in a complete graph, only anomalous flows are considered, which are computed using an on-demand HEC computation or a cached instance. In this case, given an anomaly period P, a flow not in P is deemed normal if there is any flow in P involving the same two hosts.

Figure 6 shows sample output from Savors Ring. The green box (i.e. 209.96.216.40) represents the starting host. Boxes are sized according to the number of directly associated hosts, thus larger boxes represent hosts more likely to be either compromised or the source of compromise. Protocols deemed unlikely to be the medium for attack can be filtered using the color box on the left to break the graph into subgraphs with only the protocols of concern. In the figure, the initial line segment is a standard ping request, which, if filtered, completely segments the starting host from the rest of the graph, indicating no additional damage if the starting host were compromised.

6 Response

The ultimate goal of intrusion detection is not just to find attacks, but to respond to them. This includes verifying compromise, repairing damage, preventing further damage, and enhancing detection of similar attacks. Savors integrates response directly into all of its visualization components to allow the user to instantaneously react to displayed data. A right click on any flow-related visual element within the display brings up a list of actions that may be taken based on the local/remote hosts/ports of the given flow.

For local hosts, actions include downloading logs and opening a terminal, which can be used to verify compromise and repair damage, respectively. To perform these actions on a given system, the user must have appropriate login credentials. Since any network-accessible host within an organization may be subject to attack and require an appropriate response, users may need to juggle many different sets of credentials between large numbers of hosts. Different systems may also require different access paths through bastions, etc.

To support response capabilities in a scalable manner, Savors utilizes Mesh [11], which is a lightweight grid middleware that utilizes existing SSH infrastructure. Mesh provides single sign-on across distributed resources, allowing users to access any Mesh-enabled system with the same user identity and credentials. For organizations with different user domains and varying network access paths, Mesh provides automatic translation of user identities between domains and automatic location of the Mesh bastion responsible for each host. In addition, Mesh provides fine-grained authorizations that allow Savors users to be restricted to only the capabilities defined by the administrator. Not only does this allow each Savors action to be permitted or denied on a per host and/or per user basis, it also allows additional restrictions to be enforced such as only permitting actions from specific systems (e.g. the system(s) running Savors).

Besides downloading logs and opening terminals, Savors supports blocking of traffic to/from any subset of the local/remote hosts/ports of a flow to prevent further damage. For example, inbound traffic from any port of a malicious remote host can be blocked to any port of any local host to prevent additional attacks from the same host. Blocking is achieved through integration with Diaper [10], which is a framework for dynamically controlling the access control lists (ACLs) of network devices. Diaper supports firewalls, routers, and switches from major network vendors, allowing traffic to be blocked both at the perimeter as well as between internal hosts. ACL changes are requested through SSH remote commands, which are authenticated and authorized by Mesh. Diaper supports its own fine-grained authorizations that allow the administrator to define the scope of what may be blocked on a per user basis. Thus, Savors users can be given as much or as little control over network policies as desired.

Finally, to enhance detection of similar attacks, Savors allows users to define new IDS signatures and make them operational on existing IDS sensors (currently, only Snort sensors are supported). Savors constructs a signature template using the details of the selected flow, which may be edited before activation. A signature is activated by transferring it to the sensor and forcing a reload of the corresponding ruleset, both of which are authenticated and authorized by Mesh. This feature can be utilized to apply knowledge of attack profiles gained during analysis to tune the IDS.

Through the integration with Mesh and Diaper, Savors hides low-level operational details behind its visual interface and allows actions to scale across an entire organization with just a single set of login credentials. Users do not need to know how to access a particular host, which network device is responsible for blocking/monitoring which traffic, nor how to update the ACLs/signatures of that device. Once configured by the administrator, the appropriate actions are automatically taken behind the scenes by Mesh and Diaper, allowing users to focus on monitoring and analysis tasks.

7 Conclusions and Future Work

This paper has described a new tool for security event monitoring, analysis, and response called Savors, the **S**calable **A**ural-**V**isual **O**perations **R**oom for **S**ecurity. Savors scales to real-world environments by utilizing four distinct areas of computing. Visualization-based monitoring components display high, medium, and low level representations of security event data in a form that conveys significant information in an easily understandable form. Visualization-based analysis tools utilize high-end computing resources on-demand to compile behavior profiles that point to anomalous behavior. Auralization allows both monitoring and analysis to be performed in parallel and draws attention to critical events in one tool when utilizing another. Remote data access and response capabilities across distributed resources are enabled using grid computing that provides a secure, single sign-on environment. Savors is portable across operating systems and was tested on a real data set containing over 1.3 TB of network flow data.

There are a number of directions for future work. Savors clients need an annotation capability to record interesting periods of activity. Servers could be modified to support multiple clients with a single shared view. A timeline like TNV's will be added to provide rapid time manipulation and a high-level summary of past activity. A drill down capability will be added to Savors Group to provide more detail about a specific site or host. A number of performance improvements are possible. The HEC analysis code must be optimized to eliminate file I/O bottlenecks by using shared memory capabilities. A minimum gap requirement must be added to the Savors Cycle analysis code to eliminate flows in close proximity. Finally, alternatives to SSH for streaming and Perl/Tk for rendering should be investigated to see if performance can be increased.

Acknowledgments

Thanks to Justin Cassidy and Matt Linton for supplying the flow data and suggesting the idea behind the Savors Ring tool. Thanks to Jeff Ward for suggesting the anomaly model for Savor Flow and the idea behind the Savors Cycle tool.

References

1. Abdullah, K., Lee, C., Conti, G., Copeland, J., Stasko, J.: IDS RainStorm: Visualizing IDS Alarms. In: IEEE Wkshp. on Visualization for Computer Security (October 2005)
2. Bruls, M., Huizing, K., van Wijk, J.J.: Squarified Treemaps. In: 2nd Joint Eurographics - IEEE TCVG Symp. on Visualization (May 2000)
3. Conti, G., Abdullah, K.: Passive Visual Fingerprinting of Network Attack Tools. In: ACM Wkshp. on Visualization and Data Mining for Computer Security (October 2004)
4. Conti, G., Grizzard, J., Ahamad, M., Owen, H.: Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries. In: IEEE Wkshp. on Visualization for Computer Security (October 2005)
5. Enlightened Sound Daemon, `http://www.tux.org/~ricdude/EsounD.html`
6. Foresti, S., Agutter, J., Livnat, Y., Moon, S., Erbacher, R.: Visual Correlation of Network Alerts. IEEE Computer Graphics and Applications 26(2) (March 2006)
7. Gansner, E.R., North, S.C.: An Open Graph Visualization System and Its Applications to Software Engineering. Software: Practice and Experience 30(11) (August 2000)
8. Gilfix, M., Couch, A.: Peep (The Network Auralizer): Monitoring Your Network With Sound. In: 14th USENIX Large Installation System Administration Conf., (December 2000)
9. Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Focusing on Context in Network Traffic Analysis. IEEE Computer Graphics and Applications 26(2) (March 2006)
10. Kolano, P.Z.: Maintaining High Performance Communication Under Least Privilege Using Dynamic Perimeter Control. In: 12th European Symp. on Research in Computer Security (September 2007)
11. Kolano, P.Z.: Mesh: Secure, Lightweight Grid Middleware Using Existing SSH Infrastructure. In: 12th ACM Symp. on Access Control Models and Technologies (June 2007)
12. Lee, C., Trost, J., Gibbs, N., Beyah, R., Copeland, J.A.: Visual Firewall: Real-time Network Security Monitor. In: IEEE Wkshp. on Visualization for Computer Security (October 2005)
13. Libpcap format, `wiki.wireshark.org/Development/LibpcapFileFormat`
14. Lidie, S.: Perl and the Tk Extension. The Perl Journal 1(1) (1996)
15. Papadopoulos, C., Kyriakakis, C., Sawchuk, A., He, X.: CyberSeer: 3D Audio-Visual Immersion for Network Security and Management. In: ACM Wkshp. on Visualization and Data Mining for Computer Security (October 2004)
16. Sandstrom, T.A., Henze, C., Levit, C.: The Hyperwall. In: 1st Intl. Conf. on Coordinated and Multiple Views in Exploratory Visualization (2003)
17. Trammell, B., Boschi, E., Mark, L., Zseby, T., Wagner, A.: An IPFIX-Based File Format. IETF Internet Draft (July 2007)
18. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A Comprehensive Approach to Intrusion Detection Alert Correlation. IEEE Trans. on Dependable and Secure Computing 1(3) (July 2004)
19. Xu, J., Fan, J., Ammar, M.H., Moon, S.B.: Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. In: 10th IEEE Intl. Conf. on Network Protocols (November 2002)
20. Yurcik, W.: Visualizing NetFlows for Security at Line Speed: The SIFT Tool Suite. In: 19th USENIX Large Installation System Administration Conf. (December 2005)

Complexity Analysis for Information Visualization Design and Evaluation

Ying Zhu, Xiaoyuan Suo, and G. Scott Owen

Department of Computer Science
Georgia State University
Atlanta, Georgia, USA
yzhu@cs.gsu.edu

Abstract. In this paper, we present a method for analyzing the complexity of information visualization. We have identified a number of factors that influence the efficiency of visual information read-off and integration. Here the complexity is measured in terms of visual integration, number of separable dimensions for each visual unit, the complexity of interpreting the visual attributes, and the efficiency of visual search. These measures are derived from well established psychological theories. Together they indicate the amount of cognitive load for comprehending a visualization design. This method is particularly useful for developers to quickly evaluate multiple design choices. Finally, we demonstrate our method through a complexity analysis on a computer security visualization program.

1 Introduction

Today information overload is a major challenge in many areas. For example, computer security professionals need to process data from a myriad of sources, including network devices, firewalls, intrusion detection programs, vulnerability scanners, and operating systems. Data visualization has the great potential to alleviate the heavy cognitive load associated with processing this overabundance of information. However, poorly designed visualizations can be counterproductive or even misleading. Therefore, visualizations must be carefully designed and evaluated.

The evaluations of visualization generally involve user studies. Common measures of visualization include task completion time, error rate, or subjective satisfaction. However, these are largely black-box approaches and the results often do not explain whether or why certain features of visualizations cause the performance or usability problems. Besides, user studies are often difficult to manage and can only be conducted after a program has been developed.

In this paper, we propose an alternative evaluation method – complexity analysis, which systematically evaluates a set of factors that influence the efficient processing of visual information. The proposed method is grounded in well established psychological theories [1-5] and the outcome of this analysis serves as an indicator of the cognitive load associated with comprehending a visualization design.

The proposed complexity analysis is particularly useful during the visualization design stage before any user study can take place. It allows designers to quickly evaluate multiple visualization designs in terms of their complexity. The results of the complexity analysis not only provide guidance to the design but also help generate hypotheses for user studies to verify.

2 Background and Related Works

The proposed complexity analysis is based on a number of psychological theories [1-5], including Guided Visual Search theory [5], Gestalt theory [4], and Cognitive Load Theory [1]. According to the Cognitive Load Theory, there are three types of cognitive load: intrinsic cognitive load, extraneous cognitive load, and germane cognitive load. The mental effort to comprehend a data visualization is part of the extraneous cognitive load, which is a major factor that influences the task performance. The proposed visualization complexity analysis is an attempt to measure the extraneous cognitive load of visualization comprehension.

Several researchers have touched on the issues of complexity in visual display, but none of them have considered it in a systematic way. Bertin [6] and Trafton, et al. [3] have used the number of dimensions as a measure for the complexity of visual displays, and considered visualizations with more than three variables to be complex. Brath [7] has proposed a heuristic method to measure the effectiveness of the mapping from the data dimension to the visual dimension by classifying the visual mappings into one of the four categories. Our evaluation method is more comprehensive than the previous methods and considers many factors that are not considered before.

3 Complexity Analysis

The processing of a data visualization depends on a host of psychological processes, including information read-off, integration, and inference [3]. The main goal of the proposed complexity analysis is systematically evaluate the major factors that influence the efficiency of information read-off and integration. Visual inference is currently beyond the scope of this study due to a lack of understanding of its psychological process. However, as Trafton, et al. [3] point out, visual inference is likely to depend on visual integration and information read-off.

The complexity analysis is carried out in the following steps:

1) We conduct a hierarchical analysis on the data visualization design and divide the visualization into five layers: workspace, visual frame, visual pattern, visual units, and visual attributes.
2) Next, we analyze the efficiency of integrating visual elements. Based on the hierarchical analysis, we build a tree structure that depicts how visual frames are organized in each workspace, and how visual patterns are organized in each visual frame. We call this a visual integration complexity tree (figure 1) because it shows how a reader might mentally integrate visual frames and visual patterns. The number of nodes on this tree is the maximum number of visual integrations that a reader might perform.

3) We then analyze the efficiency of interpreting visual units. For each type of visual unit, we identify the visual attributes that are used to encode data parameters. Each of the encoded visual attributes is called a dimension. For each encoded visual attribute, we estimate the complexity of mapping the visual attribute to its corresponding data parameter. The outcome is a visual mapping complexity tree (figure 2).
4) Finally we analyze the efficiency of visual search. We evaluate the target-distracter difference for four attributes that are important for efficient visual search. These attributes are color, motion, size, and orientation.

In the following sections, we give more details about the major factors we considered during the complexity analysis.

3.1 Hierarchical Analysis of Data Visualization

We divide a data visualization into five hierarchical layers: workspace, visual frame, visual patterns, visual units, and visual attributes. A workspace is one or more visual frames that are designed for a specific purpose. A visual frame is a window within a workspace and contains multiple visual patterns. A visual pattern is a set of visual units that are readily perceived as a group. In our study, visual patterns are identified based on four Gestalt laws [4]. We also identify six types of visual units: point, line, 2D shape (glyph), 3D object, text, and image. Each visual unit is defined by seven visual attributes [6]: position, size, shape, value, color, orientation, and texture.

3.2 Visual Integration

Larkin and Simon [9] point out that a main advantage of visualization is that it helps group together information that is used together, thus avoiding large amounts of search. Gestalt theory [4] describes how these visual "chunks" are perceptually grouped. In complex problem solving, however, these visual "chunks" need to be integrated [3], which adds to the extraneous cognitive load [1]. For example, readers often need to remember and integrate different visual frames and visual patterns [6, 10].

In this study, the cognitive load of visual integration is estimated by building a visual integration complexity tree (figure 1). For each visual frame, we identify the visual patterns in that frame based on four Gestalt laws: proximity, good continuation, similarity, and common fate [4, 10]. The number of nodes on the visual integration complexity tree represents the upper bound of visual integration a reader might perform, just like the asymptotic notion represents the upper bound of the computational complexity.

3.3 Separable Dimensions for Visual Units

In order to visualize a data set, different data parameters are mapped to different visual attributes of different visual units. Each encoded visual attribute is called a dimension. Such visual mappings need to be identified and remembered by readers, thus adding to the extraneous cognitive load.

Some researchers have suggested using the number of dimensions as an indicator of the complexity of visualization [3, 6, 7]. However, we need to consider the visual processing of integral and separable dimensions. In short, integral dimensions are

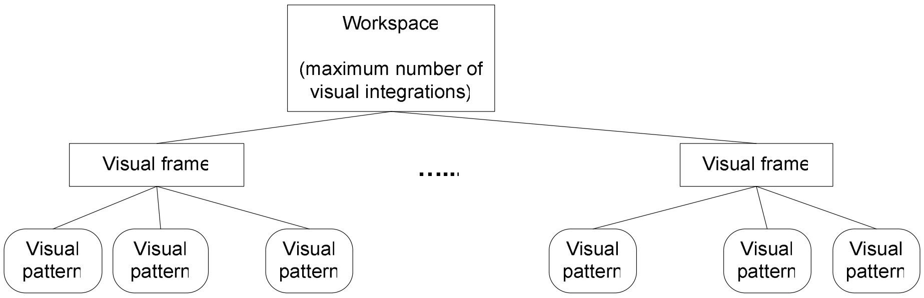


Fig. 1. Visual integration complexity tree

processed together, while separable dimensions are processed individually [2]. Based on this theory, we propose the following classification:

- X and Y coordinates for 2D position are considered as one integral dimension.
- Color and value are considered as one integral dimension.
- Shape, size, and orientation are considered as separable dimensions.

Such grouping may be changed for specific visualizations. For example, in some cases, shape and size may be considered as integral dimensions, or X and Y axes may be considered separable dimensions.

3.4 Interpreting the Values of Visual Attributes

Readers not only need to memorize the mapping between data parameters and visual attributes, they also need to interpret the value of these visual attributes. For example, what does a color code mean? What does a particular shape represent? The mental effort for such interpretation is another source of extraneous cognitive load [1]. In our analysis, each integral or separable dimension is assigned a score for the complexity of interpreting the values of the visual attribute (see table 1).

Table 1. Complexity scores for interpreting the meaning of visual units

Complexity score	*Criteria*
5	Very difficult to interpret. There is no legend. A typical reader has to memorize the mapping between the value of the visual attribute and the value of the corresponding data parameter
4	More difficult to interpret. A typical reader needs to frequently refer to a legend to interpret the value of the visual attributes
3	Somewhat difficult to interpret. A typical reader needs to refer to a legend from time to time.
2	Relatively easy to interpret. A typical reader only needs to refer to a legend occasionally.
1	This is common knowledge. There is no need to memorize or refer to a legend.

The complexity score for a type of visual unit is the sum of complexity scores of its separable dimensions. The complexity score for a visual frame is the sum of scores of different types of visual units it contains, and so on. Based on the above analysis, a visual mapping complexity tree can be built (see figure 2).

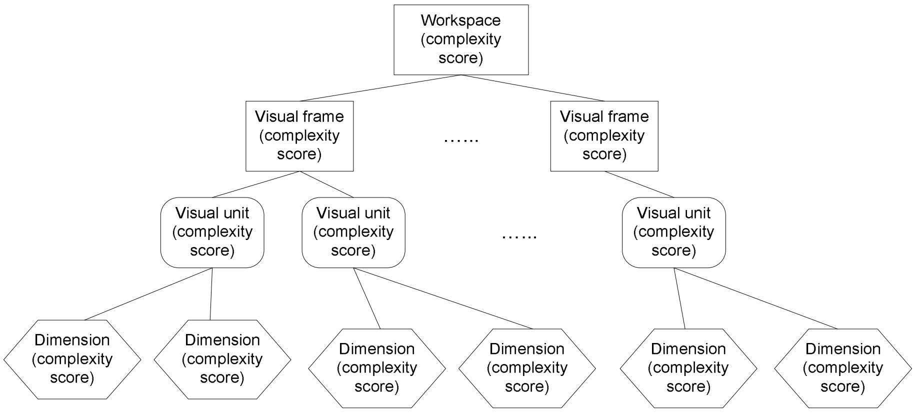

Workspace (complexity score)
Visual frame (complexity score)
……
Visual frame (complexity score)
Visual unit (complexity score)
Visual unit (complexity score)
……
Visual unit (complexity score)
Dimension (complexity score)
Dimension (complexity score)
Dimension (complexity score)
Dimension (complexity score)
Dimension (complexity score)
Dimension (complexity score)

Fig. 2. Visual mapping complexity tree

3.5 Efficiency of Visual Search

One of the main advantages of visualization is that they can support efficient visual search [5]. A data visualization should be constructed to take advantage of this feature, and it’s important to evaluate how well a visualization supports visual search.

According to Wolfe and Horowitz [5], target-distracter difference is the key to efficient visual search, and there are four major factors that affect the target-distracter differences – color, motion, size, and orientation. Target-distracter difference indicates how a target stands out from the background. Here we propose a method to evaluate the target-distracter differences for color, motion, size, and orientation.

Table 2. A look-up table for target-distracter difference scores

		Background homogeneity		
		High	Medium	Low
Target-background distance	High	5	4	4
	Medium	3	3	2
	Low	3	2	1

First we identify a visual target that we are interested in evaluating. Then we evaluate the target-distracter difference based on two factors – background homogeneity and target-background distance. Table 2 is a look-up table for estimating target-distracter difference score. A higher score means greater target-distracter difference, which means more efficient visual search. The scores in the table are based on our experience and need to be refined through empirical studies. Currently this table applies to color, motion, size, and orientation. In the future, each attribute may use a different table.

Background homogeneity indicates the level of variety of color, motion, size, or orientation in the background. It is ranked by the evaluator. For example, if the background has many different colors, then the background color homogeneity is low. If the background contains visual units with very similar orientations, then the background orientation homogeneity is high.

The target-background distance is determined differently for color, motion, size, and orientation. For color, it's the distance between the target color and the closest background color. In most cases, such distance can be estimated by evaluators. But more accurate calculation can be performed through the following equation.

$$c = \left(\left| \frac{\sum_{i=1}^{n} T_r}{N} - D_r \right| + \left| \frac{\sum_{i=1}^{n} T_g}{N} - D_g \right| + \left| \frac{\sum_{i=1}^{n} T_b}{N} - D_b \right| \right) / 765 \qquad (1)$$

where T_r, T_g, and T_b are the R/G/B values of target color, D_r, D_g, D_b are the R/G/B values of the closest distracter color. The N and n are the number of color components represented in the visual display. Number 765 is the distance between the two most distant colors.

Similarly, the difference between the target motion speed and the closest distracter speed can be estimated. So does the difference between the size of target pattern and the closest size of all the distracters, as well as the difference between the orientation of the target pattern and the closest orientation of all the distracters. In the end, the evaluator would rank the distance between the target and distracters as high, medium, or low for color, motion, size, and orientation respectively. Then the target-distracter difference scores can be looked up from table 2.

4 Case Study

We have applied the proposed complexity analysis method to evaluate a number of computer security visualization tools, including NvisionIP [11], VisFlowConnect [12], and RUMINT [13]. To evaluate each tool, we downloaded the programs and used them to visualize sample data sets provided by their developers. Here we present our complexity analysis for NvisionIP, a network traffic visualization tool that allows users to conduct security analysis for large and complex networks. It was developed by Security Incident Fusion Tool (SIFT) research team at the University of Illinois at Urbana-Champaign.

The NvisionIP interface can be divided into three workspaces: overview (figure 3), middle level view (figure 4), and detailed view (figure 5).

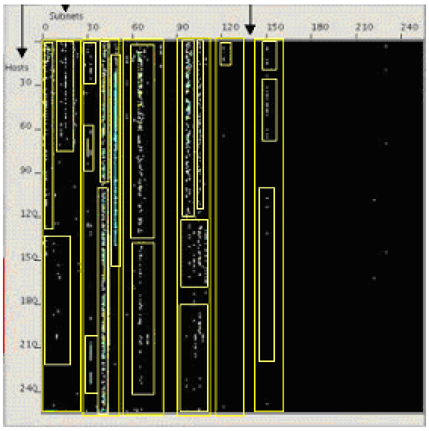


Fig. 3.

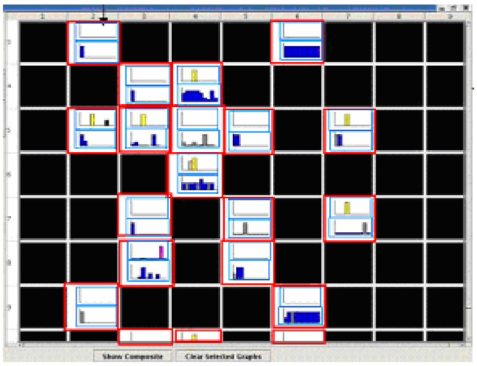


Fig. 4.

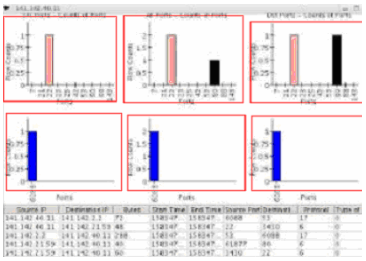


Fig. 5.

The visual integration tree for NvisionIP is shown in figure 6. In figure 3 to 5, the identified visual patterns are marked by boxes. Note that the numbers in the tree are upper bounds of visual integrations. From the visual integration tree, we can see that the middle level view is the most complex in terms of visual integration.

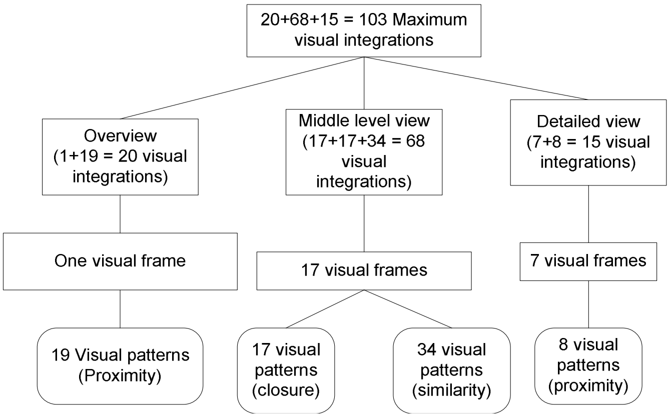

20+68+15 = 103 Maximum visual integrations
Overview (1+19 = 20 visual integrations)
Middle level view (17+17+34 = 68 visual integrations)
Detailed view (7+8 = 15 visual integrations)
One visual frame
17 visual frames
7 visual frames
19 Visual patterns (Proximity)
17 visual patterns (closure)
34 visual patterns (similarity)
8 visual patterns (proximity)

Fig. 6. Visual integration tree for NvisionIP

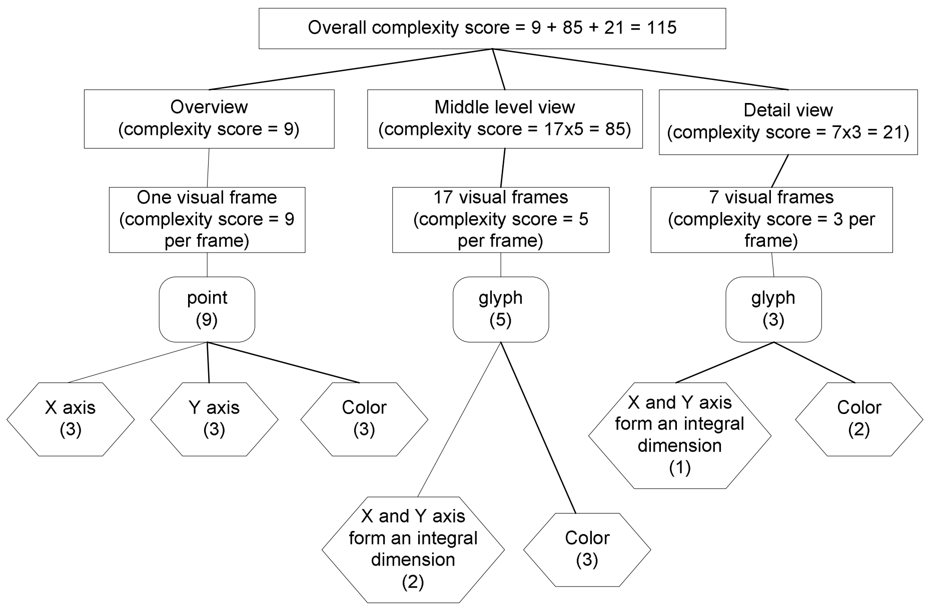

Overall complexity score = 9 + 85 + 21 = 115
Overview (complexity score = 9)
Middle level view (complexity score = 17x5 = 85)
Detail view (complexity score = 7x3 = 21)
One visual frame (complexity score = 9 per frame)
17 visual frames (complexity score = 5 per frame)
7 visual frames (complexity score = 3 per frame)
point (9)
glyph (5)
glyph (3)
X axis (3)
Y axis (3)
Color (3)
X and Y axis form an integral dimension (1)
Color (2)
X and Y axis form an integral dimension (2)
Color (3)

Fig. 7. Visual mapping complexity tree for NvisionIP

Figure 7 shows the visual mapping complexity tree for NvisionIP. The numbers in the parentheses in the dimension boxes are the complexity scores calculated based on table 1. The complexity scores for visual units are the sum of complexity scores of their separable dimensions, and the complexity score of visual frames are the sum of complexity scores of different types of visual units, and so on. In the overview workspace, X and Y axis are considered separable dimensions because the X axis is mapped to subnets and Y axis is mapped to hosts. This mapping is unintuitive, and we expect that users will have to process them separately. On the other hand, the mapping of X and Y axes in the middle and detailed views are rather intuitive, so they are considered integral dimensions. The complexity scores are an estimate of the cognitive load on working memory for interpreting different types of visual units. From figure 7, we can see that the color points in the overview workspace require the most mental effort to interpret. We would recommend replacing the mapping of X and Y axes with more intuitive mappings of subnet and hosts. Again, the middle level view is the most complex because of the large number of separable dimensions. We would recommend redesign the middle level view to reduce its complexity.

Finally, we calculate the target-distracter difference scores as follows.

- For the overview workspace, we identify that the potential target is a group of points. In this case, the background homogeneity is high, and the color distance between target and distracter is medium. Based on table 2, we give a score of 3.
- For the middle level view workspace, we identify that the potential search target is one of the small frames. In this case, the background homogeneity is low because there are many different visual frames. The distance of size, orientation, and color between target and distracters are also low. So we select a score of 1.

Table 3. Target-distracter difference scores for NvisionIP

	Target-distracter difference scores
Overview	3
Middle level view	1
Detailed view	5

- In the detailed view, the likely visual search target is a bar. In this case, both the background homogeneity and target-background color distance is high, so we select a core 5.

We find that the middle level view has the lowest target-distracter difference score, which suggests that the middle level view is the most complex for visual search.

5 Conclusion and Future Work

We have presented a systematic methodology to analyze the complexity of visualization. Here the complexity is measured in terms of visual integration, number of separable dimensions for each visual unit, the complexity of interpreting the visual attributes, and the efficiency of visual search. These measures are derived from well established psychological theories. Together they indicate the amount of cognitive load for comprehending a visualization design.

The complexity analysis is particularly useful during the design phase before any user studies can take place. It helps developers quickly evaluate different design choices. It can also be used to generate hypotheses to be tested in user studies.

Finally, we have demonstrated this method by using it to analyze a computer security visualization program. The analysis helps us identify a number of issues with the design of this visualization. We are currently applying and refining this complexity analysis in our own visualization research in the fields of computer security and bioinformatics. In the future, we plan to conduct user studies to correlate the complexity analysis with user performance, and therefore validate and improve the accuracy of this method.

Acknowledgement

The authors would like to thank Hsiu-Chung Wang for her contribution to an early version of the proposed method.

References

1. Clark, R., Nguyen, F., Sweller, J.: Efficiency in Learning: Evidence-Based Guidelines to Manage Cognitive Load. Pfeiffer (2006)
2. Garner, W.R.: The processing of information and structure. Lawrence Erlbaum, Mahwah (1974)

3. Trafton, J.G., Kirschenbaum, S.S., Tsui, T.L., Miyamoto, R.T., Ballas, J.A., Raymond, P.D.: Turning pictures into numbers: extracting and generating information from complex visualizations. International Journal of Human-Computer Studies 53, 827–850 (2000)
4. Wertheimer, M., King, D.: Max Wertheimer and Gestalt Theory: Transaction Publishers (2004)
5. Wolfe, J.M., Horowitz, T.S.: What attributes guide the deployment of visual attention and how do they do it? Nature Reviews/Neuroscience 5, 1–7 (2004)
6. Bertin, J.: Semiology of Graphics: University of Wisconsin Press (1983)
7. Brath, R.: Metrics for effective information visualization. In: Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis) (1997)
8. Zhou, M.X., Feiner, S.K.: Top-Down Hierarchical Planning of Coherent Visual Discourse. In: Proceeding of International Conference on Intelligent User Interface (IUI): ACM (1997)
9. Larkin, J.H., Simon, H.A.: Why A Diagram is (Sometimes) Worth Ten Thousand Words. Cognitive Science 11, 65–99 (1987)
10. Kosslyn, S.M.: Graphics and Human Information Processing: A Review of Five Books. Journal of the American Statistical Association 80, 499–512 (1985)
11. SLakkaraju, K., Yurcik, W., Lee, A.J.: NVisionIP: netflow visualizations of system state for security situational awareness. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (2004)
12. Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: VisFlowConnect: netflow visualizations of link relationships for security situational awareness. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (2004)
13. Conti, G.: http://www.rumint.org/ (2007)

A GPU Framework for the Visualization and On-the-Fly Amplification of Real Terrains

Yacine Amara[1], Sylvain Meunier[2], and Xavier Marsault[3]

[1] USTHB, Algeria
[2] SIC – Signal Image Communications, France
[3] MAP-ARIA, UMR CNRS 694, France

Abstract. This paper describes a GPU framework for the real-time visualization of natural textured terrains, as well as the steps that are needed to populate them on-the-fly with tens of thousands of plant and/or mineral objects. Our main contribution is a robust modular architecture developed for the G80 and later GPUs, that performs texture/seed selection and rendering. It does not deal with algorithms that procedurally model or render either terrain or specific natural objects, but uses them for demonstration purposes. It can be used to calculate and display realistic landscapes and ecosystems with minimal pre-stored data, CPU load and popping artefacts. It works in tandem with a pre-classification of available aerial images, and makes it possible to fine-tune the properties of objects added to the scene.

1 Introduction

In real-time terrain visualization, altimetric and photographic database resolution quickly becomes insufficient when the viewpoint approaches ground level. While limiting memory load and computing costs, one may add, as soon as necessary, geometrical and textural details for the ground, resulting in the emergence of sets of plant and mineral objects adapted to the viewpoint requirements. Obtaining such an enriched reality from restricted data sets is known as “amplification”. The term was first mentioned in [23]. This feature is important in many applications dealing with terrain rendering (e.g. flight simulation, video games).

Earth navigators such as Google Earth are designed to stream and display digital models of real textured terrains without amplification. On the other hand, Microsoft “Flight Simulator”, without providing access to real aerial textures, implements a kind of amplification by mapping pre-equipped generic patterns onto the ground. A first attempt to reconcile the two approaches was made with Eingana [10] by the French company EMG in 2001. Based on limited-resolution databases, and using various analytical tools, Eingana generated a fairly realistic 3D planet by fractal amplification. But all such applications suffer from inaesthetic visual popping artefacts. Level-of-detail management on the GPU is therefore an important challenge in this context.

Seed models for terrain amplification such as [26] have already been presented, but not as full GPU real-time systems. We attain this goal on the G80 GPU by the use of robust techniques (render-to-texture, asynchronous transfer, multi-render-target, stream reduction, instancing). This is our main contribution in the present paper.

After a short overview of previous work (section 2), we give an outline of our technique (section 3), then exhaustive details of our implementation (section 4). For our experimentation, we use databases from the Haute-Savoie region in the French Alps (a large area of 4,388 km^2) provided by [22]: a 16 m/vertex “digital elevation model” (DEM) and a set of 50 cm/pixel aerial textures. The results are presented and discussed in section 5, followed by the conclusion.

2 Previous Work

2.1 Data Structures for Large-Area Texture Management and Visualization

Terrain navigators currently handle tens of GB of geo-referenced textures, whereas the GPU memory barely reaches one GB. Long before graphics cards became programmable, efficient main-memory resident data structures for visibility and hierarchical optimization [28], and corresponding CPU algorithms, were designed to provide the textures strictly necessary to display the current viewpoint. This always imposed constraints on the ground mesh, except for clipmaps [25]. Little work has been done on the handling of such structures on the GPU. But in 2006, A.E. Lefohn’s [16] describes a complex C++ library that would handle them in this way. In 2005, S. Lefebvre shows in [15] how to work in a “virtual texture space” on the GPU, for arbitrary meshes. He also builds a streaming architecture for progressive and dynamic texture loading. This framework is used to traverse virtual-texture tile structures by carrying out all the complex and expensive computations on the GPU.

2.2 Seed Models

Seed models are extremely useful in the procedural specification and instantiation of landscapes. They avoid the need to precompute and store millions of plant and mineral objects just to enrich a few square kilometers. Exploiting the high redundancy that is present in nature, they make it possible to populate terrain on the fly without storing all the objects and their properties, while preserving the variety of the results.

We consider three levels of description for seeds. A single seed manages the placement of each object. It is described by a property vector: species, position on the ground, size, orientation, color, level of detail (lod). A seed cluster contains some closely-spaced objects which are processed as a single seed. Metaseeds are seeds of seeds, and store properties such as object density and activation radius. Visible objects in aerial textures (trees, shrubs, rocks) are described on the first two levels, while smaller objects (plants, flowers, piles of stones, etc.) that are invisible and numerous justify a “metaseed” description. At the seed level, it is easy to take into account geographical (altitude), climatic (moisture), seasonal (snow in winter), temporal (growth, ageing) rules, using a parameterized seed model. At the overall level, botanical and biological rules may be added to synthesize spatial distributions and specify metaseeds. Seed patterns are used to store such distributions and compose virtual patchy landscapes. Well described in [8], [9], [17], [15], they usually consist of squared generic sets of seeds or seed clusters. They can store one or more species, which makes them useful as a way of simulating varied and adaptable ecosystems. Random or controlled variations can be introduced on this level to avoid strict repetition of generic fittings. At the pattern-instancing level on the ground, distributions usually take account of local species density. The use of patterns has several advantages: genericity, storage saving, reduced number of primitives sent to the graphics card. A first GPU implementation of this technique to generate random forests can be found in [15].

2.3 Real-Time Plant Rendering

Some of the tools – e.g. [1], [18] – which are used to generate realistic 3D plant models, are unsuited to real-time rendering of dense scenes. Alternative representations (volumetric textures, image-based or point-based approximations) which are more efficient, but also maintain good visual quality, have been designed. Volumetric textures were introduced by A. Meyer and F. Neyret [19] and improved by P. Decaudin [6], making it possible to render several thousand trees in real time. The method presented in [27] for real-time grass rendering with patches uses a combination of geometry-based and volume-based approaches. In both cases, it is difficult to reach the individual level of objects, which rules these methods out for seed rendering in the context of amplification. The point-rendering method proposed by G. Gilet [12] borrows some good ideas from [5] and provides continuous detail management. It reduces the number of points when the “tree to viewer” distance increases, and retains the original object geometry for close-up viewpoints. But the algorithm remains ill-adapted to high frame rates. Simple billboarding has been widely used for plants in real-time applications, originally to replace a set of polygons by a rectangle facing the user, onto which a semi-transparent texture is projected, representing the plant for a given viewpoint. Unfortunately, the objects lack relief and parallax, even if two crossed polygons or view-dependent textures are used. Billboard clouds, by creating volume, improve realism, but not thickness. X. Decoret has proposed in [7] a simplification method for generating sets of static billboards that approximate plant geometry, offer a good parallax and give an impression of depth. And A. Fuhrmann has improved his algorithm to cope better with trees [11]. He can generate realistic models of a few dozen billboards, with several levels of detail.

3 Outline of Our Technique

Our GPU architecture is dedicated to the realtime visualization and amplification of natural textured terrains, based on 4 modules. We suppose our ground to be covered by a virtual regular grid (RG) of tiles (Fig. 2). Each module is associated with a GPU rendering pass at a different level of abstraction: “tile selection”, “ground texture rendering”, “seed selection” and “seed rendering”. This framework is generic in that it can be used with different algorithms for terrain mesh and seed-instance rendering, and the corresponding passes deal with arbitrary ground meshes, since they work at the GPU vertex level. For this purpose, we adapted the texture-streaming architecture [15] to large terrains, and improved it in many aspects (section 4).

Following [15], a module called TLM (Tile Load Map) computes the useful tiles lists for the viewpoint in one render-to-texture pass. It also deals with their visibility in the frustum, occlusion and level of detail. This information is sent in a 2D texture to a second module which stores, analyzes and compares the lists for frame coherence, and decides which tiles to send to the GPU. Another module loads the required textures asynchronously into protected "caches" on the GPU memory. During the second pass, the ground geometry is sent to the GPU to be fully textured.

The on-the-fly terrain amplification concept does not allow for the pre-storage of seeds. We therefore developed an adaptive seed model - inspired by previous publications on virtual ecosystem simulation – in which seed placement is constrained by the aerial images, whose visible detail must be finely restored. We also ensure a color relationship between aerial images and the generated objects. Upon the launch of the application, a set of P generic square patterns containing random virtual seeds is produced (only their 2D positions are computed), and stored in the GPU memory. We design patterns whose number of seeds approximate the plant or mineral density for a typical zone. Plant and mineral objects are also cached in the GPU memory.

But there is no existing real-time method for computing "land cover classification" (LCC) for each pixel. Inspired by the works of S. Premoze [21], we use a pre-classification step for aerial images, which is the current trend. It draws on recent co-operation with INRIA\MISTIS [4] and IGN\MATIS [24] using statistical modelling applied to image analysis. We demonstrate in section 4.4 how to use LCC types (in a density-map like approach) to extract seeds on the fly and produce a plausible scene. Only the "single seed" model is fully implemented in our work.

4 Implementation Details

Ground textures are packed into a quadtree made up of 512 x 512 pixels tiles, stored on disk. Each one has its own mipmap pyramid, compressed into a 1:8 lossy format DXT1 (to minimize texture-volume transfer and GPU decompression time). The ground geometry of each frame (computed by the "terrain algorithm" – see Fig. 2) is cached in a VBO during an initial "dummy rendering pass", while the depthmap is also stored in the GPU memory (for use in sections 4.2 and 4.4 with a texture access). The terrain heightmap texture is loaded onto the GPU for an instant access in pass 3.

4.1 Encoding and Packing LCC Types in a Quadtree

We must store them in a quickly accessible data structure. Using the terrain vertex structure (16 m spacing) would involve an excessive loss of precision in the seed placement. It is preferable to store LCC types in textures, on disk, for access in a quadtree, since we have to retrieve them at different levels of detail (trees being visible up to 3,000 m, their LCC types are spread over multiple levels in the quadtree).

The coding scheme (Fig. 1) is not commonplace, because ground textures mapped to the terrain are not necessarily of the highest resolution at which we have to locate the LCC types. All these constraints impose a redundant coding that should be consistent between all the relevant levels, if popping objects are to be avoided. Moreover, we cannot pack ground colors and LCC types together in the same 32-bit RGBA tiles, because the latter must not be compressed.

In sum, a specific "LCC quadtree" is needed to pack LCC types into luminance tiles. We use 3 levels for trees. To ensure consistency between these different levels, we propose the following coding method, which allows a 1 m positioning precision on the ground (which is sufficient for mountains). Level 1 uses 8 bits to store each LCC type resulting from the classification. We duplicate these types on 4 neighbors at level 0. Lastly, level 2 uses 8 bits to encode 4 neighboring texels from level 1, meaning that 3 different species can be stored on 2 bits (0 is kept for bare ground). We can increase the coding precision and the number of species supported by giving more bits to the LCC texture format. With 16 (resp. 32) bits, 15 (resp. 255) species can be addressed, although this means using more GPU memory and bus bandwidth.

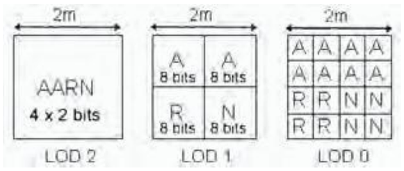


Fig. 1. LCC type quadtree-packing diagram (A, R and N are examples of LCC types)

4.2 The "Tile Load Map" (Pass 1)

This section deals with the core component of the tile-streaming architecture: the "Tile Load Map (TLM)" algorithm (Fig. 2). This module is dedicated to the selection of all the geo-referenced tiles needed to display the current viewpoint: ground tiles, associated LCC types and an aperiodic set of seed pattern indices. Since all are designed to share the same size, the computation is performed in a unique render-to-texture pass, using the subtle vertex (VS1) and fragment (FS1) programs, as described below. It produces the TLM: a 4-component texture of the same size as the RG is read back to the CPU (this is obligatory). After an analysis step, the TLM is expressed in 3 maps: the "Ground_texture Load Map (GLM)", the "Classification Load Map (CLM)" and the "seed_Pattern Load Map" (PLM), which can transmit the appropriate tiles and pattern indices to the GPU, with minimal OpenGL calls.

First, as described in [15], the ground geometry is drawn in the virtual texture space of the TLM corresponding to the entire terrain. Global texture coordinates (u_g, v_g) are computed in VS1 on the basis of 2D world vertex coordinates, which are sent to FS1 in the POSITION semantic. The corresponding vertex screen coordinates (SC) are computed with the ViewProject matrix for the current viewpoint. We also calculate the distance D between the viewpoint and the current vertex. SC and D are sent in TEXCOORDi semantics. The TLM computes 4 features for each tile: a visibility index (v_index) and 3 lod parameters (tlod_min, tlod_max, radial_lod).

Visibility_index. In FS1, up to 6 clipping planes are used to cull the ground geomentry polygons in texture space. SC coordinates are used in an occlusion-culling test to access the depthmap and discard hidden tiles. All culled tiles return a 0 value for the v_index. Strictly positive values, corresponding to visible tiles, are then used to store up to 255 possible seed pattern indices, which are computed on the fly, for each frame. In fact, this is an intuitive, and indeed better, solution than precomputing them on the CPU, for two reasons : it avoids storing memory-consuming arrays in the CPU or GPU for large terrains, and it allows some geographical, seasonal and temporal rules to be taken into account for optimal placement in VS1. Up to now, we

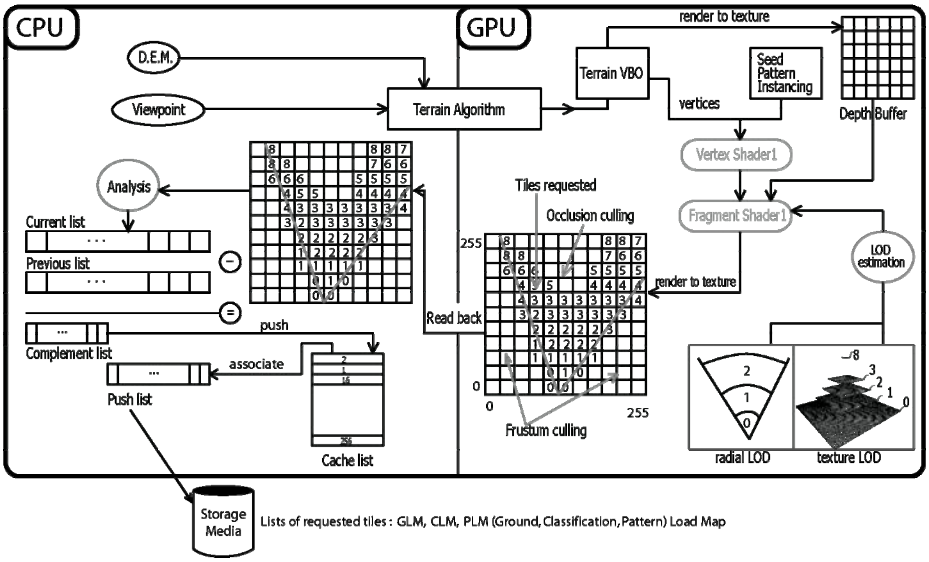


Fig. 2. The "Tile Load Map" process for the current viewpoint

have used a GPU-translation of the Park and Miller space-time-coherent, aperiodic, pseudo-random generator described in [20]. Each time the application is launched, the same seeds appear at the same place, depending only on a germ choice.

LOD estimation. Two different mechanisms are used in FS1 to estimate the required tile lods. For the ground tiles, we use the direct mipmapping estimation provided by the GPU to minimize the amount of textures needed to display the current viewpoint. For this purpose, a dedicated "LOD-texture" is created and cached on the GPU. Its mipmap pyramid is filled with integer values representing rising levels, starting from 0. In FS1, a simple access to this texture with SC coordinates gives the current lod value for each fragment. Here, we have to take account of a corrective parameter related to screen resolution in order to produce good mipmap estimations in texture space.

As regards LCC tiles and nested seed patterns, we use a different technique, given that the mipmapping mechanism is not suited to them: a user on the ground or at the height of a peak is likely to obtain horizontal polygons whose high lod values would result in erroneous LCC readings, and prevent the display of the trees. To eliminate this problem, the distance D is used to evaluate a radial lod: when a pattern is used at a distance of less than 3 000 m, a radial selection is computed for the first 3 lods of the LCC quadtree, with thresholds of 700 m, 1 400 m and 3 000 m. The fourth component of the structure is then used to store this radial level (0, 1 or 2), value 3 being reserved for non-active patterns.

TLM readback to CPU memory. The 4 TLM features are written in the output structure of FS1 with the "max blending" and polygon antialiasing options turned on (respectively for cumulative results and conservative rasterization per tile). To optimize transfers, we use the 8-bit/component BGRA texture format.

TLM analysis on the CPU. This stage simultaneously manages lists of tiles likely to be downloaded to the GPU (in one or several ‘Texture-cache’) with conservation of temporal coherence. It optimally ensures the nesting between useful tiles extracted from the GLM/CLM and the corresponding tiles in their respective quadtrees. Here, the tlod_min and tlod_max that estimate the lod range of a tile are used in conjunction with the v_index to select grouped tiles in the ground quadtree and build the “currentList”. We maintain Texture-cache coherence by using a stack containing free positions in the “cacheList” (Fig. 2). Comparisons between the currentList and the previousList result in a third list called “pushList”, which stores the indices of textures to be loaded up from the hard disk. Unused locations in the Texture-cache are pushed into “cacheList” using a “complement list”. Given that on the G80, 8192 x 8192 pixels are available for Texture-cache, no overflow management for “cacheList” is necessary, and this saves computing time on the CPU. At the end of this pass, the required seed-patterns are grouped by type, so as to optimize the number of VBO calls in pass 3.

4.3 Ground-Tile Rendering (Pass 2)

The required ground tiles are first downloaded into the corresponding Texture-cache, which manages a tile pool of up to 256 locations on the G80. A rectangular luminance texture coding the quadtree indices is sent to the FS2. For texturing the current fragment, (u_g, v_g) are processed in a loop between tlod_min and tlod_max with a series of scaling and translating transformations, so as to obtain the associated tile location in Texture-cache, and the corresponding relative coordinates. The fragment color can then be read. Our solution to solve the discontinuity problem that appears in mipmapping in this cache involves a simple process using precomputed borders for the internal tiles.

4.4 Visible Seed Selection (Pass 3)

To retrieve visible seeds with minimal computing time, only visible patterns (in a radius corresponding to the visibility range of a tree) are extracted and stored in the PLM (section 4.2). In the same way as for the tiles, the graphic pipeline is then programmed for a “one-pass rendering” in a 2D texture called SLM (Seed Load Map). The size of this texture is at most 512 x 512 pixels, which can store enough seeds to populate large scenes. All the seeds are processed by the graphics card, and sent from the PLM by OpenGL calls to the patterns, which are 1D arrays initially cached on the GPU (Fig. 3) in vertex buffer objects (VBO). Each seed stores 8 GPU-computable properties: 3D position on the ground, LCC type (species), surrounding sphere (center, radius), level of detail, size, orientation, ground color, and an “index” within the VBO. The surrounding “sphere” qualifies the seed instance, whose center is placed at mid-height from it. The “ground color” is designed for a colorimetric follow-up of the terrain (sections 4.4 and 4.5). Pseudo-random functions are introduced in order to disturb size, orientation and color, and to obtain non-redundant variations, as in nature. Except for the “index” (which is used to compute the output position in the SLM), scalar heights are transmitted to the output structure of the fragment program FS3, for use in pass 4. But, a texture can store up to 4 floating components. So the

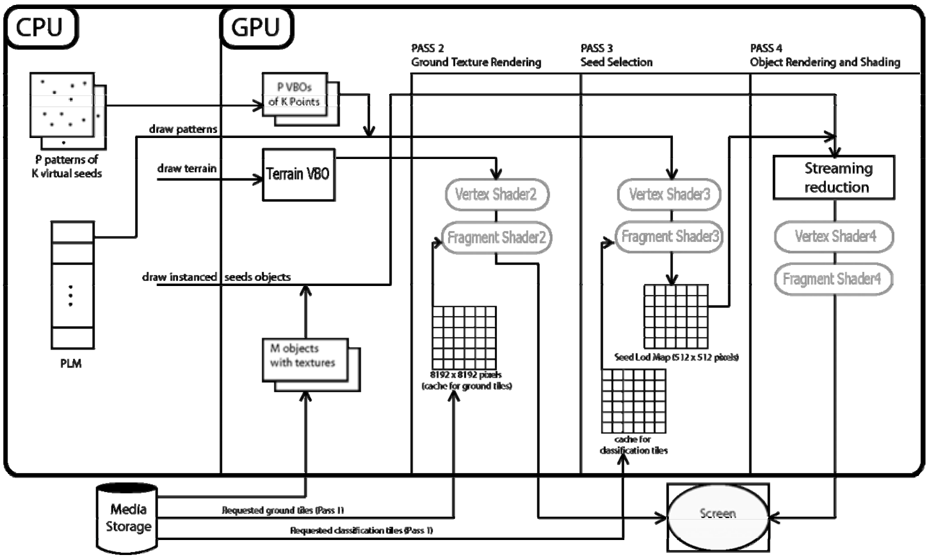


Fig. 3. Details of the seed and ground rendering passes

ARB_draw_buffers extension is enabled in order to activate the Multi Render Target (MRT). FS3 returns ($+\infty$, 0,…, 0) if the corresponding seed is invalid, hidden, or outside the viewing frustum. The five following steps describe in details how seeds can be selected and stored in the SLM.

Seed positioning on the ground. Generic patterns only store seed locations (x, y), and so seeds are first positioned in 2D, using translations performed in VS3. This is followed by an orthogonal projection onto the ground mesh to compute the z value. Since the ground mesh is processed by a multiresolution algorithm, values for z may be view-dependent, except when the viewpoint is close to the ground. To avoid visual artefacts, all values of z must be the same, whatever the level of detail of the ground. We decided to anchor objects to the ground on their smallest z component. Each seed in the ground heightmap being surrounded by at most 4 basic vertices, its anchor point is obtained by a bilinear interpolation on 4 z values. Lastly, we ensure that the lower part of the trunk goes down a little way into the ground. This technique is particularly suited to amplified terrain meshes with detailed heightmaps whose amplitude is known in advance.

Seed visibility. This takes place in two steps, like the TLM process. VS3 performs a culling test: a seed whose surrounding sphere is totally exterior to the frustum is removed. Let R be the transformed radius of the sphere (taking account of on-the-fly random "size" computations), and ε a tolerance value. We project the center of the sphere into the "Normalized Device Coordinate" (NDC) space, then check if it belongs to $[-1-R-\varepsilon, 1+R+\varepsilon] \times [-1-R-\varepsilon, 1+R+\varepsilon] \times [0, +\infty[$. The occlusion-culling test is performed in FS3, and discards a seed hidden by the ground (behind a mountain for example) if its depth is greater than the value stored in the depthmap. For maximal precision, this value is computed at the "summit" position of the object projected onto the screen. For distant seeds, it is necessary to take account of the mesh simplification performed by the terrain multiresolution algorithm. An empirical tolerance distance of 50m is added so that visible objects will not be lost. Lastly, we enable the GL_CLAMP for the nearest mode of the depthmap texture, because an object whose summit does not appear on the screen may not have an associated depth.

Lod computation. This is performed by the VS3, which outputs a radial lod evaluation for each seed combined with its real size, so as to take into account the visual print during the lod transitions.

LCC value. The virtual seed species is instanced in FS3 for each seed : we access values in the LCC quadtree using the encoding scheme and the methods set out in sections 4.1 and 4.3. FS3 discards the current seed and returns $(+\infty, 0,\ldots, 0)$ if no valid LCC type is found.

Ground color. This is accessed in FS3 using the same method given for ground-tile rendering in section 4.3. A random sampling of ground colors is carried out in the neighborhood of each seed. This allows fine modulations of plant or mineral colors to take place in pass 4.

4.5 Visible Object Rendering (Pass 4)

Although selection and rendering algorithms are independent, five parameters (stored in the SLM) ensure their nesting: species, size, orientation, lod and ground color. Our aim is to render selected entities with multiple hardware instancing calls, while avoiding a complete SLM readback in the CPU. Given M mineral and V plant objects, expressed in N_M and N_V lods, the number of instancing calls is $(M.N_M + V.N_V)$. To specify the "instancing stream" of each call, we just have to know the number of primitives to send, with instancing properties being accessed in VS4 with SLM readings. But the SLM contains not only "visible seeds" but also more numerous "holes", corresponding to discarded seeds. Since the SLM size is known, VS4 and FS4 could in fact directly process all vertices and fragments coming from seeds, and discard those coming from "holes", but this would be totally ineffective. A better way to tackle the problem is to use a GPU "stream reduction" algorithm, like the one described in [13]. This has several advantages: filtering capabilities that sort seeds by type and pack them without holes, and access to the number of packed seeds, which is transmitted to the CPU at very negligible cost using a readback of very few pixels.

The colorimetric ground follow-up naturally takes account of shadowed zones, and reduces unnecessary computations. The colorimetric transition is performed by a smooth blending with the ground textures at the rendering step of each instance. The foliage and trunks of the trees are processed independently, and the luminosity of the texels is modulated according to their relative positions within the object. Clouds (based on dynamic time-morphing textures) are also projected onto these objects and the ground textures, for more realism. All shading is performed in FS4.

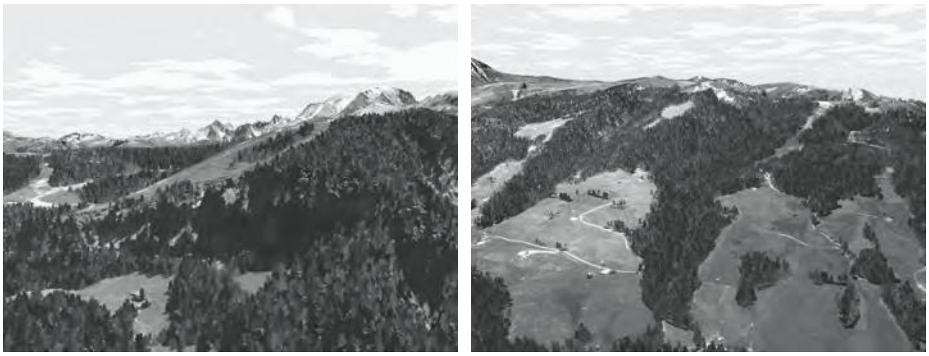


Fig. 4. Two viewpoints, at different distance, of the full textured and amplified terrain (with the permission of use from “Régie de Données des Pays de Savoie RGD73-74”)

5 Experiments and Results

All the codes were developed in C++, OpenGL, Cg and GLSL. The following tests were performed at 1280 x 1024 (res 1) and 1024 x 768 (res 2) display resolutions, on a computer with a 2.93 GHz CPU, 2 GB RAM and a Geforce 8800GTX. Only trees were used for the experiments: the maximum forest density is 21,000 trees/km^2. Computing times for the 4 passes are shown in Table 1 and give an overall impression of the performance of our system. For demonstration purposes, RMK2 [3], based on an improved version of the SOAR algorithm [17], was the library chosen to carry out the ground-geometry rendering. An improved implementation of our terrain algorithm using geometry clipmaps [2] is in progress, and will provide higher frame rates.

Table. 1. Performance analysis of the 4 passes, for 3 typical viewpoints and a large number of checked seeds per frame. Computing time for pass 3 depends only on the number of checked seeds which is related to the overall plant density. The influence of resolution is obvious for pass 2 and pass 4 results, the latter being the most time-consuming part of our seed architecture. The ratio selection/generation time of models to rendering time may be improved later. Global comparisons with CPU techniques are difficult, since few other works have been carried out in this field. Nevertheless, a rapid study shows that the TLM and SLM algorithms on the GPU seem almost 10 times faster than their CPU implementation.

Viewpoint	1		2		3	
Resolution	1	2	1	2	1	2
Terrain mesh (ms)	3.84	3.5	4.55	4.37	1.2	1.1
Pass 1 (ms)	**1.42**	**1.4**	**1.56**	**1.49**	**1.19**	**1.16**
Pass 2 (ms)	**2.04**	**1.78**	**2.23**	**1.82**	**1.62**	**1.4**
Pass 3 (ms)	**2.87**	**2.78**	**2.99**	**2.83**	**2.5**	**2.4**
Pass 4 (ms)	**4.52**	**3.98**	**5.91**	**5.03**	**3.08**	**2.82**
Selected seeds	5 102 / frame		23711 / frame		3164 / frame	
Checked seeds	115 712 / frame		141 728 / frame		101 227 / frame	
FPS (Hz)	**68**	**74**	**58**	**64**	**104**	**112**

For trees, we use billboard clouds (our special thanks go to A.Fuhrmann) with 3 lods composed respectively of 17, 8 and 2 polygons textured with 16-bit luminance atlas images of at most 4096 x 2048 pixels.

We process and read back a 256 x 256 pixel TLM in pass 1: this does not really affect the performance of the application, provided that this size is not exceeded. Frustum and occlusion culling substantially reduce the computing costs, both in CPU and GPU, especially for pass 3. In pass 2, the transition between levels 0 and 1 of the ground quadtree occurs at an optimal 700 m distance from the ground (corresponding to 1 pixel = 1.14 texel). With this adjustment, two mipmap levels per tile only, and the 4x anisotropic filter activated, we obtain good display quality (Fig. 4), and at a low cost, without overloading the graphics bandwidth.

6 Conclusions and Prospects

We are putting forward a robust and complete GPU approach in 4 passes for the visualization and on-the-fly population of large-textured (and even ground-mesh-amplified) terrains in real time, with tens of thousands of natural elements. We use a pre-classification of ground textures, and powerful streaming and instancing algorithms. For each frame, the selection of textures and seeds is performed entirely on the GPU, driven by an optimal combination of two visibility algorithms. The choice of the rendering algorithms is thus independent of the previous step. Our animations contain no popping effects that could be due to aggressive LODs or rough simplifications. One of our future tasks will be to process “clusters” and “metaseeds” in order to obtain detailed ecosystem display. A challenge will be to apply smoothly-controlled on-the-fly amplification techniques to all textures, in conjunction with seasonal behavior. The TLM algorithm will also be extended to deal with planetary systems.

References

1. AMAP - botAnique et bioinforMatique de l’Architecture des Plantes, http://amap.cirad.fr/
2. Asirvatham, A., Hoppe, H.: Terrain Rendering Using GPU-based Geometry Clipmaps. In: GPUGems2, Addison-Wesley, Reading (2005)
3. Balogh, A.: Real-time Visualization of Detailed Terrain. Thesis of Automatic and Compuing University of Budapest (2003)
4. Blanchet, J., Forbes, F., Schmid, C.: Markov Random Fields for Recognizing Textures Modeled by Feature Vectors. In: International Conference on Applied Stochastic Models and Data Analysis, France (2005)
5. Dachsbacher, C., Vogelgsang, C., Stamminger, M.: Sequential Point Trees. ACM Trans. On Graphics (2003)
6. Decaudin, P., Neyret, F.: Rendering Forest Scenes in Real-Time. In: Eurographics Symposium on Rendering, Norrköping, Sweden (2004)
7. Decoret, X., Durand, F., Sillion, F.X., Dorsey, J.: Billboard Clouds for Extreme Model Simplification. In: Proceedings of the ACM Siggraph (2003)
8. Deussen, O., Colditz, C., Stamminger, M., Drettakis, G.: Interactive Visualization of Complex Plant Ecosystems. In: Proceedings of the IEEE Visualization Conference, IEEE Computer Society Press, Los Alamitos (2002)

9. Deussen, O., Hanrahan, P., Lintermann, B., Mech, R., Pharr, M., Prunsinkiewicz, P.: Realistic Modeling and Rendering of Plant Ecosystems. In: Computer Graphics, Siggraph (1998)
10. Eingana: Le premier atlas vivant en 3D et images satellite, Cdrom, EMG (2001)
11. Fuhrmann, A., Mantler, S., Umlauf, E.: Extreme Model Simplification for Forest Rendering. In: Eurographics Workshop on Natural Phenomena (2005)
12. Gilet, G., Meyer, A., Neyret, F.: Point-based Rendering of Trees. In: Eurographics Workshop on Natural Phenomena (2005)
13. Roger, D., Assarson, U., Holzschuch, N.: Whitted Ray-Tracing for Dynamic Scenes using a Ray-space Hierarchy on the GPU. In: Proc. of the Eurographics Symp. on Rendering (2007)
14. Lane, B., Prunsinkiewicz, P.: Generating Spatial Distributions for Multilevel Models of Plant Communities. In: Proceedings of Graphics Interface (2002)
15. Lefebvre, S.: Modèles d'Habillage de Surface pour la Synthèse d'Images. Thesis of Joseph Fourier University, Gravir/Imag/INRIA. Grenoble, France (2005)
16. Lefohn, A.E., Kniss, J., Strzodka, R., Sengupta, S., Owens, J.D.: Glift: Generic, Efficient, Random-access GPU Data Structures. ACM Transactions on Graphics (2006)
17. Lindstrom, P., Pascucci, V.: Terrain Simplification Simplified: a General Framework for View-dependant Out-of-core Visualization. IEEE Trans. on Vis. and Comp. Graphics (2002)
18. Lintermann, D., Deussen, O.: Xfrog, http://www.xfrogdownload.com
19. Meyer, A., Neyret, F.: Textures Volumiques Interactives. Journées Francophones d'Informatique Graphique, AFIG, 261-270 (1998)
20. Park, S.K., Miller, K.W.: Random Number Generators: Good Ones Are Hard To Find. Com. of the ACM 31, 1192–1201 (1998)
21. Premoze, S., Thompson, W.B., Shirley, P.: Geospecific Rendering of Alpine Terrain. Department of Computer Science, University of Utah (1999)
22. RGD73-74: Régie de Gestion des Données des Deux Savoies, http://www.rgd73-74.fr
23. Smith, A.R.: Plants, Fractal and Formal Languages. In: Proceedings of Siggraph (1984)
24. Trias-Sanz, R., Boldo, D.: A High-Reliability, High-Resolution Method for Land Cover Classification Into Forest and Non-forest. In: 14th Conf. on Image Analysis, Finland (2005)
25. Seoane, A., Taibo, J., Fernandez, L.: Hardware-independent Clipmapping. In: WSCG 2007. International Conference in Central Europe on Computer Graphics, Czech Republic (2007)
26. Wells, W.D.: Generating Enhanced Natural Environments and Terrain for Interactive Combat Simulation (GENETICS), PhD of the Naval Postgraduate School (2005)
27. Boulanger, K., Pattanaik, S., Bouatouch, K.: Rendering Grass in Real Time with Dynamic Light Source and Shadows. Irisa, internal publication n°1809 (2006)
28. Bittner, J., Wonka, P.: Visibility in Computer Graphics. Jour. of Env. and Planning (2003)

Iterative Methods for Visualization of Implicit Surfaces On GPU

Rodrigo de Toledo[1,2], Bruno Levy[2], and Jean-Claude Paul[3]

[1] Tecgraf – PUC-Rio, Brazil
[2] INRIA – ALICE, France
[3] Tsinghua University, China

Abstract. The ray-casting of implicit surfaces on GPU has been explored in the last few years. However, until recently, they were restricted to second degree (quadrics). We present an iterative solution to ray cast cubics and quartics on GPU. Our solution targets efficient implementation, obtaining interactive rendering for thousands of surfaces per frame. We have given special attention to torus rendering since it is a useful shape for multiple CAD models. We have tested four different iterative methods, including a novel one, comparing them with classical tessellation solution.

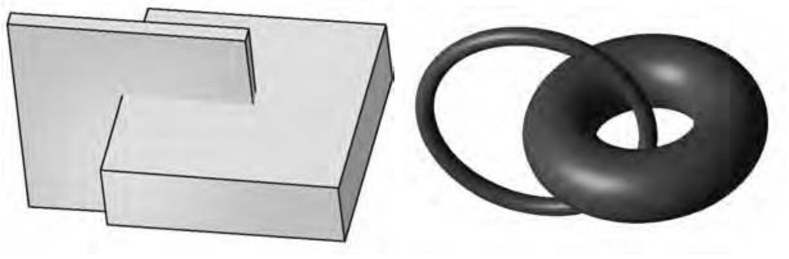


Fig. 1. The faces of two bounding boxes are used to trigger the fragment shader responsible for rendering the tori

1 Introduction

When programmable GPU were designed, the main goal was to produce images with better quality, while using standard triangle rasterization. However, this innovation gave large flexibility to vertex and pixel processing, motivating some completely new applications. One promising research area is the creation of new GPU primitives, extending the default ones (triangle, line and point). These new primitives do not aim to substitute the typical ones, but work together. The first one to appear in the literature was the quadrilateral primitive [1], which is based on mean value coordinates [2]. Implicit surfaces have also been directly implemented on GPU based on ray casting, but, until recently, they were restricted to second order ones. Spheres, ellipsoids and cylinders are some examples of quadrics ray-casted on GPU used in applications for molecule rendering [3,4] and tensor-field visualization [5]. The benefits of this new primitives compared to tessellation are: image quality (precise silhouette and per-pixel depth/shading), less memory usage and rendering efficiency.

Higher-order implicit primitives are a challenge for GPU implementation. Shader languages still have important restrictions when compared to CPU programming, including no support for stacks and recursive functions. We are interested in rendering cubics and quartics with a scalable implementation. The goal is to use them in practical applications that can benefit from this speed-up. Systems for massive model visualization are an example of target use. Among possible implicit surfaces of third and fourth degree, torus is the one with the most useful shape. It is largely found in CAD models. According to Nourse et al. [6] and Requicha et al. [7], 85% of industrial objects are described by plans and quadrics and this number grows to 95% if toroidal patches are also allowed. For this reason we have devoted special attention to torus as a new GPU primitive.

Loop and Blinn [8] were the first ones that investigated GPU implicit primitives with degree up to four. In their work, the intersection equation (between ray and surface) is solved using *analytic* techniques. In contrast, we adopted *iterative* methods, which resulted in faster rendering and more precise surfaces. We have used the Sturm method for both cubics and quartics. In the specific case of tori, we have tested four different approaches, including a novel one called *double derivative bisection*. In this work, we target per-pixel and per-vertex optimizations that result in high performance. We have measured the rendering speed for a single torus and also for multiple tori (up to 16,000 tori at the same time in the screen).

2 Related Work

GPU primitives are visualized through a ray-casting algorithm implemented inside the graphics card. The main computations are done in the pixel stage of the pipeline. To trigger the per-pixel algorithm, it is still necessary to raster a set of standard primitives. To keep GPU primitives compatible with rasterized surfaces, the visibility issue between objects is solved by the z-buffer, updated by both rasterization and ray-casting methods.

The concept of extended GPU primitives was first introduced by Toledo and Levy [9]. They have created a framework to render quadrics on GPU without tessellation. It is possible to visualize these surfaces with textures and self-shadows. The potential applications mentioned in their work were molecule rendering (using spheres) and tensor of curvature visualization (using ellipsoids). Later, Romeiro et al. [10] extended the original idea to reconstruct CSG models.

Bajaj et al. [3] have developed special GPU primitives for molecule visualization: spheres, cylinders and *helices*. The later are a derivation of cylinders used to represent secondary structures on molecules. They have reported interactive rendering for molecules with up to 100,000 atoms.

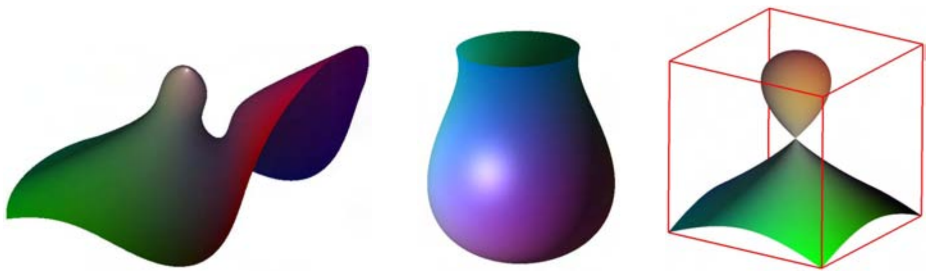


Fig. 2. Cubic surface examples. They are all clipped outside the domain $[-1,1]^3$ (represented by a cube in the last image).

Kondratieva et al. [5] and Sigg et al. [4] exclusively use points (`GL_POINTS`) as the rasterized primitive that triggers the fragment shader. Kondratieva et al. make use of ellipsoid-GPU primitives for diffusion tensor field visualization. Sigg et al. proposed a more general quadratic surfaces approach but their examples were restricted to molecule visualization with spheres and cylinders.

Loop and Blinn [8] broke through the second-order barrier and introduced the first GPU primitives up to fourth order. Based on Bézier tetrahedron, they succeeded in rendering cubics and quartics. However, the approach adopted by Loop and Blinn have some drawbacks in speed-up and quality. The performance limitation are a result of several conditions: (i) exhaustive per-fragment computation to solve all roots; (ii) excessive waste of fragments that are rasterized but not rendered (for example, the tetrahedron does not tidily fit the torus); (iii) complex per-vertex computation to transform Bézier vertices; and (iv) view-dependent CPU computation per tetrahedron. From the quality point of view, Loop and Blinn solution has a lack of precision near silhouette edges and self intersections. This is a result of numerical noise that can be aggravated depending on the choice of near and far clipping planes, because of computations that are not done on local coordinate space. A positive aspect of their implementation is the possibility of rendering piecewise algebraic surfaces, decomposed in several tetrahedrons.

3 Cubics

We extended the idea of GPU primitives for cubic implicit surfaces. It is possible to form interesting surfaces (see Figure 2) by using cubic equations, but these shapes are not as popular as the classical quadrics (e.g., sphere, cylinders, cones). Comparing to quadrics, the computation is much more intense.

The difficult in implementing the cubic primitive is the intersection searching process. It includes a root find problem of third degree. There are many polynomial root finders known in the literature [11] that could work in the cubic case. We choose to use a binary search based on the *Sturm theorem*. The binary search is a good solution in our case, since we have a restricted domain (we use a bounding box with local coordinates in $[-1,1]^3$).

Sturm Theorem. The theorem is based on a set of functions, known as *Sturm functions*, which are derivate from the base function $f(x)$:

$$f_0(x) = f(x), \quad f_1(x) = f'(x)$$
$$f_n(x) = -\left\{ f_{n-2}(x) - f_{n-1}(x) \left[\frac{f_{n-2}(x)}{f_{n-1}(x)} \right] \right\}, \quad n \geq 2$$

So, for a cubic function, there are four Sturm functions we need to generate (the last one, f_3, is a constant). With them, it is possible to find the number of real roots of an algebraic equation over an interval. After evaluating the set of functions for the two points defining the interval, the difference in the number of sign changes between them gives the number of roots in the interval.

Algorithm. Based on Sturm theorem, we can do a binary search to find the first positive intersection between the ray $R : (x, y, z) = o + t\boldsymbol{v}$ and the cubic surface. The initial interval is between $t_1 = 0$ and $t_2 = \lambda$, where the point $P_2 = o + \lambda\boldsymbol{v}$ is the point where the ray leaves the domain $[-1, 1]^3$. At each iteration, the interval is divided into two, $t_m = \frac{t_1+t_2}{2}$. Remark that, since we search only for the first intersection, we can use the number of sign changes ($n1$) of the origin point ($t = 0$) in all iterations. So, we just recompute the function values and the number of sign changes for the searching point (t_m).

Results. Sturm algorithm, although fast, is not as high performance as quadric ray casting. We achieved 300 fps in a GeForce 7900 graphics card. The quality of the results is very good (see Figure 2). However, in some special situations (close to singular points), the computation exceeds the precision of the GPU and some errors may appear, which are evident after zooming. A positive point of Sturm approach is the discard that is done before loop. If the fragment is not discarded at this moment, it means that there is at least one intersection. Another interesting consequence of using binary search is the possibility of adaptive quality. We can adjust the number of iterations according to the surface distance to the camera (Level Of Detail), improving speed when the surface is viewed from afar.

4 Quartics and Tori

We use Sturm technique, described on previous section, for generic quartics. It demands one more function evaluation than for cubics. In this section, we have given special attention to the GPU torus because, among all cubics and quartics, it is the most common primitive found in massive CAD environments (for example, a quarter of torus is typically used for pipe junctions).

For the torus GPU primitive we use a well fitted bounding box to trigger the fragments running our shader. A torus can be described by a quartic implicit function. Equation 1 defines a zero centered torus with main direction in z. This canonical position is the one used by our fragment shader to ray cast the torus, including one more definition: biggest and smallest radii sum is 1 ($R + r = 1$).

Given a ray $\boldsymbol{R} : (x, y, z) = o + t\boldsymbol{v}$, where $\boldsymbol{v} = [v_x, v_y, v_z]$ is a normalized vector, Equation 2 describes the ray-torus intersection[1].

$$(x^2 + y^2 + z^2 - (r^2 + R^2))^2 - 4R^2(r^2 - z^2) = 0 \tag{1}$$

$$T(t) : at^4 + bt^3 + ct^2 + dt + e = 0 \tag{2}$$

$$\begin{aligned}
\text{where}\quad & a = 1,\ b = 4(\boldsymbol{o} \cdot \boldsymbol{v}) \\
& c = 2((\boldsymbol{o} \cdot \boldsymbol{o}) - (R^2 + r^2) + 2(\boldsymbol{o} \cdot \boldsymbol{v})^2 + 2R^2(v_z)^2) \\
& d = 4((\boldsymbol{o} \cdot \boldsymbol{v})((\boldsymbol{o} \cdot \boldsymbol{o}) - (R^2 + r^2)) + 2R^2 v_z o_z) \\
& e = ((\boldsymbol{o} \cdot \boldsymbol{o}) - (R^2 + r^2))^2 - 4R^2(r^2 - o_z^2)
\end{aligned}$$

Finding the root of a quartic equation for several pixels in interactive rates is not a simple task. We have tried four different approaches (described in the following subsections) to choose an adequate algorithm for our GPU torus.

4.1 Sturm

We have extended the algorithm used in our cubic GPU primitive (see Section 3). Compared to ray-cubic intersection, ray-torus intersection has an extra computation since there is one more function to be evaluated in each iteration, totaling five functions. As a result, Sturm is not so fast for solving the torus-ray intersection (see Table 1). Another problem with the Sturm approach is the numerical precision. The complexity of terms on each Sturm function may overflow the floating-point capacity. This problem is viewing-angle dependent and in some cases may produce incorrect images (see Figure 4).

4.2 Double Derivative Bisection

This technique is also a binary search, as in Sturm technique. The idea is an extension of *Bisection method*. In this method, given two points t_0 and t_1, where $f(t_0)$ and $f(t_1)$ have different signs, we can ensure that there is at least one root (where $f(x)$ is a continuous function). Using the interval's midpoint $t_m = 0.5(t_0 + t_1)$ we evaluate the function, $f(t_m)$. Based on its sign we reduce the interval to be between t_0 and t_m or to be between t_m and t_1. This is a simple method that is always successful.

Derivative Bisection. We extend the bisection algorithm for working on two other special situations: both $f(t_0)$ and $f(t_1)$ have positive signs but with one and only one local minimum in the interval; and both $f(t_0)$ and $f(t_1)$ have negative signs but with one and only one local maximum in the interval. In these circumstances we can guarantee that, if there is an intersection (actually, up to two intersections), the *Derivative Bisection* algorithm can find it (them).

The algorithm does a binary search for the local minimum/maximum similarly to simple bisection but verifying the sign of derivatives. If the searching point

[1] We use a correction of Hanrahan equation [12] suggested by Eric Haines (see: `http://steve.hollasch.net/cgindex/render/raytorus.html`).

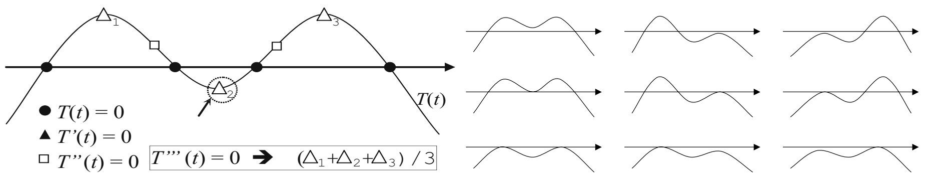


Fig. 3. *Left:* The round points mark the roots, the triangles mark the first derivative roots and the squares mark the second derivative root. The third derivative root is actually the average of the first derivative roots (see Equation 4). Its location must be somewhere around the second first-derivative root ($\triangle_2$). *Right:* All possible plots for the function $T(t)$ where there is at least one root (or one ray-torus intersection), except the four roots case.

(t_m) crosses the abscissa (in other words, $f(t_m)$ changes the sign), the algorithm switches for a simple bisection search. If local minimum/maximum is found and $f(t_m)$ did not change the sign, then there were no roots.

Double Derivative Bisection for Torus. The ray-torus intersection involves the solution of an equation of fourth degree (Equation 2), which yields a maximum of four possible intersections. One can easily imagine a ray traversing a torus and crossing its boundary four times. If we plot the evaluation of the torus intersection function $T(t)$ for this four-times crossing ray, we obtain something close to the form presented in Figure 3 (*left*). T has at most two local maximum and at most one local minimum (see all possible cases for T in Figure 3).

The basic idea in the *Double Derivative Bisection* is to initially divide the problem into two, running the single derivative bisection twice, first on the portion before the local minimum ($\triangle_2$ in Figure 3) and, if no intersection was found, on the portion right after. However, finding exactly the local minimum includes the solution of the derivative equation (Equation 3), which is a third-degree equation. Instead, we approximate the local minimum location by a much simpler computation. We compute the third-derivative root, which is the average of the three first-derivative roots (based on Vieta's Formulas):

$$\begin{gathered}
T'(t) \ : \ 4at^3 + 3bt^2 + 2ct + d = 0 \\
T'(t) = (t - \triangle_1)(t - \triangle_2)(t - \triangle_3) \quad \Rightarrow \quad \frac{3b}{4} = -(\triangle_1 + \triangle_2 + \triangle_3) \\
T'''(t) = 4t + b \ , \quad T'''(t_M) = 0 \quad \Rightarrow \quad t_M = -\frac{b}{4} = \frac{\triangle_1 + \triangle_2 + \triangle_3}{3}
\end{gathered} \tag{3}$$

The computation to find the root of the third derivative ($T'''(t_M) = 0$) is very simple since it uses only b, whose value is $4(\boldsymbol{o} \cdot \boldsymbol{v})$: $t_M = -(\boldsymbol{o} \cdot \boldsymbol{v})$. Although the third derivative is only an approximation (see Figure 3), it is good enough to divide the root finding algorithm into two for applying the derivative bisection (that is why we call our technique: *double derivative bisection*).

Double derivative bisection is slower than the Sturm technique (see Table 1). However, we have got results without the numerical problems found with Sturm, guaranteeing an error $\epsilon(r) \leq 0.0014$ relative to the minor radius r.

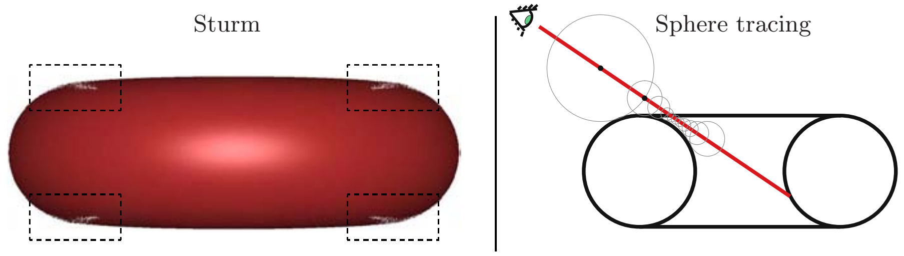


Fig. 4. *Sturm* Due to floating-point imprecision, Sturm method results in visual defects (which are more evident in the viewing-angle shown in this picture). *Sphere tracing* Example of critical situation for ray-torus intersection. The ray almost touches the torus reducing the step size. Therefore, more iterations are necessary.

4.3 Sphere Tracing

The sphere tracing was proposed by Hart in 1996 [13]. The idea is to find the ray-intersection by stepping closer and closer through the ray. Given the Euclidean distance function $d(x)$ to a surface, we can march along the ray from point x the distance $d(x)$ without penetrating the surface. For our canonical torus described in Equation 1, the distance function is $d(x) = ||(||(x, y)|| - R, z)|| - r$.

Compared to other methods, sphere tracing for ray-torus intersection needs less computation in each iteration. However, there are some critical situations whose approximation is very slow, increasing the iterations (see Figure 4).

To overcome these critical points, we have tested two sphere tracing for each ray with different starting points. The first one starts on the entering point in the torus bounding box, the second one starts on t_M (see Section 4.2). If after some iterations with the first sphere tracing $d_i(x)$ becomes greater than $d_{i-1}(x)$ then we proceed with the second sphere tracing.

With our two-rays implementation, we achieved better performance than the single-ray (see Table 1). However, the convergence of this algorithm is still slow. In the next subsection we present the implementation that resulted in the best performance among the four ones we have tried.

4.4 Newton-Raphson

The Newton-Raphson method (a.k.a. *Newton's method*) also uses the derivative evaluation in each iteration (as in Sturm and in double derivative bisection). The Newton's formula derives from the Taylor series, which is:

$$f(x + \delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2 + \cdots \qquad (4)$$

If δ is small enough, we can ignore the high-order terms so, for each iteration, we can move a δ step with:

$$\delta = -\frac{f(x)}{f'(x)}. \qquad (5)$$

Newton-Raphson algorithm converges quadratically. This means that near a root, the algorithm doubles the significant digits after each step [11]. However, far from the root, it may have a bad behavior. For example, if the current position is too close to a local extreme, the derivative is almost zero and δ vanishes to infinite. For this reason we took some extra cares in our GPU ray-torus intersection implementation.

Implementation. We use a well fit bounding box around the torus from where the ray-casting starts. Therefore, the starting point is not so far from the root, but we can still push it forward. Before applying the Newton iterations, we start by executing a simple ray-sphere intersection (the sphere radius is the sum of the torus radii). If there is no intersection, we can discard the fragment; if the intersection is negative (before the starting point), we ignore it; and if the intersection is positive, then we move the starting point to this position. However, it is still possible to have a local extreme between the starting and the intersection points. To avoid a vanishing situation, we use some bounds to guarantee that the step is not bigger than λ. Empirically, we found that $\lambda = 0.15$ (relative to our canonical torus) efficiently avoids the vanishing cases without loosing performance. The Newton-Raphson algorithm gave the best performance for our GPU-Torus. We use a threshold to stop the iterations that assures an error smaller than 0.1% of the minor radius r ($\epsilon(r) \leq 0.001$). We have also tested with $\epsilon(r) \leq 0.00003$. The results are shown in Table 1.

4.5 Normal Computation

One possible way to compute the normal vector for a point lying in the torus surface is by taking the three partial derivatives of the torus function at this point, which is quite expensive. Actually, we have implemented a geometric solution. Considering the canonical torus (Equation 1), the normalized normal $\boldsymbol{n}$ at the point P lying on torus surface is:

$$\boldsymbol{n}_{torus} = \frac{P - C}{r} \text{ , where } \quad C = \begin{cases} C_{xy} = \parallel P_{xy} \parallel \\ C_z = 0 \end{cases} \tag{6}$$

5 GPU Torus Results

5.1 Rendering One Torus

We have done several tests measuring the performance of each one of the four GPU Torus methods: Sturm, Bisection, SphereTracing and Newton. We have considered two different implementations for SphereTracing (one-ray and two-rays) and two implementations for Newton (varying the threshold). The results are presented in Table 1, which also contains the performance of traditional polygonal rasterization method. We have measured the frame rate from different angles for each different method, averaging them on the last column of Table 1. Among all GPU Primitive methods, *Newton 0.001* have presented the

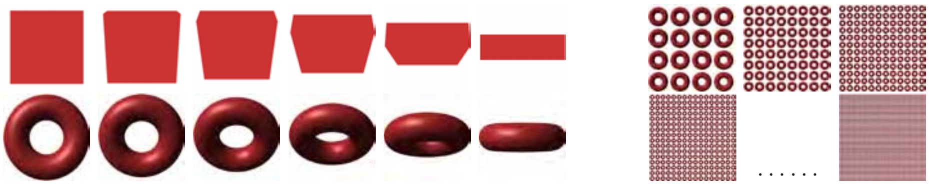


Fig. 5. *Left:* Several viewing-angles used for testing torus rendering performance (see Table 1). In the first row the bounding-box used for our GPU primitive, in the second row the torus itself. *Right:* Rendering multiple tori for the results in Figure 6.

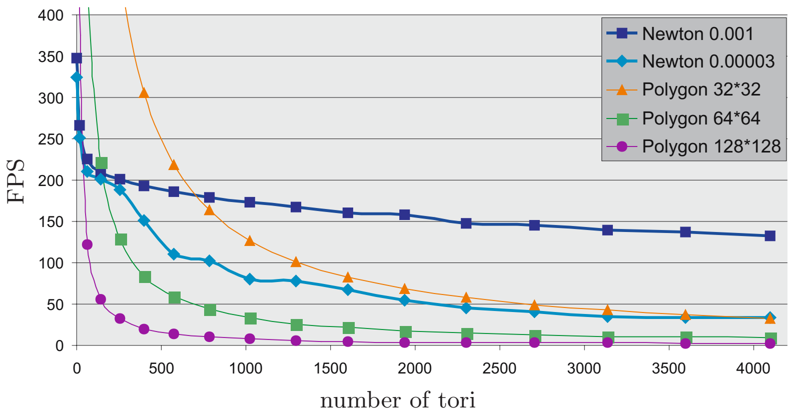


Fig. 6. Performance of different techniques for multiple tori rendered on the screen

best performance. As presented in next subsection, the performance of our GPU torus becomes interesting for multiple tori. However, we can see that even for individual torus, we can obtain competitive numbers. For an error $\epsilon(R+r) \approx 0.000350$, the polygonal version is slightly better. On the other hand, for an error $\epsilon(R+r) \approx 0.000015$, the GPU Torus is much faster.

5.2 Rendering Multiple Tori

One advantage of our GPU torus is that the bottleneck is no longer on the vertex stage, but on pixel stage. It means that the performance will not be reduced as much as using the polygonal torus version when increasing the number of tori. In Figure 6 we compare the GPU Newton technique in two different thresholds with three different resolutions of polygonal torus. We can verify that *Newton 0.001* is the fastest for more than 700 tori. Comparing *Newton 0.001* and *polygonal 128*128* (which have equivalent error) the Newton method is always faster. For more than 16000 tori, *Newton 0.001* is the only one that keeps an interactive frame rate (50 fps).

Table 1. Comparison between several torus rendering techniques. The tests were done in a 1024×1024 viewport with a torus filling the window, with a GeForce 7900 graphics card. **Projection Angle:** The number of pixels of the ray-casting area projection is determinant for the final frame rate and it varies according to the torus angle. For this reason we have done tests with 6 different viewing-angles (including top and perpendicular viewing), see Figure 5. **Fragment waste:** Number of fragments discarded divided by total fragments. **Polygon** $N * N$**:** Polygonal version of torus with N rings and N sides. ***Error*:** Our GPU torus techniques use an error threshold measured relative to the smaller radius: $\epsilon(r)$. We computed the error of the polygonal tori relative to their total radius: $\epsilon(R+r) = 1 - \cos\left(\frac{\pi}{N}\right)$. We can extract from one error the other one by using the radii proportion of our testing torus ($r = 0.3$ and $R = 0.7$). **Megapixels/second:** It is the corresponding multiplication of FPS and bounding-box pixels. This number indicates how many times the ray-casting algorithm was executed (in millions) per second. **StdDev:** Some of our ray-casting techniques suffer different per-pixel performance depending on the viewing-angle. To identify this fact we computed the standard deviation of each technique. Bisection and SphereTracing techniques had the most view-dependent performance.

	Projection Angle						Avg		
	0°	18°	36°	54°	72°	90°			
Bbox pixels	524 k	516 k	495 k	430 k	324 k	209 k	416 k		
Torus pixels	312 k	305 k	285 k	256 k	203 k	146 k	251 k		
Fragment waste	40.4%	40.9%	42.2%	40.5%	37.3%	29.8%	38.5%	*Error* (10^{-6})	
	FPS							$\epsilon(r)$	$\epsilon(R+r)$
Sturm	68	68	71	82	108	163	93	25	11
D.D. Bisection	15	15	16	19	25	46	23	1400	600
Sphere Tracing	16	16	16	20	27	46	24	1000	429
2-rays S. Tracing	66	66	65	80	100	153	88	1000	429
Newton 0.00003	246	254	261	302	380	537	330	30	13
Newton 0.001	267	283	291	324	410	590	**361**	1000	429
Polygon 32*32	2653	2649	2668	2704	2730	2744	2691	11236	4815
Polygon 64*64	1238	1238	1238	1238	1238	1238	1238	2811	1205
Polygon 128*128	369	369	369	369	369	369	369	703	301
Polygon 256*256	96	96	96	96	96	96	96	176	75
Polygon 512*512	24	24	24	24	24	24	24	44	19
	Megapixels/second							StdDev	$\frac{\text{StdDev}}{\text{Average}}$
Sturm	35.64	35.16	35.15	35.32	35.06	34.07	35.07	0.485	1.38%
Bisection	7.86	7.75	7.92	8.18	8.12	9.61	8.24	0.631	7.65%
Sphere Tracing	8.39	8.27	7.92	8.61	8.76	9.61	8.60	0.527	6.14%
2-rays S. Tracing	34.60	34.12	32.18	34.46	32.46	31.98	33.30	1.111	3.34%
Newton 0,00003	128.95	131.31	129.22	130.08	123.36	112.23	125.86	6.587	5.23%
Newton 0,001	139.95	146.31	144.07	139.56	133.10	123.31	137.72	7.654	5.56%

6 Conclusion and Future Work

Iterative methods are faster than analytical solutions for ray-casting, mainly because they compute only one root (for the first ray-surface intersection). Using a 640×480 viewport, the GPU torus with *Newton 0.001* method renders at 1300 fps in comparison to 500 fps obtained by analytical approach [8].

In future work, it is possible to extend the GPU iterative methods to higher order surfaces. However, some numerical precision problems may appear. A possible future application for our GPU tori could be their use for CAD models visualization. For instance, in a industrial environment, tori (and slices of torus) are easily found in tubular structures, chains and CAD patterns. In this kind of application, the tori must be used in combination with triangle meshes and with other extended GPU primitives, such as cylinders and cones. The frame rate obtained for multiple tori corroborates for the use of GPU primitives for massive CAD models visualization.

References

1. Hormann, K., Tarini, M.: A quadrilateral rendering primitive. In: HWWS 2004. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pp. 7–14. ACM Press, New York (2004)
2. Floater, M.S.: Mean value coordinates. Comput. Aided Geom. Des. 20, 19–27 (2003)
3. Bajaj, C., Djeu, P., Siddavanahalli, V., Thane, A.: Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. In: VIS 2004. Proceedings of the conference on Visualization 2004, pp. 243–250. IEEE Computer Society Press, Washington, DC, USA (2004)
4. Christian, S., Tim, W., Mario, B., Markus, G.: Gpu-based ray-casting of quadratic surfaces. In: Symposium on Point-Based Graphics, Boston,Massachusetts, USA, pp. 59–65. ACM Siggraph, New York (2006)
5. Kondratieva, P., Krüger, J., Westermann, R.: The application of gpu particle tracing to diffusion tensor field visualization. In: Proceedings IEEE Visualization 2005, IEEE Computer Society Press, Los Alamitos (2005)
6. Nourse, B., Hakala, D.G., Hillyard, R., Malraison, P.: Natural quadrics in mechanical design. Autofact West 1, 363–378 (1980)
7. Requicha, A.A.G., Voelcker, H.B.: Solid modeling: a historical summary and contemporary assessment. IEEE Computer Graphics and Applications 2, 9–22 (1982)
8. Loop, C., Blinn, J.: Real-time gpu rendering of piecewise algebraic surfaces. In: SIGGRAPH 2006. ACM SIGGRAPH 2006 Papers, pp. 664–670. ACM Press, New York (2006)
9. Toledo, R., Levy, B.: Extending the graphic pipeline with new gpu-accelerated primitives. In: 24th International gOcad Meeting, Nancy, France, Also presented in Visgraf Seminar 2004, IMPA, Rio de Janeiro, Brazil (2004)
10. Romeiro, F., de Figueiredo, L.H., Velho, L.: Hardware-assisted Rendering of CSG Models. In: Proceedings of SIBGRAPI 2006, Manaus, IEEE Computer Society Press, Los Alamitos (October 2006)

11. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1992)
12. Hanrahan, P.: A Survey of Ray - Surface Intersection Algorithms. In: An introduction to ray tracing, pp. 33–77. Academic Press Ltd., London (1989)
13. Hart, J.C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer 12, 527–545 (1996)

Fast Codebook Generation by Sequential Data Analysis for Object Classification

Alexandra Teynor and Hans Burkhardt

Albert-Ludwigs-Universität Freiburg, Department of Computer Science
Georges Köhler Allee 052, 79110 Freiburg, Germany
{teynor,Hans.Burkhardt}@informatik.uni-freiburg.de

Abstract. In this work, we present a novel, fast clustering scheme for codebook generation from local features for object class recognition. It relies on a sequential data analysis and creates compact clusters with low variance. We compare our algorithm to other commonly used algorithms with respect to cluster statistics and classification performance. It turns out that our algorithm is the fastest for codebook generation, without loss in classification performance, when using the right matching scheme. In this context, we propose a well suited matching scheme for assigning data entries to cluster centers based on the sigmoid function.

1 Introduction

A lot of visual recognition systems use local features to identify members of visual object classes. They are characterized by their wide applicability and their robustness against variations in object appearance, shape and partial occlusions.

The locations for feature extraction are determined by different techniques, ranging from regular grids over interest point detectors to random locations and scales. Most commonly, interest point detectors are used, since they consider specific types of structures (e.g. blobs, edges, corners), and can also have a certain degree of invariance built in, e.g. scale or affine invariance [13]. Once regions of interest are found in images, different types of features can be extracted to describe these areas.

Typically, features obtained in this way are not used directly for learning, but they are clustered and so called "visual codebooks" are created. A huge variety of different approaches use visual codebooks at some step in the recognition chain, e.g. [2,4,12,14], just to mention a few.

There are mainly two reasons for the use of codebooks. One of them is to be able to deal with the huge number of local features extracted from the training images, especially if we want to compare the distribution of local appearance vectors in some feature space. For a typical multi class object recognition problem, hundreds of thousands of local representations might be extracted, which is too much to be handled directly by most algorithms. The other reason is to better model the variability of the different parts, by averaging over different examples. A single representation, typically the cluster mean, serves as representative for the different cluster members and can be used for training as well as classification.

For codebook generation, we would like to focus on dense regions in feature space, i.e. regions where common structures occur. A typical feature vector used for local description consists of 128 dimensions, e.g. SIFT [10] or GLOH [11]. Precise density estimation in this 128-dimensional space is prohibitive, since on the one hand, we would need an enormous mass of data, and on the other hand, the representation of this density would be very difficult. If we would choose, e.g., an unparametric representation in form of a histogram with each dimension quantized into 4 bins, we would end up with a $4^{128} = 1.2 \cdot 10^{77}$ dimensional feature vector, which is almost as much as the estimated number of atoms in the universe ($\sim 10^{80}$).

However, not all locations in this feature space are equally probable, so we only want to use those that are relevant for our problem. The goal of codebook creation in our context can be seen as to define a reduced partition of this high dimensional space. In this way, we are able to define relevant "parts" or "structures".

In this work, we propose a novel approach on how to obtain visual codebooks by identifying sufficiently dense regions in feature space given a certain similarity threshold and for creating clusters with low variance. The method is much faster than other commonly used clustering algorithms as K-means or agglomerative clustering and can therefore be used to process more local features.

The outline of this paper is as follows: first, we give an overview about related work in section 2, then we describe our approach in section 3. In section 4, we show some experiments which are discussed subsequently. Finally, the conclusions are drawn in section 5.

2 Related Work

A variety of different clustering algorithms have been applied to visual codebook creation, e.g. hierarchical clustering (divisive clustering [9] as well as agglomerative clustering), clustering based on function optimization (e.g. expectation maximization (EM) type clustering [15]) or mixed techniques [8]. A general overview about clustering algorithms can be found in [15]. The two most commonly used techniques for codebook generation are agglomerative hierarchical clustering as well as K-means clustering. We review them briefly here. The task is always to cluster a set of local features $\mathbf{x}_n, n = 1, \ldots, N$.

1. **Agglomerative clustering**
   In the beginning, all data entries are regarded as single clusters. In each subsequent step, the most similar clusters are grouped, until only a single cluster remains. In this way, a tree structure of the data is created. To obtain individual clusters, the tree is "cut", either according to the desired number of clusters or a given minimum similarity between cluster members. In order to determine the similarity between clusters, different linkage strategies can be applied. Typically, the average link paradigm is used as it produces compact clusters, although it has a rather high time ($O(N^2 \log(N))$) and space

complexity ($O(N^2)$). This method is, for example, applied by Agarwal et al. [1] and Leibe et al. [7].

2. **K-means clustering**
   K-means clustering is an iterative procedure, where a function J describing the within-cluster variance gets minimized:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N_k} d(\mathbf{x}_{ij}, \boldsymbol{\mu}_j) \tag{1}$$

   We have K clusters, each consisting of N_k members, $\boldsymbol{\mu}_j$ is the cluster mean and $d(\cdot, \cdot)$ is a distance function. The time complexity of this algorithm is $O(NKq)$, where q is the number of iterations needed. The main advantage of K-means is its simplicity, however, it is sensitive to outliers. The number of clusters has to be fixed a priori, and the cluster means might lie far away from the cluster members. When random initialization is used, the clustering result might differ between runs. K-means clustering is used, for example, by Weber et al. [16] and Lazebnik et. al [6].

3 Proposed Approach

3.1 Sequential Clustering

Our goal is to find a partitioning of a high dimensional feature space for part based object class recognition. Typical clustering algorithms as described in the previous section do in fact more than that. They try to recover the structure of the data in feature space, e.g. by building a tree or minimizing an error criterion. For common objective functions, as in equation 1, this results in a higher number of cluster centers in more densely populated regions in feature space.

If we follow the principle of Occam's razor, we should select the simplest method that solves our problem. We only need to identify "sufficiently dense" regions in feature space and distribute cluster centers in these areas. We propose a simple sequential algorithm with low runtime complexity. The basic idea is to create hyperspheres with a certain radius. As all clustering algorithms, we assume that the distance in feature space does resemble the visual similarity of the patches. So the radius to be chosen depends on the distance in which samples are still considered visually similar. This has to be done experimentally.

The proposed algorithm is based on the Modified Basic Sequential Algorithmic Scheme (MBSAS) described in [15]. It is a two pass algorithm where first candidate cluster centers are determined. Then, the data is assigned to the respective closest cluster centers. After all data has been assigned, new cluster representatives are calculated from the cluster members in order to represent them better. Clusters with too few members get discarded. The algorithmic description can be found in algorithm 1. There, C denotes a set of features and $|C|$ the cardinality of the set C.

The difference to the original algorithm described in [15] is that the cluster centers are calculated after the assignment of all members, and only if the minimum member constraint has been fulfilled. This further speeds up computation.

Algorithm 1. Modified Sequential Clustering for Codebook Generation

```
Input: patch features $\mathbf{x}_n$, $n = 1, \dots, N$; hypersphere radius $\epsilon$; min density $\theta$ ;
Output: codebook entries $\mathbf{k}_l$, $l = 1, \dots, K$
begin
    $m \longleftarrow 1$;
    $C_m \longleftarrow \{\mathbf{x}_1\}$;
    for $i = 2$ to $N$ do
        Find $C_k : d(\mathbf{x}_i, C_k) = min_{1 \le j \le m} d(\mathbf{x}_i, C_j)$;
        if $d(\mathbf{x}_i, C_k) > \epsilon$ then
            $m \longleftarrow m + 1$;
            $C_m \longleftarrow \{\mathbf{x}_i\}$;
        end
    end
    for $i = 1$ to $N$ do
        Find $C_k : d(\mathbf{x}_i, C_k) = min_{1 \le j \le m} d(\mathbf{x}_i, C_j)$;
        $C_k \longleftarrow C_k \cup \{\mathbf{x}_i\}$;
    end
    $l \longleftarrow 1$;
    for $i = 1$ to $m$ do
        if $|C_i| \ge \theta$ then
            $\mathbf{k}_l \longleftarrow$ cluster representative for $C_i$;
            $l \longleftarrow l + 1$;
        end
    end
    $K \longleftarrow l - 1$;
end
```

The result of the clustering algorithm will depend on the order of the input. To not bias the result, the features should not be fed to the algorithm in the order they were extracted from the images, but shuffled beforehand. In order to determine the cluster representative, different methods are possible. In this work, the mean vector of the cluster members is taken, but also the median could have been used. The time complexity of MBSAS is $O(NK)$, which is smaller than the comlexity of agglomerative or K-means clustering. Please note that for calculating the time complexity, K is the initial number of candidate clusters generated in the first part of the algorithm, not the final number of valid clusters. How significant the speed up is can be seen from the actual clustering times for sample datasets in section 4.3.

3.2 Matching to Codebook Entries

In order to assign newly extracted features to codebook entries, different methods can be applied. We have tested two commonly used approaches, namely nearest neighbor and threshold based matching. We also applied a weighted matching scheme based on the sigmoid function, which we found very suitable.

– **N nearest neighbor matching:**
  The feature vector gets matched to the n nearest cluster centers, no matter what the distance is. We use $n = 3$ in our experiments.

- **Threshold based matching:** The features match to all cluster centers that are within a certain threshold. The threshold used is the hypersphere radius used for clustering. Thus, a vector might match to zero, one or more clusters.
- **Weighted matching using a sigmoid function:** A new feature vector $\mathbf{x}$ matches to a codebook entry k_l with the weight w_l determined by a sigmoid function:

  $$w_l = \frac{1}{1 + e^{\alpha(d(\mathbf{k}_l,\mathbf{x})-\epsilon})}$$

  The rationale behind this assignment function is that within a certain radius ϵ, the patches are all visually similar and should get the same high matching score. Patches with a distance above a threshold are too dissimilar and should get a low matching score. The region in between can be modelled by the factor α. It determines how steep the sigmoid function is.

For histogram creation, the matching values get normalized to sum up to one, i.e. each feature contributes the same to the distribution.

4 Experiments

In order to show the performance of our approach, we conduct different experiments. We use two different databases, each with a different classification task. One dataset is the Caltech3[1] dataset (airplanes, faces, motorbikes), the other is the Caltech101 dataset[2].

For each image in the respective database, we extract Harris-Laplace and Hesse-Laplace interest points [13]. The Harris-Laplace detector fires on corners, where the Hesse-Laplace detector finds blob like structures. The two types of interest points are treated separately in order to verify that the qualitative results do not depend on the type of interest point detector used. For each region, we calculate rotation sensitive GLOH [11] descriptors. For the computations, the binaries provided by Mikolajczyk are used[3].

The similarity threshold for the MBSAS clustering and the hard histogram matching was determined as follows: pairs of random sample patches from the database were shown to 7 different individuals who had to judge the visual similarity of the patches being "very well", "well", "not sure", "dissimilar" or "very dissimilar". The threshold was set between the average Euclidean distance in feature space for pairs that were judged to match "very well" and "well".

From each database, 30000 random patches were drawn from the training images and clustered with the MBSAS, K-means and agglomerative clustering scheme. This number was mainly limited by the time complexity of the agglomerative clustering algorithm. For the Caltech101 database, we used 15 randomly selected training and test images. We drew three independent sets and averaged

[1] from http://www.robots.ox.ac.uk/~vgg/data3.html
[2] from http://www.vision.caltech.edu/Image_Datasets/Caltech101
[3] from http://www.robots.ox.ac.uk/~vgg/research/affine/

the respective results. For the Caltech3 database, the same training and test images were used as in [5].

We discard single member clusters for all clustering results, since they are considered as too uncommon to generalize well. So $\theta = 2$ in our case. In order to have comparable codebook sizes, the cut value for the agglomerative clustering was chosen such that after the removal of the single member clusters the cluster number is the same as for the MBSAS clustering. The initial number of clusters for the K-means clustering was set so that the resulting number of non single member clusters was as close as possible to the value of the two other approaches. Since our K-means algorithm uses random initialization, it is hard to obtain exactly the same number.

4.1 Codebook Statistics

First, we want to look at certain statistics of the codebooks generated. We list the number of clusters obtained after the removal of the single member clusters as well as the single cluster ratio (scr), i.e. the percentage of the clusters that contained just a single member. For each cluster, we compute the cluster variance, i.e. the average squared distance of the cluster members to the cluster center. We list the average cluster variance per codebook and also the distribution of the cluster variances. The results can be seen from table 1 for the Caltech3 database, and from table 2 for the Caltech101 database.

The results are very consistent across the different databases and interest point detector types. The clustering with MBSAS results in visually very compact clusters, with a low average cluster variance. As can be seen from the variance distribution, there are no clusters with a very big variance, as e.g. for the K-means clusters. Cluster centers obtained as an average from widely spread data points are not guaranteed to represent the members adequately. This can also be confirmed by visual inspection of the clusters: patches belonging to some K-means clusters are visually quite distinct. As a consequence, the single cluster ratio is quite high for MBSAS codebooks as opposed to the other approaches, since only areas with a certain part density in a small neighborhood are kept.

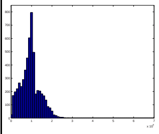
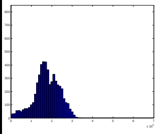
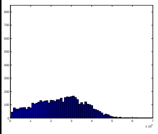
4.2 Classification Results

In order to compare the codebooks from a qualitative point of view, we performed two classification tasks. We first solve a two class problem on the Caltech3 database, where objects have to be distinguished from a background class. We then deal with a multi class problem on the Caltech101 database.

Since we only want to test the quality of the codebooks obtained, we use a simple "bag of feature" approach: we create histograms of object parts using the different codebooks and matching strategies. We neglect any spatial information. For classification, we use a standard SVM implementation (libSVMTL[4]) with a histogram intersection kernel. For the multi class problem, a one-vs-rest SVM was used.

[4] http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/

Table 1. Cluster statistics for codebooks for the Caltech3 database: scr = single cluster ratio; avg var = average variance; var dist = variance distribution

Hesse-Laplace	MBSAS	agg	K-means
# clusters	5489	5489	5005
scr	0.71	0.40	0.33
avg var	$0.94 \cdot 10^6$	$1.76 \cdot 10^6$	$2.42 \cdot 10^6$
var dist			

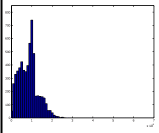
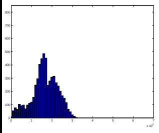
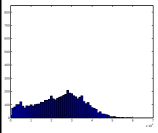
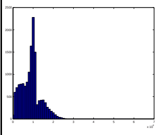
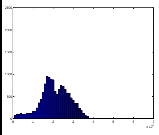
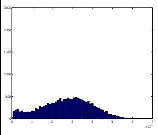
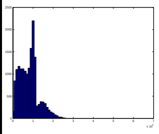
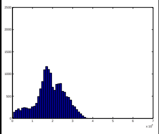
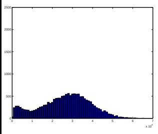
Harris-Laplace	MBSAS	agg	K-means
# clusters	5817	5817	5540
scr	0.69	0.39	0.26
avg var	$0.83 \cdot 10^6$	$1.65 \cdot 10^6$	$2.37 \cdot 10^6$
var dist			

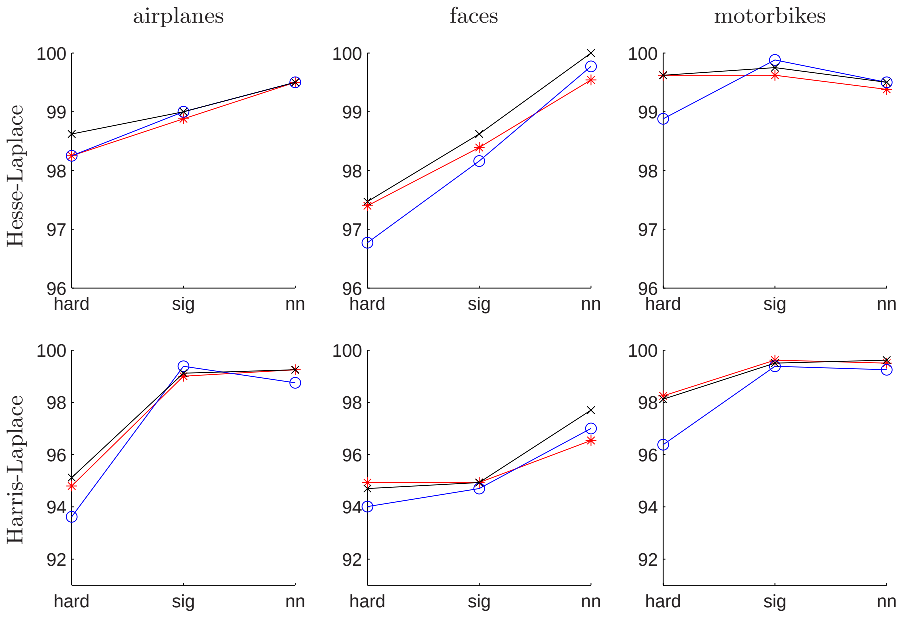
Table 2. Cluster statistics for codebooks for the Caltech101 database: scr = single cluster ratio; avg var = average variance; var dist = variance distribution

Hesse-Laplace	MBSAS	agg	K-means
# clusters	4917	4917	4967
scr	0.77	0.38	0.34
avg var	$0.89 \cdot 10^6$	$2.09 \cdot 10^6$	$2.71 \cdot 10^6$
var dist			

Harris-Laplace	MBSAS	agg	K-means
# clusters	5490	5490	5518
scr	0.75	0.38	0.26
avg var	$0.80 \cdot 10^6$	$1.88 \cdot 10^6$	$2.64 \cdot 10^6$
var dist			

The two class problem on Caltech3 is relatively easy as the images contain distinct structures for the individual object classes. Caltech101 is more diverse and contains a variety of different structures. The classification results for the different interest point detector types and matching strategies can be seen from table 3 for the Caltech3 database and from table 4 for the Caltech101 database.

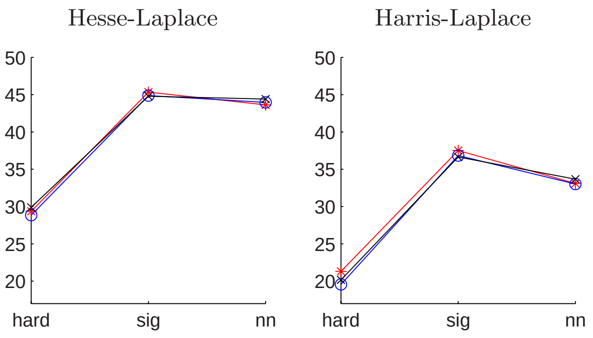
Table 3. Classification rate in % for the different Caltech3 problems. Results for MBSAS codebooks are shown in blue(o), for agglomerative codebooks in red (*) and for K-means codebooks in black(x). The results are given for the different matching strategies: hard = hard, sig = sigmoid and nn = 3 nearest neighbor.



The overall classification performance for Caltech3 is very well, in particular we could obtain a classification rate of 100% for the faces class with Hesse-Laplace interest points and K-means clustering. In general, Hesse-Laplace interest points performed better than the Harris-Laplace interest points. For Caltech101, more sophisticated classification strategies incorporating also spatial information brought better results (see e.g. [6]). However, in this work we only want to compare the relative performance of different clustering schemes.

From a classification point of view, there is no real winner in the clustering scheme. For the Caltech3 database and sigmoid matching, the MBSAS codebooks are slightly superior compared to the others regarding the categories airplanes and motorbikes, for the faces category, K-means clustering is superior when using nearest neighbor matching. Agglomerative clustering gives best results for the Caltech101 database and sigmoid matching.

Table 4. Classification rate in % for the Caltech101 problem. Results for MBSAS codebooks are shown in blue(o), for agglomerative codebooks in red (*) and for K-means codebooks in black(x). The results are given for the different matching strategies: hard = hard matching, sig = sigmoid matching and nn = 3 nearest neighbor matching.



Using hard matching with a tight threshold typically gives inferior results compared to using sigmoid or nearest neighbor matching. Here the MBSAS clusters are especially sensitive. When the structures are distinct as in the Caltech3 database, nearest neighbor matching is superior in almost all cases, since the nearest parts matched are likely to be from the same class. For more diverse databases as the Caltech101 sigmoid matching performed best.

4.3 Run Times

In this section, we list experimental run times for the different approaches. We performed the clustering for a different number of GLOH features computed around Hesse-Laplace interest points extracted from the Caltech101 dataset. The processing times were measured on 2.6 GHz AMD opteron processors. The results are listed in table 5. For the agglomerative and K-means clustering, we used the C clustering library by de Hoon et al. [3]. In this implementation, the K-means algorithm iterates until the assignment of features to clusters does not change any more. The MBSAS implementation was done by ourselves. The run times for the agglomerative clustering represent the time the entire tree needs for building, the run times for K-means and MBSAS clustering are given for settings that result in about the same number of clusters. For larger amounts of data, we increased the required number of member for clusters to be valid. We can observe that the processing time for K-means and MBSAS clustering grows over-proportional to the number of features. This is due to the fact that for the K-means algorithm, the number of iterations (q) until convergence is larger, and for the MBSAS algorithm, the number of candidate clusters in the first part of the algorithm is larger. The MBSAS algorithm runs very fast compared to the other algorithms. Thus it is possible to use more local representations in order to get a more complete view on the data. When increasing the hypershpere radius ϵ in which structures are considered similar, an even larger speed up is possible.

Table 5. Experimental run time results for different numbers of local features and clustering schemes

# local features	10^4	$3 \cdot 10^4$	10^5	$5 \cdot 10^5$
# of clusters	~ 750 ($\theta = 2$)	~ 3000 ($\theta = 2$)	~ 3700 ($\theta = 4$)	~ 5000 ($\theta = 10$)
Agg. clustering	26.6 min	11.6 h	n.a.	n.a.
K-means clustering	15.3 min	4.0 h	36.3 h	n.a.
MBSAS clustering	1.9 min	16.7 min	2.0 h	45.6 h

5 Conclusions

In this work we have presented a novel scheme to obtain codebooks for part based object classification. We compared our method to other commonly used algorithms for codebook creation. Our experiments have shown that despite the different properties of the resulting clusters, all three approaches performed similarly in a bag of feature classification approach. It seems to be sufficient to have cluster centers distributed in about the right area of feature space. Following the principle of Occam's razor, we have shown that no complicated algorithms with huge memory and runtime requirements are necessary, a simple sequential clustering scheme is sufficient. So more local structures can be used in codebook generation, to get a more complete view on the data distribution.

We have also shown that the matching scheme has more influence on recognition performance than the clustering algorithm: for diverse structures, sigmoid matching has shown to be superior, but also simple nearest neighbor matching is well suited, especially for simple problems.

All these observations were made for a bag-of-feature type classification approach. We plan to also test whether these observations hold for geometry based approaches, where distinct object parts have to be selected from codebooks.

Acknowledgments

This work was partially funded by the European Union MUSCLE NoE (FP6-507752).

References

1. Agarwal, S., Awan, A., Roth, D.: Learning to Detect Objects in Images Via a Sparse, Part-Based Representation. IEEE TPAMI 26(11), 1475–1490 (2004)
2. Agarwal, S., Roth, D.: Learning a Sparse Representation for Object Detection. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2353, pp. 113–127. Springer, Heidelberg (2002)
3. de Hoon, M.J.L., Imoto, S., Nolan, J., Miyano, S.: Open Source Clustering Software. Bioinformatics 20(9), 1453–1454 (2004)
4. Li Fei-Fei, R., Fergus, R., Perona, P.: One-shot Learning of Object Categories. IEEE TPAMI 28(4), 594–611 (2006)

5. Fergus, R., Perona, P., Zisserman, A.: Object Class Recognition by Unsupervised Scale-Invariant Learning. In: Proc. CVPR, vol. 2, pp. 227–264 (2003)
6. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: Proc. CVPR, vol. 2, pp. 2169–2178 (2006)
7. Leibe, B., Leonardis, A., Schiele, B.: Combined Object Categorization and Segmentation with an Implicit Shape Model. In: Proc. of the Workshop on Statistical Learning in Computer Vision, Prague, Czech Republic (May 2004)
8. Leibe, B., Mikolajczyk, K., Schiele, B.: Efficient Clustering and Matching for Object Class Recognition. In: Proc. BMVC (2006)
9. Linde, Y., Buzo, A., Gray, R.: An Algorithm for Vector Quantizer Design. Communications, IEEE Transactions on 28(1), 84–95 (1980)
10. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV 60, 91–110 (2004)
11. Mikolajczyk, K., Leibe, B., Schiele, B.: Local features for Object Class Recognition. In: Proc. ICCV, vol. 2, pp. 1792–1799 (2005)
12. Mikolajczyk, K., Leibe, B., Schiele, B.: Multiple Object Class Detection with a Generative Model. In: Proc. CVPR (2006)
13. Mikolajczyk, K., Schmid, C.: Scale and Affine Invariant Interest Point Detectors. IJCV 60(1), 63–86 (2004)
14. Opelt, A., Pinz, A., Zisserman, A.: A Boundary-Fragment-Model for Object Detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 575–588. Springer, Heidelberg (2006)
15. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 3rd edn. Academic Press, London (2006)
16. Weber, M., Welling, M., Perona, P.: Unsupervised Learning of Models for Recognition. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1842, pp. 18–32. Springer, Heidelberg (2000)

Iris Recognition: An Entropy-Based Coding Strategy Robust to Noisy Imaging Environments

Hugo Proença and Luís A. Alexandre

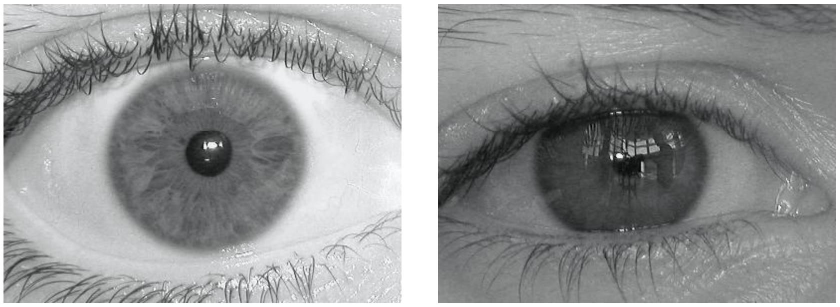
Universidade da Beira Interior, IT - Inst. Telecom.
UBI, R. Marquês D'Ávila e Bolama, 6200-001, Covilhã, Portugal
{hugomcp,lfbaa}@di.ubi.pt

Abstract. The iris is currently accepted as one of the most accurate traits for biometric purposes. However, for the sake of accuracy, iris recognition systems rely on good quality images and significantly deteriorate their results when images contain large noisy regions, either due to iris obstructions (eyelids or eyelashes) or reflections (specular or lighting). In this paper we propose an entropy-based iris coding strategy that constructs an unidimensional signal from overlapped angular patches of normalized iris images. Further, in the comparison between biometric signatures we exclusively take into account signatures' segments of varying dimension. The hope is to avoid the comparison between components corrupted by noise and achieve accurate recognition, even on highly noisy images. Our experiments were performed in three widely used iris image databases (third version of CASIA, ICE and UBIRIS) and led us to observe that our proposal significantly decreases the error rates in the recognition of noisy iris images.

1 Introduction

Continuous efforts have been made in searching for robust and effective iris coding methods, since Daugman's pioneering work on iris recognition was published. Iris recognition has been successfully applied in such distinct domains as airport check-in or refugee control. However, for the sake of accuracy, current systems require that subjects stand close (less than two meters) to the imaging camera and look for a period of about three seconds until the data is captured. This cooperative behavior is indispensable to capture images with enough quality to the recognition task. Simultaneously, it restricts the range of domains where iris recognition can be applied, namely within heterogeneous lighting conditions or under natural lighting environments. In this context, the overcome of these imaging constrains has motivated the efforts of several authors and deserves growing attention from the research community.

Although some of the published iris recognition algorithms perform a noise detection stage and produce a binary mask - used to avoid that noisy components of the biometric signatures are taken into account - we believe that highly heterogeneous lighting environments (specially under natural light) lead to the appearance of regions which, even for humans, are very difficult to classify as ”noisy” or ”noise-free”. Figure 1 illustrates some of the noise factors that result of less constrained image capturing environments. In figure 1b large iris regions obstructed by reflections (lighting and specular) can be observed, some of them very difficult to distinguish from the noise-free ones.



(a) Iris image with good quality. *(b) Noisy iris image.*

Fig. 1. Comparison between a good quality image and a noise-corrupted one. Figure 1a was captured under high constrained imaging conditions and is completely noise-free. Oppositely, figure 1b incorporates several types of noise, resultant from less constrained imaging conditions. It can be observed several iris obstructions - due to eyelids and eyelashes - and large regions of the iris corrupted by reflections, either lighting or specular.

In this paper our main goal is to propose an iris coding and comparison strategy with high robustness to noise. We start by measuring the entropy of consecutive and overlapped angular patches of normalized iris images. This gives an unidimensional signal that contains enough information to distinguish between individuals and, for this reason, is used as biometric signature. Further, in the comparison between signals we take into account shifted segments of varying dimension. The rationale is to profit the portions of the signals that were extracted from noise-free regions and hope that they contain enough information to reliably perform recognition.

Our experiments were performed in three iris image databases with different amounts of noise (third version of CASIA [1], ICE [2] and UBIRIS [3]). For comparison, we selected three of the most cited iris recognition algorithms (Daugman's [4], Wildes' [5] and Tan *et al.'s [6]*), that we believe to represent the majority of the published approaches. As described in the results' section, although the proposed method obtained roughly similar results to the other algorithms in the less noisy data sets (CASIA and ICE), it achieved smaller error rates in the recognition of the highly noisy iris images of the UBIRIS database. However, it should be stressed that some of this improvement was obtained at the expenses of a significant increase in the computational requirements of our proposal, since its computation time was about the double of the compared algorithms.

The remaining of this paper is organized as follows: section 2 briefly summarizes the most cited iris recognition methods. A detailed description of the proposed feature extraction and comparison method is given in section 3. Section 4 reports the experiments and discusses the results and, finally, section 5 presents the conclusions and points some directions for our further work.

2 Iris Recognition

Figure 2 illustrates the typical stages of iris recognition systems, which, in spite of the specificities of the different proposals, share the given structure. The initial stage deals with iris segmentation. This process consists in localize the iris inner (pupillary) and outer (scleric) borders, assuming either circular or elliptical shapes for both of the borders. In 1993, J. Daugman [4] proposed an integro-differential operator to find both the iris inner and outer borders. Similarly, [7] proposed integro-differential operators that search over the $\mathbb{N}^3$ space, with the goal of maximizing the equations that identify the iris borders. Wildes [5] achieved iris segmentation through a gradient based binary edge map construction followed by circular Hough transform. In [8], the authors proposed a method based in Wildes' method, that, together with a clustering process, achieves robustness for non-cooperative imaging environments.

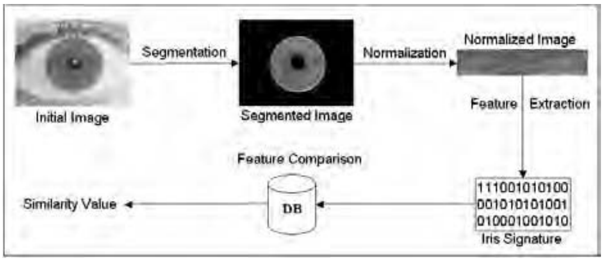


Fig. 2. Typical stages of iris recognition systems

In order to compensate variations in the pupils size and in the image capturing distances, it is usual to translate the segmented iris data into a fixed length and dimensionless polar coordinate system. This stage is usually accomplished through the method proposed by Daugman [9].

Regarding feature extraction, iris recognition approaches can be divided into three major categories: phase-based methods (e.g., [4]), zero-crossing methods (e.g., [10]) and texture-analysis based methods (e.g., [5]). Daugman [4] used multiscale quadrature wavelets to extract texture phase information and obtain an iris signature with 2048 binary components. Boles and Boashash [10] computed the zero-crossing representation of a 1D wavelet at different resolutions of concentric circles. Wildes [5] proposed the characterization of the iris texture through a Laplacian pyramid with 4 different levels (scales).

Lastly, the comparison between iris signatures is performed, producing a numeric dissimilarity value. If this value is higher than a threshold, the system outputs a *non-match*, meaning that each signature belongs to different irises. Otherwise, the system outputs a *match*, meaning that both signatures were extracted from the same iris. In this stage, it is common to apply different distance metrics (Hamming [4], Euclidean [11], Weighted Euclidean [12]) or methods based on signal correlation [5].

3 Proposed Recognition Method

As observed by Ma *et al.* [13], iris features tend to run in the radial direction, meaning that the most valuable information can be found by examining variations in the angular direction of the iris. Also, as reported by Tisse *et al.* [14], independently of the pupil dilation, the small-scale radial features of the iris remain stable.

Since the work of Shannon [15], the measuring of entropy has been widely used in the information theory domain and, more particularly, in image processing. Its use was recently reported for several purposes, namely to describe the visual information of images (e.g., [16] and [17]). It is defined as the average number of binary symbols necessary to code an input, given the probability of that input appearing an a stream. High entropy is associated with a high variance in the pixel values, while low entropy indicates that the pixel values are fairly uniform, and hence little detail can be derived from them. The information contained in an image can be regarded as mathematically identical to negative entropy. If any possible intensity value is likely to be next to any other there would be no information present. In this context, the probabilities of which pixel is going to be next to which other gives the information.

Let I be a grayscale image quantized to l intensity levels, the image entropy $h() : \mathbb{N}^{N \times N} \rightarrow \mathbb{R}^+$ is given by

$$h(I) = -\sum_{i=1}^{l} p_i(I)\ log_2(p_i(I)) \tag{1}$$

where p_i is the probability of the i^{th} quantized intensity level in the image I

$$p_i(I) = \frac{1}{r \times c} \sum_{r'=1}^{r} \sum_{c'=1}^{c} \mathbb{I}_{\{I(r',c') \,\in\, [inf_i, sup_i]\}} \tag{2}$$

where r and c are the number of columns and rows of the image, $I(r', c')$ is the intensity value of the pixel (r', c'), $[inf_i, sup_i]$ is the interval of intensities of the i^th quantized level and $\mathbb{I}_{\{.\}}$ is the characteristic function.

3.1 Feature Extraction

As illustrated by figure 3, our feature extraction strategy measures the entropy of overlapped angular windows of the normalized iris images. The goal is to construct an unidimensional signal s that contains information about the variations in the image entropy across the angular iris direction, which is believed to provide the most valuable discriminating biometric information.

Let N be a segmented and normalized iris image, with r_N rows and c_N columns $(r_N \times c_N)$. Also, let W_i be an $r_N \times (c_W + 1)$ image window, centered in the i^{th} column of N and composed by the columns $\{i - \frac{c_W}{2}, \ldots, i + \frac{c_W}{2}\}$ of N. The extracted signal $s = \{s_1, \ldots, s_{c_N}\}$ contains c_N components given by

$$s_i = h(W_i) \tag{3}$$

Fig. 3. Proposed feature extraction strategy. We construct an unidimensional signal through the measuring of the entropy of overlapped angular patches of normalized iris images. Each component of the signal s is given by the image entropy of the image window centered at the i column, spanned over all the rows of the normalized image and with fixed width.

Figure 4 contains examples of signals extracted from three normalized iris images. Figures 4a and 4b respectively illustrate signals extracted from a noise-free and a noisy image of the same iris. Here, the high similarity between the first 150 components of both signals can be observed. These components were extracted from the noise-free regions of the noisy iris image. Figure 4c represents a signal extracted from a different iris and the differences regarding any of the other signals are evident.

As described in the next sub-section, the main challenge is to reliably conclude about the subjects' identity independently of the signal variations that result from different imaging environments and noisy imaging conditions. This is the goal of our feature comparison method.

3.2 Feature Comparison

The underlaying idea of our proposed feature comparison method was published in [18] and consists in the direct correspondence between the number of compared components of the biometric signatures and the dissimilarity threshold that distinguish between matches and non-matches. The goal is to perform recognition of an individual using exclusively small portions of its biometric signature, those that are not corrupted by any type of noise.

In the following discussion, we will use a superscript to distinguish between signals extracted from different iris images, such as s^1 and s^2. Also, $s^1(a, b)$ denotes the shifted segment (a positions) of the signal s^1 with b components, such that $s^1(a, b) = \{s_a, \ldots, s_{(a+b)\ mod\ c_N}\}$.

The function $t(x) : \mathbb{N} \rightarrow \mathbb{R}$ gives the threshold that distinguishes between match and non-match comparisons for signatures with x components

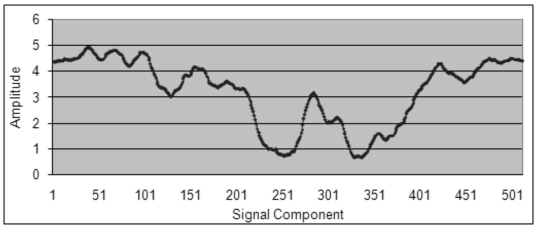
$$t(x) = \frac{(M_d - m_d)(x - m_l)}{M_l - m_l} + m_d \tag{4}$$

where m_l and M_l are respectively the minimum and maximum number of components of the comparable signals and m_d and M_d are the minimum and maximum threshold values, $m_l < M_l$ and $m_d < M_d$.

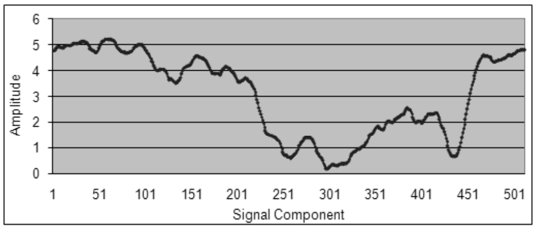
The function $fc(s^1, s^2)$ performs the comparison between segments of s^1 and s^2 and produces the decision about the identity of the subjects from where the signals were extracted:

$$fc(s^1, s^2) = \begin{cases} Match & , d(s^1(a_i, b_j), s^2(a_i, b_j)) \leq t(b_j) \\ Non - Match & , \text{otherwise} \end{cases} \tag{5}$$

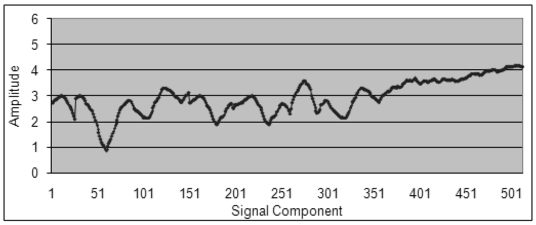
where $m_l \leq a_i, b_j \leq M_l$ and $d()$ is the Euclidean distance.



(a) Signal extracted from a noise-free image of subject 1.



(b) Signal extracted from a noisy image of subject 1.



(c) Signal extracted from subject 2.

Fig. 4. Examples of the unidimensional extracted signals. Figures 4a and 4b contain signals extracted from different images of the same iris, respectively from a noise-free (figure 4a) and a noisy (figure 4b) image. It is notorious the similarity between the first 150 components of the signals, which were extracted from the noise-free regions on both images. Also, the difference between both signals and the illustrated in figure 4c (extracted from an image of a different iris) is evident.

4 Experiments and Discussion

To enable the test of the proposed recognition method, we analyzed the available iris image databases and selected the most appropriate for our purposes. In the following we briefly describe the available public iris image databases and the data sets choused for our experiments.

4.1 Iris Databases

There are presently seven public and freely available iris image databases for biometric purposes: Chinese Academy of Sciences [1] (CASIA, three distinct versions), Multimedia University (MMU), University of Bath (BATH), University of Olomuc (UPOL), Iris Challenge Evaluation [2] (ICE), West Virginia University (WVU) and University of Beira Interior [3] (UBIRIS).

CASIA database is by far the most widely used for iris biometric purposes and has three distinct versions. However, its images incorporate few types of noise, almost exclusively related with eyelid and eyelash obstructions, similarly to the images of the MMU and BATH databases. UPOL images were captured with an optometric framework, obtaining optimal images with extremely similar characteristics. Although ICE and WVU databases contain images with more noise factors, their lack of images with significant reflections within the iris rings constitutes a weak point, regarding the simulation of less constrained imaging conditions. Oppositely, images of the UBIRIS database were captured under natural lighting and heterogenous imaging conditions, which explains their higher heterogeneity.

Due to this, we selected 800 images from 80 subjects of the UBIRIS database. In order to evaluate the recognition accuracy both in highly and less noisy environments, an equal number of images from the CASIA (third version) and ICE databases were selected. Further, we divided each data set into two halves. The first data sets - $UBIRIS_{tr}$, $CASIA_{tr}$ and ICE_{tr} - were used as training sets and the later - $UBIRIS_{tt}$, $CASIA_{tt}$ and ICE_{tt} - to evaluate the recognition accuracy.

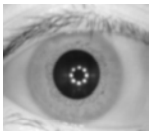
Each data set enables respectively 1800 and 78000 intra- and inter-class comparisons. Images of the UBIRIS data sets contain iris obstructions by eyelids and eyelashes, poor focused and motion blurred irises and irises with specular and lighting reflections, while those of the ICE data sets contain iris obstructions by eyelids and eyelashes, off-angle iris images and eyes with contact lenses. Images of the CASIA data sets have minor portions of noise, almost exclusively related with iris obstructions by eyelids and eyelashes.

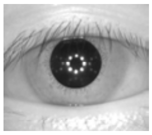
4.2 Recognition Methods Used as Comparison

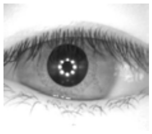
As above stated, we compared the results obtained by the proposed recognition method and three of the most cited iris recognition algorithms (Daugman [9], Wildes [5] and Tan *et al* [6]), which we believe to represent the majority of the published proposals. Also, the algorithm proposed by Daugman is the basis of all the commercially deployed recognition systems.

4.3 Results and Discussion

For all images of the above described data sets we extracted the biometric signatures, according to each above mentioned approach. Further, we performed the feature comparison with all the remaining signatures of the same data set. In order to avoid that inaccuracies in the iris segmentation and normalization stages corrupt the obtained results, we manually verified the accuracy of the iris segmentation algorithms.


(a) Poor focused iris image.

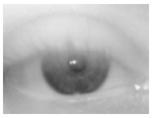

(b) Iris with eyelid obstructions.

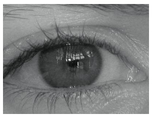

(c) Iris with eyelid and eyelash obstructions.

(d) Off-angle iris image.

(e) Eye with contact lenses.

(f) Iris with extreme eyelid and eyelash obstructions.


(g) Motion blurred iris image.


(h) Iris with large reflection areas.

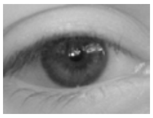
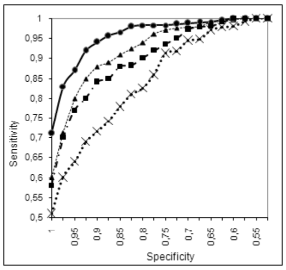
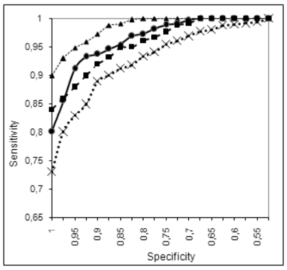

(i) Iris with large reflection areas and eyelids and eyelash iris obstructions.

Fig. 5. Examples of the images used in our experiments. The first row contains images of the third version of the $CASIA$ database, which commonly have iris obstructions due to eyelids or eyelashes and poor focused images. The second row contains images of the ICE database. It incorporates off-angle images, iris obstructions and several eyes with contact lenses. Finally, the third row exemplifies some of the highly noisy images contained by the $UBIRIS$ database. This database contains, apart from the above referred types of noise, images with large reflection regions, either specular or lighting.

Figure 6 compares the obtained receiver operating curves (ROCs) by our proposal (continuous lines with circular data points), Daugman's (dashed lines with triangular data points), Wildes' (dotted lines with cross data points) and Tan *et al.*'s (large dashed lines with square data points) in the $UBIRIS_{tt}$ (figure 6a), ICE_{tt} (figure 6b) and $CASIA_{tt}$ (figure 6c) data sets. It can be observed that the error rates obtained in the less noisy data sets were similar to those obtained by the Wildes and Tan *et al.* algorithms. In both these data sets, Daugman's recognition algorithm achieved better results. However, in the highly noisy images of the $UBIRIS_{tt}$ data set, which incorporates images with very large reflection regions, our proposal outperformed the results obtained by any of the other recognition algorithms, which confirms its higher robustness to noise.

Table 1 summarizes the error rates obtained by the above described algorithms in the $UBIRIS_{tt}$, ICE_{tt} and $CASIA_{tt}$ data sets. The first column identifies the classification method, the second contains the false rejection rates when preventing the false


(a) ROCs obtained in the $UBIRIS_{tt}$ data set


(b) ROCs obtained in the ICE_{tt} data set

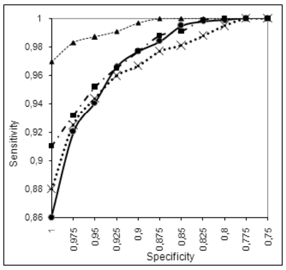

(c) ROCs obtained in the $CASIA_{tt}$ data set

Fig. 6. Comparison between the ROCs obtained in the $UBIRIS_{tt}$ (figure 6a) , ICE_{tt} (figure 6b) and $CASIA_{tt}$ (figure 6c) data sets. The continuous lines with circular data points represent our proposal. Daugman's is represented by the dashed lines with triangular data points and Wildes' by the dotted lines with cross data points. Finally, the large dashed lines with square data points represents the recognition method proposed by Tan *et al.*. Although our proposal obtained higher error rates than the other methods used as comparison in the recognition of images of the $CASIA$ and ICE data sets, it outperforms these methods in the recognition of highly noisy images (figure 6a).

Table 1. Comparison between the error rates obtained by the tested algorithms in the images from the $UBIRIS_{tt}$, ICE_{tt} and $CASIA_{tt}$ data sets

Recognition method	FRR, FAR=0 (%)	EER (%)	FR
$UBIRIS_{tt}$ data set			
Proposed	29.17 ± 0.04	08.91 ± 0.02	86.17
Daugman	40.02 ± 0.06	11.47 ± 0.03	75.83
Wildes	49.73 ± 0.07	18.73 ± 0.03	54.19
Tan *et al.*	42.77 ± 0.06	13.70 ± 0.03	70.41
ICE_{tt} data set			
Proposed	20.05 ± 0.03	06.70 ± 0.02	109.14
Daugman	11.30 ± 0.03	04.82 ± 0.02	143.19
Wildes	27.11 ± 0.04	10.58 ± 0.03	105.12
Tan *et al.*	16.06 ± 0.03	08.86 ± 0.02	127.95
$CASIA_{tt}$ data set			
Proposed	14.03 ± 0.03	05.03 ± 0.02	119.72
Daugman	03.97 ± 0.01	01.90 ± 0.01	147.16
Wildes	12.09 ± 0.03	05.25 ± 0.02	105.91
Tan *et al.*	09.27 ± 0.02	04.94 ± 0.02	127.93

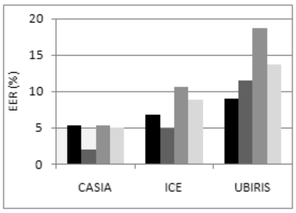
accept errors $(FRR, FAR = 0)$. EER corresponds to the approximated equal error rate and, finally, the last column contains the value of a Fisher-ratio test (FR) given by:

$$FR = \frac{(\mu^E - \mu^I)^2}{\frac{\sigma^{I^2}}{N^I} + \frac{\sigma^{E2}}{N^E}} \tag{6}$$

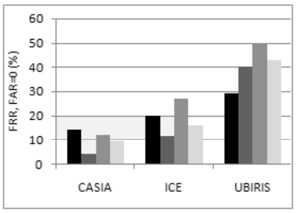
where μ^I and μ^E respectively indicate the mean of the intra- and inter-class dissimilarities. σ^I and σ^E indicate the respective standard deviations and N^I and N^E are, respectively, the number of intra- and inter-class comparisons. All the error rates are expressed for a confidence interval of 95%.

Once again, it can be observed that Daugman's recognition method achieved higher separability between the intra- and inter-class comparisons in the less noisy data sets ($CASIA_{tt}$ and ICE_{tt}). In both cases, the values of the FR test are higher and the error rates (either the EER or the $FRR,\ FAR = 0$) smaller. However, when images incorporate large reflection regions, all the algorithms used in the comparison with our proposal significantly decreased their accuracy, showing their small robustness to noise. In the $UBIRIS_{tt}$ data set the proposed recognition method achieved significantly lower error rates.

The high robustness of our proposal can be highlighted by the proportion values between the results obtained in the noisiest and less noisy iris image data sets: ($\frac{UBIRIS_{tt}}{CASIA_{tt}}$) and ($\frac{UBIRIS_{tt}}{ICE_{tt}}$). For the EER we obtained the proportion values of respectively 1.77 and 1.32, which are significantly lower than those obtained for the Daugman's (6.03 and 2.37), Wildes' (3.56 and 1.77) and Tan *et al.*'s (2.77 and 1.54) recognition algorithms. Similar values were obtained for the other error (FRR, FAR=0) and separability (FR) measures.



(a) Obtained EER in the three data sets.



(b) Obtained FRR, FAR=0 in the three data sets.

Fig. 7. Degradation in the accuracy of the tested algorithms, regarding the amount of noise of the data sets. The $UBIRIS$ is the noisiest and $CASIA$ the less noisy. Our proposal, Daugman's, Wildes' and Tan *et al.*'s are respectively represented by the darkest to the brightest bar series. Although our proposal obtained higher error rates in the less noisy data sets (CASIA and ICE), a smaller degradation in the results is evident.

Figure 7 illustrates the increase of the error rates (EER in figure 7a and FRR, FAR=0 in figure 7b) obtained by the experimented algorithms, regarding the amount of noise in the iris images. The vertical axes contains the error rates (%) and the horizontal contain the data sets where these errors were obtained, ordered from the less noisy to the noisiest data sets. Our proposal, Daugman's, Wildes' and Tan *et al.*'s are respectively represented by the darkest to the brightest bar series.

Computation Time. The algorithms were implemented in C++, following an object-oriented paradigm and running in an image-processing framework developed by the authors. This framework is not optimized for execution speed, as the algorithm's implementation was made without these concerns, but instead with a user-friendly objective. However, the proportion between the average execution time of the implemented algorithms can be evaluated and assumptions about their optimized execution time can be made. The fastest recognition algorithm was the one of Wildes, with average execution time of about 1.6 seconds. The computation time observed for the Daugman's and Ma *et al.* algorithms was about 17 and 21% higher. Finally, our recognition proposal has higher computational requirements, essentially due to the feature comparison stage. The observed average execution time was more than 90% higher than the one of Wildes.

We are focused on the development and evaluation of alternate feature comparison strategies that present similar robustness to noisy signals and have smaller computational requirements.

5 Conclusions and Further Work

Having observed the significant impact of noise in the accuracy of the most relevant iris recognition algorithms, we described a new iris recognition strategy more robust to noise. The proposed method encodes the iris information into an unidimensional signal, that measures the entropy of overlapped iris angular patches. For the purpose of signature matching, we compared segments of the signals with varying dimension, hoping that noise-free components allow reliable biometric recognition. Here, the dissimilarity threshold that distinguishes between match and non-match comparisons has direct correspondence with the length of the compared signal segments.

Although our proposal did not outperform the compared ones in the less noisy data sets (CASIA and ICE), we observed a significantly minor degradation in the recognition of the highly noisy images of the $UBIRIS$ database. This makes our proposal more suitable for the application within less constrained imaging environments.

However, the computational requirements of the proposed method are a concern. Our efforts are presently concentrated in decrease the computational complexity, as well in the search of alternate feature comparison methods more accurate in the recognition of noise-free images.

Acknowledgements

We acknowledge the support of the Portuguese Fundação para a Ciência e Tecnologia (FCT) and FEDER, project PTDC/EIA/69106/2006.

References

1. Institute of Automation, Chinese Academy of Sciences: CASIA iris image database (2004), `http://www.sinobiometrics.com`
2. National Institute of Standards and Technology: Iris challenge evaluation (2006), `http://iris.nist.gov/ICE/`

3. Proença, H., Alexandre, L.A.: UBIRIS: A noisy iris image database. In: ICIAP 2005. Proceedings of the 13^{th} International Conference on Image Analysis and Processing, pp. 970–977 (2005), `http://iris.di.ubi.pt`
4. Daugman, J.G.: High confidence visual recognition of persons by a test of statistical independence. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(11), 1148–1161 (1993)
5. Wildes, R.P.: Iris recognition: an emerging biometric technology. In: Proceedings of the IEEE, vol. 85(9), pp. 1348–1363. IEEE Computer Society Press, Los Alamitos (1997)
6. Ma, L., Tan, T., Zhang, D., Wang, Y.: Local intensity variation analysis for iris recognition. Pattern recognition 37(6), 1287–1298 (2004)
7. Camus, T., Wildes, R.: Reliable and fast eye finding in close-up images. In: Proceedings of the IEEE 16th International Conference on Pattern Recognition, Quebec, pp. 389–394. IEEE Computer Society Press, Los Alamitos (2002)
8. Proença, H., Alexandre, L.A.: Iris segmentation methodology for non-cooperative iris recognition. In: IEE Proc. Vision, Image & Signal Processing, vol. 153(2), pp. 199–205 (2006)
9. Daugman, J.G.: How iris recognition works. IEEE Transactions on Circuits and Systems for Video Technology 14(1), 21–30 (2004)
10. Boles, W.W., Boashash, B.: A human identification technique using images of the iris and wavelet transform. IEEE Transactions on Signal Processing 46(4), 1185–1188 (1998)
11. Huang, Y., Luo, S., Chen, E.: An efficient iris recognition system. In: Proceedings of the First International Conference on Machine Learning and Cybernetics, China, pp. 450–454 (2002)
12. Ma, L., Wang, Y., Tan, T.: Iris recognition using circular symmetric filters. In: ICPR 2002. Proceedings of the 25^{th} International Conference on Pattern Recognition, pp. 414–417 (2002)
13. Ma, L., Tan, T., Wang, Y., Zhang, D.: Personal identification based on iris texture analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(12), 2519–2533 (2003)
14. Tisse, C., Martin, L., Torres, L., Robert, M.: Person identification technique using human iris recognition. In: Proceedings of the 25th International Conference on Vision Interface, Calgary, pp. 294–299 (2002)
15. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 623–656 (1948)
16. Ferraro, M., Boccignone, G., Caelli, T.: Entropy-based representation of image information. Pattern Recognition Letters 23, 1391–1398 (2002)
17. Yanai, K., Barnard, K.: Image region entropy: a measure of ”visualness” of web images associated with one concept. In: Proceedings of the 13^{th} annual ACM International Conference on Multimedia, Singapore, pp. 419–422. ACM Press, New York (2005)
18. Proença, H., Alexandre, L.A.: Toward non-cooperative iris recognition: A classification approach using multiple signatures. IEEE Transactions on Pattern Analysis and Machine Inteligence 9(4), 607–612 (2007)

Radial Edge Configuration for Semi-local Image Structure Description

Lech Szumilas, Horst Wildenauer, Allan Hanbury, and René Donner

Pattern Recognition and Image Processing Group
Vienna University of Technology
A-1040, Vienna, Austria
lech@prip.tuwien.ac.at
http://www.prip.tuwien.ac.at/∼lech

Abstract. We present a novel semi-local image descriptor which encodes multiple edges corresponding to the image structure boundaries around an interest point. The proposed method addresses the problem of poor edge detection through a robust, scale and orientation invariant, descriptor distance. In addition, a clustering of descriptors capable of extracting distinctive shapes from a set of descriptors is described. The proposed techniques are applied to the description of bone shapes in medical X-ray images and the experimental results are presented.

1 Introduction

Edges are an intuitive way to represent shape information, but the problems associated with poor edge detection often affect the final result based on edge matching or classification. To overcome this problem we introduce a novel semi-local shape descriptor which represents the shape of an image structure by means of edges and their configurations. Our *Radial Edge Configuration*-descriptor (REC) encodes edges found in a neighborhood of an interest point as a sequence of radial distances in a polar coordinate system (centered on the interest point). Thus, the similarity of shape is assessed by the comparison of local edge configurations. Building on RECs we investigate the possibility to extract groups of similar edge structures by unsupervised clustering of edges. Here, our main contributions are: The definition of a rotation and scale-invariant distance measure between edge configuration descriptors. The most important property of the distance measure is it's ability to match multiple edges, preserving their spatial relationships and rejecting outlier edge pairs at the same time. This allows for a comparison of image structures across different scales, with only partially established correspondences. Another particularity of the chosen approach is that scale and orientation are not estimated during descriptor extraction. Instead they are established as relative entities between two REC descriptors during the distance calculation, which leads to more stable results. The second contribution is the introduction of hierarchical edge clustering using this distance. Clustering similar edges in an image is particularly difficult when edges have different lengths or are fragmented due to poor detection. However, the proposed distance measure was

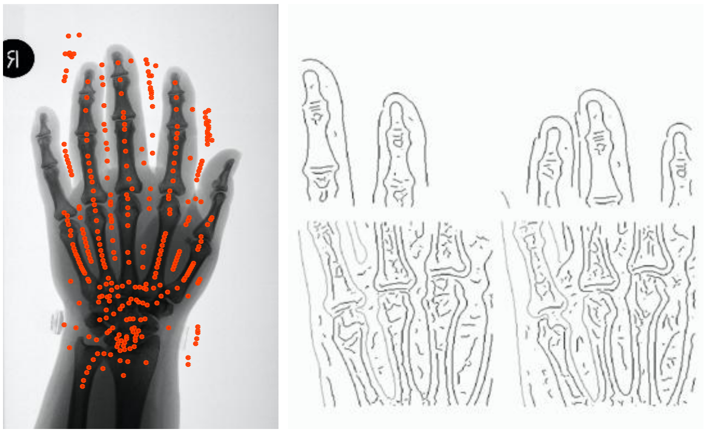


Fig. 1. Left: examples of interest point distribution. Right: examples of edge detection.

devised to properly handle pairings of edges of arbitrary length, thus allowing for the clustering of edge configurations in these otherwise complex cases.

The ability to describe local edge configurations around interest points and to cluster them opens up many application areas. We consider automatic model extraction for medical images, where we attempt to capture similar image structures in radiographs of the hand in an unsupervised way. This is similar to approaches which have had much success in object recognition, for example the clustering of SIFT features [6].

Organization: Matching single and multiple edges to define a distance measure between REC descriptors is discussed in Sections 2 and 3. Clustering these descriptors is described in Section 4. Section 5 presents an experimental evaluation of the discriminability of the REC descriptor and in Section 6 we draw a conclusions.

1.1 Related Work

Our approach is related to the work of Carmichael and Hebert [1], where *local edge configurations* are used for recognition of wiry objects in cluttered scenes. The most important difference is that their edge description is based on edge probes which are analogous to a Gaussian receptive field, while we use explicit shape description in the form of possibly open curves. Furthermore the learning procedure in our case is concentrated on automatic detection of the model which leads to a direct representation of stable shapes. Other recent object recognition methods, utilizing edge information, exist [3, 4], but their methodology differs significantly from the one presented in this paper. Comprehensive overviews of the field can be found in [1, 3].

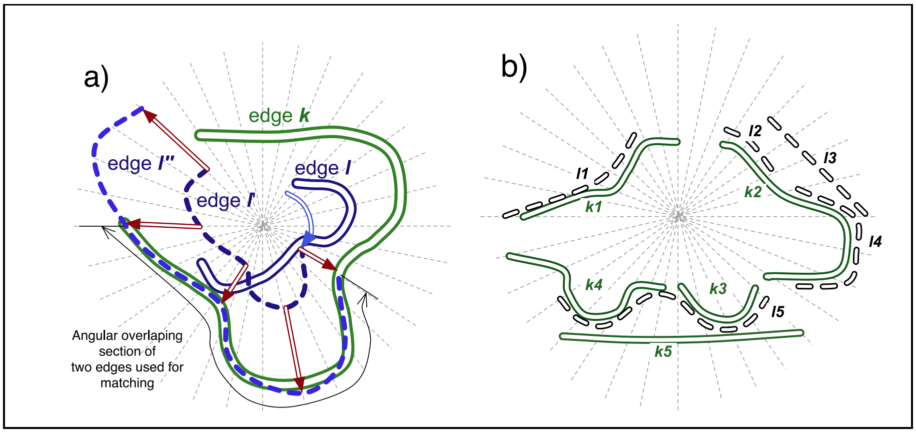


Fig. 2. a) - example of matching edge k and l in polar coordinates. Edge l' is a rotated version of l and l'' is scaled version of l' relatively to the origin of the coordinate system. b) - example of edge correspondences in two descriptors (edges k and l).

2 Edge Matching in Polar Coordinates

The complexity of edge matching is primarily associated with the difficulty in assigning a scale to the edge – even a tiny part of one edge may be matched to another edge or to itself at larger scale (e.g. fractal like structures). Polar coordinates allow the definition of an edge scale locally, based on the relative position to the origin of a coordinate system. However, the matching of a part of an edge to a part or whole of another edge is still admissible.

The origin of coordinate system is associated with the interest point location. We use the symmetry based interest point detector introduced in [5] and the Canny edge detector for obtaining edges around interest points. Examples of interest point distribution and edges detected in the hand X-Ray are shown in the Figure 1.

The REC descriptor consists of a variable number of K continuous edges. The k-th edge $\mathit{\Gamma}_k$ is encoded as an ordered list of radial boundary points, each representing the distance $r_{k,i}$ along the i-th ray from the origin of the polar coordinate system:

$$\mathit{\Gamma}_k = \{r_{k,i} : i \subset \mathbb{N}_0^+; i = (b_k ... b_k + n_k) \bmod N\} \tag{1}$$

where b_k denotes the index of the first ray and n_k is the number of rays the edge occupies. The modulo operation is used to ensure that index $i < N$, where N describes the total number of rays (polar resolution) and in all our experiments is set to 64, which we found to offer a good compromise between accuracy and computational cost..

Calculating the distance between two REC descriptors usually involves finding correspondences between multiple edges. We describe a method to find the best fit between two edges, assuming one of the edges can be rotated and scaled relative to the origin of polar coordinate system, associated with the interest point (as shown in Figure 2). This operation is a prerequisite for the estimation of distance between two REC descriptors.

Fitting one edge to another corresponds to finding a transformation (rotation and scaling) which globally minimizes the spatial distance between corresponding boundary points of the two edges. It is important to note that while the scaling of an edge is performed in the continuous domain, the relative rotation is quantized into N rays. The relative scale $\varsigma_{k,l}^{a,b}$ between edge k belonging to the descriptor a and edge l belonging to the descriptor b, rotated by α rays, is calculated as follows:

$$\varsigma_{k,l}^{a,b}(\alpha) = \left(\sum_{i=b_{kl}}^{b_{kl}+n_{kl}} r_{k,i}^{a} r_{l,\bar{i}}^{b} \right) / \left(\sum_{i=b_{kl}}^{b_{kl}+n_{kl}} (r_{l,\bar{i}}^{b})^2 \right) \tag{2}$$

where b_{kl} is the first ray containing boundary points of both edges, n_{kl} is the number of consecutive rays containing boundary points from both edges for a given rotation α and $\bar{i} = (i - \alpha) \bmod N$. It is important to note that this scheme allows for partial edge matching, which means that only the overlapping section of the two edges is matched (as shown in Figure 2). However, only combinations of α for which $n_{kl} \geqslant \tau$ (in our experiments τ=5) are used, due to the fact that extremely short sections of an edge usually carry less information, which is made worse by the quantization process. It can be easily proven that the spatial distance between corresponding boundary points of the edges k and l, for a given rotation α, is minimized when edge l is scaled (multiplied) by $\varsigma_{k,l}^{a,b}(\alpha)$.

One way of estimating how well two edges fit together is to calculate the variation of relative scale between the corresponding boundary points:

$$\epsilon_{k,l}^{a,b}(\alpha) = \frac{1}{n_{kl}} \sum_{i=b_{kl}}^{b_{kl}+n_{kl}} \left| \log^2\left(\frac{r_{k,i}^{a}}{r_{l,\bar{i}}^{b}} \right) - \log^2\left(\varsigma_{k,l}^{a,b}(\alpha) \right) \right| \tag{3}$$

This equation is a scale independent fitting distance between two edges for a given relative rotation α. The $\log^2()$ operation is used to avoid impairment associated with the $\frac{r_{k,i}^{a}}{r_{l,\bar{i}}^{b}}$ measure. The relative rotation giving the best fit of the two edges is the one which minimizes the distance $\epsilon_{k,l}^{a,b}$:

$$\epsilon_{k,l}^{a,b} = \min_{\alpha} \left(\epsilon_{k,l}^{a,b}(\alpha) : n_{kl} \geqslant \tau \right) \tag{4}$$

Finding the transformation resulting in the best fit between two edges requires $\epsilon_{k,l}^{a,b}(\alpha)$ to be evaluated for all α (for which $n_{kl} \geqslant \tau$ holds).

3 Descriptor Distance

The REC descriptor typically contains a set of edges that are the result of edge detection around the corresponding interest point. In reality we should expect that some perceptible edges may be missing or fragmented due to weak gradients and noise. An additional problem is related to the fact that only a subset of edges in the two descriptors may correspond well, while others are related to non-similar image structures. For example we can find patches on a giraffe skin with a high shape similarity at a local scale, but the random distribution of the patches makes shape comparison irrelevant on a large scale. Thus we have to search for a subset of edges in both descriptors, which together give a low fitting error, while other edges are rejected as outliers.

The primary idea behind the matching of multiple edges in the descriptors a and b is summarized below:

1. Perform edge fitting for admissible edge pair combination k and l, resulting in P putative transformations.
2. Repeat multiple edge fitting for P transformations. Choose the one which gives the lowest overall fitting error for the descriptor.
   (a) Rotate and scale all edges in descriptor b according to the current transformation and find the edge correspondences between two descriptors.
   (b) Remove outliers and calculate the final distance from all corresponding edge pairs.

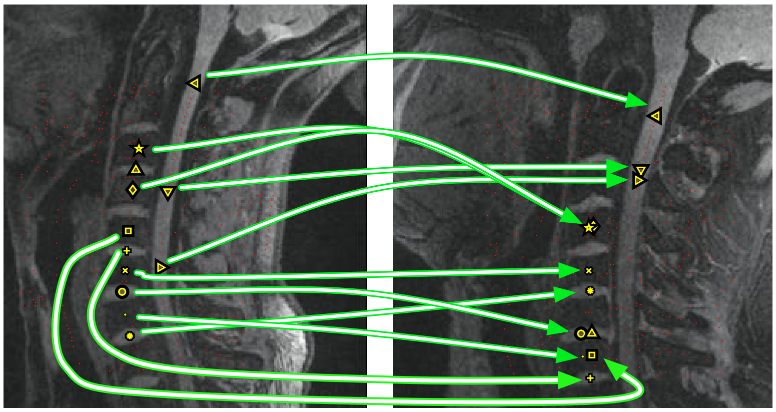
One of the most computationally demanding tasks is finding edge correspondences for a given relative scale and rotation. The major difficulty is associated with the possibility that a single edge in one descriptor may correspond to many non-overlapping edges in the other descriptor. An example of such multi-correspondences is shown in the Figure 2-b – edge $k2$ corresponds to edges $l2$ and $l4$, while edges $k4$ and $k3$ correspond to edge $l5$. Note that edge $l3$ could be also matched to the edge $k2$, but it overlaps with edges $l2$ and $l4$, which produce a better fit with edge $k2$. The process of finding edge correspondences can be divided into several steps:

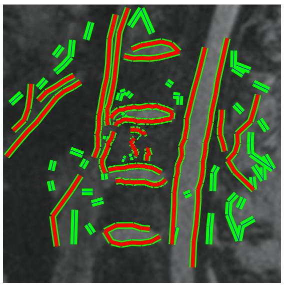
1. Find overlapping edge pairs in a: $\phi^{a}_{k1,k2} = \begin{cases} 1, \text{ if } k1 \text{ and } k2 \text{ overlap } \geqslant \tau \\ 0, \quad \text{otherwise} \end{cases}$.
2. Find overlapping edge pairs in b: $\phi^{b}_{l1,l2} = \begin{cases} 1, \text{ if } l1 \text{ and } l2 \text{ overlap } \geqslant \tau \\ 0, \quad \text{otherwise} \end{cases}$.
3. Find overlapping edge pairs between a and b: $\phi^{ab}_{k,l} = \begin{cases} 1, \text{ if } k \text{ and } l \text{ overlap } \geqslant \tau \\ 0, \quad \text{otherwise} \end{cases}$.
4. Find edge correspondence. The edge l is correspondent to edge k if:

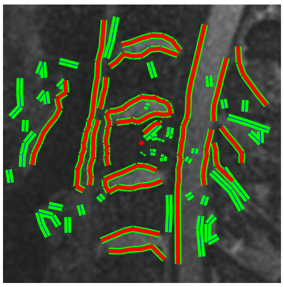
$$\epsilon^{a,b}_{k,l} = \min_{f,g}\left(\epsilon^{a,b}_{f,g} : f \in \{\phi^{ab}_{f,l} = 1 \wedge \phi^{a}_{f,k} = 1\}; g \in \{\phi^{ab}_{k,g} = 1 \wedge \phi^{b}_{l,g} = 1\}\right) \quad (5)$$

   which means that edges k and l correspond when the distance $\epsilon^{a,b}_{k,l}$ is the minimum among all combinations of edges f and g which overlap with k and l. This condition allows the association of multiple non-overlapping edges in one descriptor with a single edge in another descriptor.

During our matching tests we found that a simple outlier removal scheme helped to improve results when only a part of the structure in the two descriptors was found to correspond.







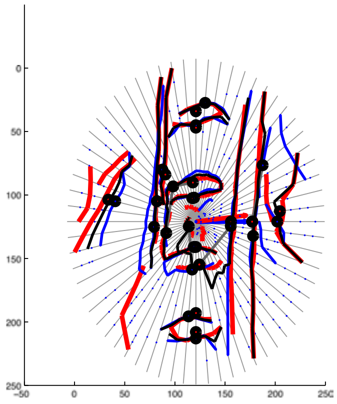


Fig. 3. Top row: example of descriptor matching between different MRI images. Only a representative subset of interest point matches is shown to avoid clutter. Bottom row: example of two similar image structures matched. The first two images show corresponding image patches and the extracted edges. The third image shows correspondence of edges from two descriptors (red and blue respectively) and the resulting edges after descriptor merging (black). Note that not all edges have been matched. We strongly advise to view all images in color.

Examples of finding similar image structures through the edge matching are presented in Figures 3, 4 and 5.

4 Descriptor Clustering

Clustering of descriptors is a frequently used technique in object recognition which allows for a compact (low-dimensional) representation of distinctive image structures to be obtained. Among the most popular clustering methods are hierarchical, k-means and kd-tree clustering. The primary difference between clustering of typical image descriptors and clustering of the REC descriptor is

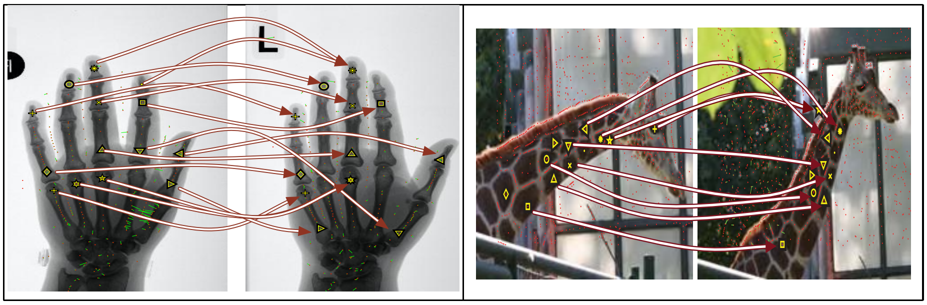


Fig. 4. Example of descriptor matching

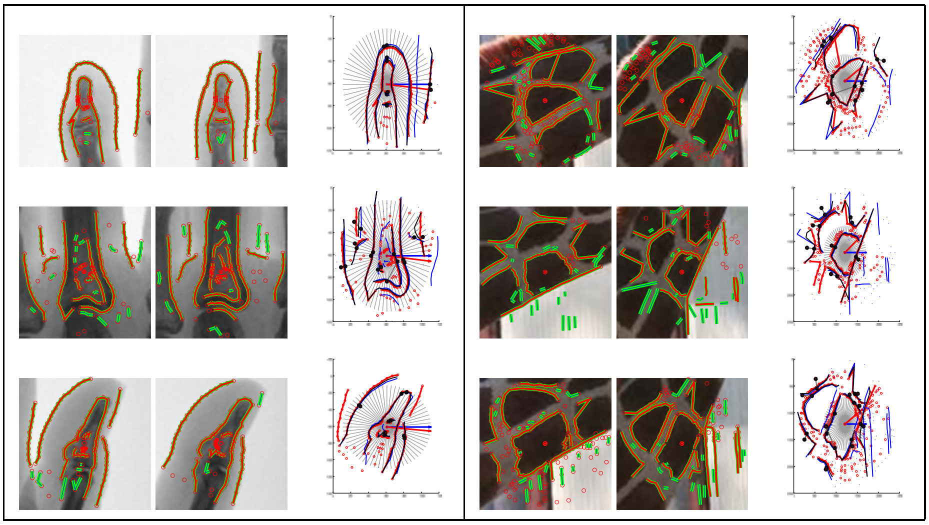


Fig. 5. Examples of edge matching in X-Ray images of hands and the giraffe skin

that the latter produces a variable length feature vector (the number of edges can vary significantly). This prevents the use of k-means and kd-tree clustering which require constant dimensionality of the feature vectors.

We cluster the REC descriptor using agglomerative hierarchical clustering [2] based on the REC distance defined in Section 3. The clustering is performed until the desired number of clusters is obtained or no more clusters can be produced.

Clustering starts with finding the closest pairs between a set of descriptors extracted from the training data set. The closest pairs are merged into nodes at the next clustering level and the same procedure is repeated on these nodes. The merging of two descriptors is an operation which generates a single edge for each set of corresponding edges in two descriptors as described in Section 3. Recall that a single edge in one descriptor can correspond to several edges in another descriptor and that some edges do not correspond at all, which means that they disappear in the merged descriptor. The edge kl, which is a result of merging of edges k and l, is obtained by averaging the boundary point positions from both edges:

$$\Gamma_{kl} = \{0.5(r_{k,i} + r_{l,i-\alpha \bmod N_0^+}) : i \subset \mathbb{N}; i = (b_{kl} ... b_{kl} + n_{kl}) \bmod N\} \tag{6}$$

In addition, each boundary point is assigned the weight corresponding to the distance between two merged boundary points. This way edges are prioritized according to their similarity.

$$w_{kl}(i) = \exp\left(- \left(1 - \frac{\max(r^a_{k,i}, \varsigma^{a,b}_{k,l} r^b_{l,\bar{i}})}{min(r^a_{k,i}, \varsigma^{a,b}_{k,l} r^b_{l,\bar{i}})} \right)^2 / \sigma^2 \right) \tag{7}$$

where σ is set to 0.25, which strongly penalizes relative scale outliers.

The result of clustering is a set of REC descriptors, which contain edges resulting from edge merging across a number of clustering levels. The weights assigned to the edges are then used during matching clusters to descriptors in the test data set. The edge distance (3) is then replaced with:

$$\epsilon^{a,b}_{k,l}(\alpha) = \frac{1}{n_{kl}} \sum_{i=b_{kl}}^{b_{kl}+n_{kl}} w^a_{k,i} \left| \log^2\left(\frac{r^a_{k,i}}{r^b_{l,\bar{i}}}\right) - \log^2\left(\varsigma^{a,b}_{k,l}(\alpha)\right) \right| \tag{8}$$

where descriptor a corresponds to the cluster and weights for descriptor b are set to 1.

5 Evaluation

Our testing strategy is focused on investigating the representative power of the REC descriptor. Tests are conducted on X-Ray images of human hands, and we want to assess how well the REC descriptor discriminates between four categories related to different finger bone types plus one background category (see Figure 6). To this end, descriptors extracted from the training set (10 images) are clustered and the resulting clusters are matched against descriptors from test set (20 images) with assigned category labels (obtained by manual annotation of a test set). The clustering process is expected to create consistent representations of similar shapes (shape alphabet) from the training data set (which also correspond to similar interest point locations). As clusters are themselves represented by single RECs, one can assess their representative quality by simply comparing them to labeled descriptors (see 4. We consider a shape cluster as highly consistent when it exhibits a majority of closest matches to descriptors stemming from a single category. However, the primary difficulty related to category discrimination is associated with the choice of categories and the fact that different types of bones share some similarities (especially on a local scale). Taking also into account some inconsistency in interest point distribution and poor edge detection, one can not expect that all clusters will be representative for one distinguished category.Therefore, we divide clusters in two groups: a highly informative group, with more than 50% matches within a single category and an uninformative group with the remaining clusters.

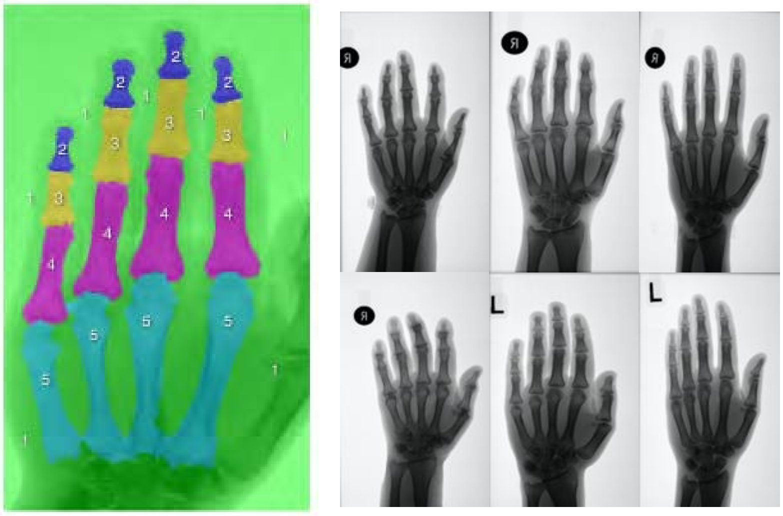


Fig. 6. Left: Example of hand X-Ray annotation. The categories represent 5 perceptible shapes, plus background. Note that this coarse annotation is not used during the training, but only during evaluation. The annotation defines association of interest points with the shape categories. Right: A subset of images used for clustering and testing. The shape and size of the bones varies between different images.

Another problem is the choice of the optimal number of clusters. Since each category covers four similarly shaped bones repeated in 10 training images, we should expect approximately 40 similar shapes per specific relative interest point location within a category. This corresponds to clustering level 5 and 6 where each clustering node corresponds to maximally 32 or 64 original REC descriptors merged together. We've chosen the 5-th clustering level, which results in over 600 clusters out of original approx. 5000 interest points.

Table 1 shows the average percentage of highly informative cluster matches in each category in the first column. The second column shows the percentage of interest points in each category, which correspond to the highly informative cluster matches.

The positive aspect revealed during these tests was the good consistency of highly informative clusters. We found that a large population of clusters scored more than 80% of matches within a single category. However, we found that in several cases less than 50% of interest points were matched against those clusters, which can be attributed to the following factors:

- Currently there is no procedure built into the clustering method, which would prevent the merging significantly different cluster nodes. This can happen when there is large imbalance in the number of interest points assigned to each category. Category 2 is at a particular disadvantage, accounting for only 2% of all interest points, while categories 3-5 correspond to 6, 9 and 23% of

Table 1. The cluster matching statistics for the database of hand X-Ray images corresponding to the highly informative group of clusters. The second column can be interpreted as the probability of detecting corresponding category when the related cluster has been matched to the descriptor associated with the interest point.

Category	Avg. cluster matches within same (correct) category [%]	Amount of descriptors matched to highly informative clusters [%]
1 - background	89.4	81.8
2 - Distal phalanges	68.6	33.6
3 - Intermediate phalanges	79.2	41.3
4 - Proximal phalanges	75.7	46.2
5 - Metacarpals	86.2	57.4

all interest points respectively. This negatively affects the clustering stage; clusters related to lowly populated categories are more likely to be merged with the clusters corresponding to highly populated categories.

- Small deviations in interest point positions increased the number of clusters needed to describe image structures related to the same category.
- Local shape similarity between categories – the central locations inside elongated bones, which account for the majority of interest points are similar to each other across multiple categories. This is one of the primary reasons why only 30-40% of all clusters are highly discriminative for categories 2 and 4.

6 Conclusions

We have presented an edge based semi-local shape descriptor (REC) together with a robust scale and rotation invariant distance measure. This allows us to perform unsupervised clustering of the descriptors in order to obtain a consistent representation of similar local image structures.

We have shown that REC descriptor is robust with respect to incorrectly detected, missing and fragmented edges and that it is possible to obtain a meaningful edge-based representation of stable shapes using clustering of descriptors. Future research will concentrate on the adaption of the clustering stage which will lead to a significant increase of cluster quality. Furthermore, we will focus our work on improvements with respect to the interest point selection and an exhaustive evaluation of our method using more diverse data sets.

Acknowledgements

This work was partly supported by the European Union Network of Excellence MUSCLE (FP6-507752), MOVEMENT (IST-2003-511670), Robots@home (IST-045350) and the Austrian Science Foundation (FWF) under grant SESAME (P17189-N04).

References

[1] Carmichael, O., Hebert, M.: Shape-based recognition of wiry objects. In: IEEE Conference On Computer Vision And Pattern Recognition, IEEE Press, Los Alamitos (2003)

[2] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience, Chichester (2000)

[3] Jurie, F., Schmid, C.: Scale-invariant shape features for recognition of object categories. In: International Conference on Computer Vision & Pattern Recognition, vol. II, pp. 90–96 (2004)

[4] Opelt, A., Pinz, A., Zisserman, A.: Incremental learning of object detectors using a visual shape alphabet. In: Proceedings of the CVPR, vol. 1, pp. 3–10 (2006)

[5] Szumilas, L., Donner, R., Langs, G., Hanbury, A.: Local structure detection with orientation-invariant radial configuration. In: Proceedings of the CVPR, vol. 1 (2007)

[6] Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: a comprehensive study. In: Beyond Patches workshop, in conjunction with CVPR (2006)

Learning 3D Object Recognition from an Unlabelled and Unordered Training Set

Raimund Leitner

CTR Carinthian Tech Research AG
Europastrasse 4/1, 9500 Villach/St. Magdalen, Austria
raimund.leitner@ctr.at
http://www.ctr.at

Abstract. This paper proposes an unsupervised learning technique for object recognition from an unlabelled and unordered set of training images. It enables the robust recognition of complex 3D objects in cluttered scenes, under scale changes and partial occlusion. The technique uses a matching based on the consistency of two different descriptors characterising the appearance and shape of local features. The variation of each local feature with viewing direction is modeled by a multi-view feature model. These multi-view feature models can be matched directly to the features found in a test image. This avoids a matching to all training views as necessary for approaches based on canonical views.

The proposed approach is tested with real world objects and compared to a supervised approach using features characterised by SIFT descriptors (Scale Invariant Feature Transform). These experiments show that the performance of our unsupervised technique is equal to that of a supervised SIFT object recognition approach.

Keywords: object recognition, unsupervised learning.

1 Introduction

Recent developments in local object recognition address two issues: (i) unsupervised learning [1,2] and (ii) multi-view object models [3,4,5,6]. Only a few approaches combine both [7,8], but require an ordered training set. We present an approach which learns multi-view feature models from correspondences found in an unlabelled and unordered training set for the recognition of 3D objects in cluttered scenes.

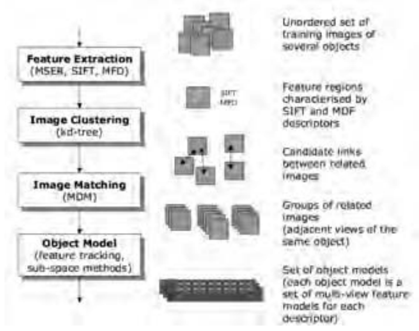
Local object recognition approaches which store canonical views to allow 3D object recognition [3,9] have the disadvantage that the features of an image must be matched to all training views. We avoid this by deriving multi-view feature models from the different training views of the features. These multi-view feature models can be matched directly to the features in an image. The advantages of multi-view feature models for supervised learning have been shown by [4,9,6].

The paper is organised as follows: section 2 gives an overview of the learning and recognition of the proposed approach and section 3 presents the methods in detail. The results are discussed in section 4. Conclusions are drawn and future work is proposed in section 5.

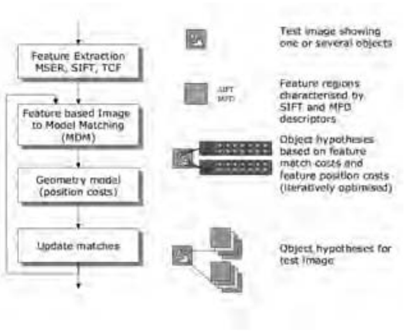
2 Overview

Both learning and recognition start with a feature extraction using Maximally Stable Extremal Regions (MSER [10]) followed by a characterisation of the region's appearance (SIFT [3]) and the region's contour (Modified Fourier Descriptor (MFD [11]). In the learning stage an image clustering based on a kd-tree [4] identifies initial sub sets of images containing the same object to avoid an exhaustive one-by-one matching of the entire training set. Within these sub sets pairs of images are matched using two different feature descriptors [7] to obtain local correspondences. A greedy optimisation algorithm determines an image order suitable for a tracking of the training features and the multi-view feature model generation. A local probabilistic model of the mutual feature positions completes the object models (Fig. 1 (a)).

As with learning, recognition starts with MSER detection and characterisation of the regions using SIFT and MFD. The main part of recognition is a matching of the feature descriptors of the image to the multi-view feature models of the object model. This process iteratively optimises the set of matches using the descriptor match costs and the mutual feature position costs (Fig. 1 (b)).


Feature Extraction (MSER, SIFT, MFD)
Image Clustering (kd-tree)
Image Matching (MDM)
Object Model (feature tracking, sub-space methods)
SIFT
MFD
Unordered set of training images of several objects
Feature regions characterised by SIFT and MDF descriptors
Candidate links between related images
Groups of related images (adjacent views of the same object)
Set of object models (each object model is a set of multi-view feature models for each descriptor)

(a)


Feature Extraction MSER, SIFT, TCF
Feature based Image to Model Matching (MDM)
Geometry model (position costs)
Update matches
SIFT
MFD
Test image showing one or several objects
Feature regions characterised by SIFT and MFD descriptors
Object hypotheses based on feature match costs and feature position costs (iteratively optimised)
Object hypotheses for test image

(b)

Fig. 1. The elements of the learning (a) and recognition (b)

3 Methods

3.1 Feature Extraction and Rectification

We use Matas' MSER algorithm to detect regions robust to viewpoint changes, scaling and illumination conditions [10]. Affine invariance is achieved by rectifying each region with an affine transformation so that an ellipse fitted to the region becomes a circle. This provides limited viewpoint invariance, as a perspective projection can be approximated locally by an affine transformation. The remaining view dependent variation (locally curved surfaces, shading, self occlusion, etc.) will be modeled by the multi-view feature model (cf. 3.6).

3.2 Appearance and Shape Descriptors

The appearance of the region is characterised using SIFT descriptors with a 4x4 partition of the region and an 8 bin orientation histograms as proposed by Lowe [4]. The shape descriptor is based on an extension of the Modified Fourier Descriptor (MFD) proposed by Rui [11]. We approximate the region's contour with a piecewise cubic spline to get a constant number of equally spaced sample points before the MFD calculation. This approximation reduces the quantisation effect of the pixel grid. Noise in the contour pixel locations (due to suboptimal segmentation of the region) is also less disturbing. A similarity distance is used for both descriptors (l denotes the number of elements of the descriptor):

$$d_{sim}(desc_1, desc_2) = \frac{1}{l} \sum |desc_1 - desc_2| \ . \tag{1}$$

3.3 Image Clustering

The training set consists of a set of unordered and unlabelled images and we want to avoid the combinatorial complexity of an exhaustive matching. Thus, the image clustering is used to partition the training set into several sub sets of images showing the same object. A kd-tree of SIFT descriptors [4] is used to determine a *view adjacency* $V_A(I_i, I_j)$ based on the number of feature pairs $(F_u, F_v), F_u \in I_i, F_v \in I_j$ having a similarity d_{sim} smaller than a chosen threshold t_{sim}. The distance

$$d_{clust}(I_i, I_j) = \frac{1}{1 + V_A(I_i, I_j)} \tag{2}$$

is used for a modified single linkage clustering which accepts a link only if an image is linked to the same cluster by at least one other image.

3.4 Image Matching

The image sub sets found by the image clustering are subjected to a detailed matching to remove false positives and refine the sub sets. We use a matching and verification procedure based on [7].

For both the appearance and shape descriptors two match cost matrices, $\mathcal{M}_A$ and $\mathcal{M}_S$, are determined with the distance function d_{sim} (cf. equation (1)). The optimal set of matches between both images with minimum matching costs is determined using the Munkres algorithm [12,13]. This is done separately for the appearance and shape descriptors to provide two sets of matches $\mathcal{A}_1$ and S_1. False matches are efficiently suppressed by accepting only matches present in both assignments by $\mathcal{M}_1 = \mathcal{A}_1 \cap \mathcal{S}_1$.

The number of correct matches can be increased by using the following assumption: The probability that the correct match for a certain feature is among the best, e.g. three, matches is higher than the probability that it is the best match. Consequently, we compute three *optimal* descriptor assignments subsequently by prohibiting the previously accepted matches (optimal assignment of the previous run) for the next run of the Munkres algorithm. $\mathcal{A}_{tot}$ denotes all possible matches between the regions of two images and the first run of the Munkres algorithm determines the assignment $\mathcal{A}_1$ which causes the lowest costs regarding to the appearance cost matrix M_A. By prohibiting the matches $\mathcal{A}_1$ for the next run of the Munkres algorithm, i.e. restricting the allowed matches to $\mathcal{A}_{tot} \backslash \mathcal{A}_1$, the second optimal assignment $\mathcal{A}_2$ is determined. Accordingly the k-th optimal assignment $\mathcal{A}_k$ is calculated by allowing only $\mathcal{A}_{tot} \backslash (\bigcup_{i=1}^{k-1} \mathcal{A}_i)$. By applying the same procedure to the shape descriptor matches, the final set of matches is found by $\mathcal{M}_n = (\bigcup_{i=1}^{n} \mathcal{A}_i) \cap (\bigcup_{i=1}^{n} \mathcal{S}_i)$. Empirical tests showed, that for most matching experiments $\mathcal{M}_3$ contains almost twice the number of correct matches and only a few false matches more than $\mathcal{M}_1$.

3.5 Image Order

Clustering the images partitioned the training set into sub sets for each object; the image matching determined dense matches in the image sub sets for each object. Our approach of using local correspondences to learn multi-view models of each feature requires that the images of each sub set are ordered by a view adjacency. In this paper we assume a linear order (view adjacency) of the images, thus each image has a predecessor and a successor. Assumed that the image matching has determined u pairs of images $P_i = (I_r(i), I_s(i))$. Based on these pairs we want to find an order for these m images which minimises the distance between adjacent images. We define a score function for an order O as

$$score(O) = \sum_{i=1}^{u} d(P_i) \tag{3}$$

$$d(P_i) = min(abs(O(I_r(i)) - O(I_s(i)))) \tag{4}$$

The order O is represented as a lookup table with m entries, one entry for each image. The algorithm starts with an initial order O_0 which is a random permutation of the numbers 1 to m. Six different operations are used iteratively to reduce the score of the order O_i:

1. **Pair operation (P):**
   (a) Swap: $[..., 4, 8, ...] \rightarrow [..., 8, 4, ...]$
   (b) Right Shift: $[..., 5..., 4, 8, ..., 9, ...] \rightarrow [..., 4, ..., 5, 9, ..., 8, ...]$
   (c) Left Shift: $[..., 3, ..., 4, 8, ..., 7, ...] \rightarrow [..., 4, ..., 3, 7, ..., 8, ...]$
2. **Block operation with n elements (Bn):**
   (a) Mirror Block: e.g. n=3: $[..., 3, 4, 5, ...] \rightarrow [..., 5, 4, 3, ...]$
   (b) Right Block Shift: e.g. n=3: $[..., 2, ..., 3, 4, 5, ...] \rightarrow [..., 5, ..., 2, 3, 4, ...]$
   (c) Left Block Shift: e.g. n=3: $[..., 3, 4, 5, ..., 6, ...] \rightarrow [..., 4, 5, 6, ..., 3, ...]$

At the beginning the set of allowed operations is $\mathcal{S}_0$=(P,B1). The set of allowed operations is changed if the score did not decrease in the last iteration. The set of allowed operations $\mathcal{S}_i$ is changed from *local* optimisations to more *global* ones: $\mathcal{S}_0$=(P,B1), $\mathcal{S}_1$=(P,B2),..., $\mathcal{S}_n$=(P,Bn).

3.6 Multi-view Feature Model

The correct order of images was estimated for each training sub set, which is the training data for a single object. A model of the object that consists of a set of multi-view feature models is then constructed. By using the image order to sequence the correspondences between images we can track the features over the ordered image sub set. The appearance and shape descriptors of each feature track are subjected to a PCA separately. Empirical tests showed that three principal components are sufficient for modeling the view dependent variation of the appearance and shape descriptors left by the rectification (locally curved surface, shading, self-occlusion, etc.).

3.7 Mutual Feature Positions

The mutual feature positions are modeled using Gaussian models $\mathcal{M}_{F_i}$ of the angle $\mathcal{N}_\alpha$ and distance $\mathcal{N}_d$ between each of the k features F_i of an object and its n nearest neighbours[1]. The nearest neighbours of F_i can be different from image to image, thus we take the n most frequent features within the nearest neighbors of feature F_i in the training images where F_i has been detected.

$$\mathcal{N} = (\mathcal{N}_d, \mathcal{N}_\alpha), \mathcal{N}_d = (\mu_d, \sigma_d), \mathcal{N}_\alpha = (\mu_\alpha, \sigma_\alpha) \tag{5}$$

$$\mathcal{M}_{F_i} = (N_0, N_1, ..., N_n) \tag{6}$$

$$\mathcal{M}_{FP} = (M_{F_0}, M_{F_1}, ..., M_{F_k}) \tag{7}$$

3.8 Recognition

Recognition is performed in a similar way to the image matching except that the image is matched to an object model. The reconstruction error d_{MARE} of the image feature and the feature model's principal components is used as similarity measure. To achieve robust recognition in case of background clutter and scaled or partially occluded objects, an iterative matching determines a set of matches that minimises the matching costs of the descriptors and the costs of the feature's mutual positions:

- **Initialisation:**
  Initialise $M_A(u,v) = d_{MARE}(A_u, A_v), M_S(u,v) = d_{MARE}(S_u, S_v)$ with A_u, S_u denoting the descriptors of the image feature and A_v, S_v the descriptor's principal components of the object model. Set $\delta_A = \frac{1}{2n}\sum M_A$ and $\delta_S = \frac{1}{2n}\sum M_S$
- **Step 1 of iteration i: Descriptor matching**
  A set of matches $\mathcal{M}_i$ with the minimal matching costs for M_A and M_S is determined using the matching approach described in section 3.4.
- **Step 2 of iteration i: Validation using mutual feature positions**
  Update the corresponding elements in $M_A(u,v) = M_A(u,v) + \delta_A$ and $M_S(a,b) = M_S(a,b) + \delta_S$ of each match in $\mathcal{M}_i$ between feature F_u and F_v if it is rejected by the mutual feature position model.

[1] The angle α is measured relative to the orientation and the distance d relative to the size of feature F_i.

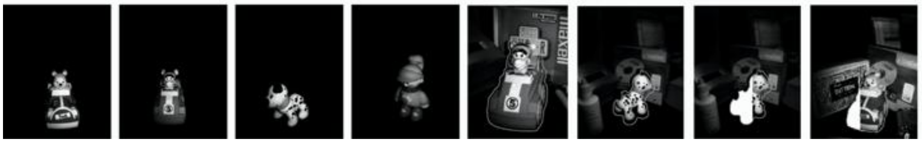


Fig. 2. Some training images of the ten objects in the training set (a)-(d). Some test images with background clutter and partial occlusion (e)-(h). The sucessfully matched features and the object hypotheses are superimposed.

Steps 1 and 2 are executed iteratively until the set of matches $\mathcal{M}_i$ converges. If the final set of matches contains more than three matches, the object is regarded as being recognised. This is repeated for all object models.

4 Results

The proposed approach has been used to learn the recognition of ten real world objects from a training set of 360 images (36 images for each object corresponding to a single rotation). Fig. 2 (a)-(d) shows four images of these ten objects from the unlabelled and unordered set of training images. Lowe showed that an image clustering based on a kd-tree of SIFT keys works properly [4] and we will not pursue the issue here. Consequently, we discuss only the results of the image matching, the image order reconstruction and the object recognition.

4.1 Image Matching

Both learning and recognition are based on the image matching described in 3.4. Fig. 3 shows the matching performance for rotated objects (a) and scale changes (b). The ratio of correct matches is not especially high, but the important advantage is shown in Fig. 3 (c). A matching based on SIFT descriptor alone is considerably sensitive to the threshold while the proposed approach is not. This is an important advantage as we do not have to optimise the threshold for a certain application. The proposed approach does not need any global geometric constraint (e.g. epipolar geometry) and could therefore deal also with non-rigid transformations, but this is beyond the scope of this paper.

4.2 Image Order

For visualisation purposes the images have been passed to the algorithm in the correct order ($I_0, ..., I_n$ of object 1, $I_0 ..., I_n$ of object 2, etc). However, the algorithm is initialised with a random order and thus, this information is not available for the algorithm. The order is optimised iteratively by applying appropriate operations which reduce the score as can be seen in Fig. 3 (a). The final order found for each of the ten objects is correct for 96% of the images (only 14 images of all 360 images are wrong). Fig. 4 (b) shows the reconstructed

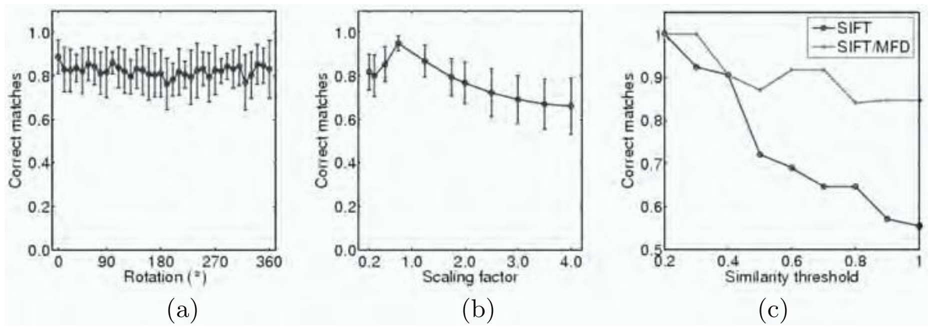


Fig. 3. Matching results for rotation (a) and scale changes (b). Ratio of correct matches to overall matches depending on a threshold for the similarity (c). While the ratio decreases considerably for the SIFT if larger thresholds are used, it remains above 85% for the combined SIFT/MFD matching.

image order graphically. The images of some objects are arranged in two separate rotations (in depth). However, this is valid as the algorithm does not *know* that the training data contain a single rotation in depth for each object only.

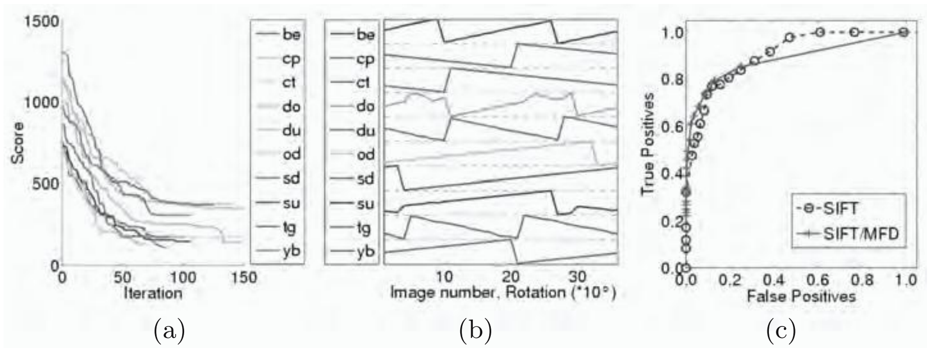


Fig. 4. The score decrease during the reconstruction of the image order (a) and the final image order (b). ROC curves of the SIFT approach and the proposed approach (SIFT/MFD) for the recognition in cluttered scenes.

4.3 Recognition

The proposed approach is compared to a supervised SIFT approach [3]. The true positive versus false positive rate (ROC curve) is shown in Fig. 4 (c). The recognition performance is similar although the proposed approach learns from an unlabelled and unordered set of training images and does not require a threshold for the descriptor similarity.

5 Conclusion and Outlook

We have presented an approach for unsupervised learning of a set of multi-view feature models from an unlabelled and unordered set of training images. The learning stage determines the sub sets of training images for the unknown number of objects and reconstructs an image order for the derivation of feature tracks. Multi-view feature models are generated from the feature tracks using sub space methods. The approach provides similar recognition performance to a supervised SIFT approach although the presented approach is completely unsupervised.

This paper used a training set with a linear view adjacency, i.e. a single rotation in depth for each object. As a next step we want to use training sets with a two dimensional adjacency, i.e. working on the upper half the viewing sphere and with a reduced number of training images. Another future issue will be the evaluation of the approach for the recognition of flexible objects.

References

1. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1842, pp. 18–32. Springer, Heidelberg (2000)
2. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 264–271. IEEE Computer Society Press, Los Alamitos (2003)
3. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proc. of the International Conference on Computer Vision ICCV, Corfu., pp. 1150–1157 (1999)
4. Lowe, D.: Local feature view clustering for 3d object recognition. In: IEEE CVPR (Conference on Computer Vision and Pattern Recognition), Kauai, Hawai, pp. 682–688 (2001)
5. Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: 3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In: CVPR (2003)
6. Ferrari, V., Tuytelaars, T., van Gool, L.: Integrating multiple model views for object recognition. In: CVPR, vol. II, pp. 105–112 (2004)
7. Leitner, R., Bischof, H.: Recognition of 3d objects by learning from correspondences in a sequence of unlabeled training images. In: DAGM (2005)
8. Thomas, A., Ferrari, V., Leibe, B., Tuytelaars, T., Schiele, B., van Gool, L.: Towards multi-view object class detection. In: IEEE Computer Vision and Pattern Recognition (CVPR), New York, IEEE Computer Society Press, Los Alamitos (2006)
9. Bülthoff, H.H., Wallraven, C., Graf, A.B.A.: View-based dynamic object recognition based on human perception. In: International Conference on Pattern Recognition, vol. 3, pp. 768–776 (2002)
10. Matas, J., Chum, O., Urban, M., Pajdla T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proc. BMVC, pp. 384–393 (2002)
11. Rui, Y., She, A.C., Huang, T.S.: Modified fourier descriptors for shpae representation - a practical approach. In: First International Workshop on Image Databases and Multi Media Search (1996)
12. Kuhn, H.W.: The Hungarian method for the assignment problem. In: Naval Research Logistics Quarterly, vol. 2, pp. 83–97 (1955)
13. Munkres, J.R.: Algorithms for the assignment and transportation problems. SIAM 5, 32–38 (1957)

Is Pinocchio's Nose Long or His Head Small? Learning Shape Distances for Classification

Daniel Gill[1], Ya'acov Ritov[1], and Gideon Dror[2]

[1] Department of Statistics, The Hebrew University, Jerusalem 91905, Israel
gill@mta.ac.il, yaacov@mscc.huji.ac.il

[2] Department of Computer Science, The Academic College of Tel-Aviv Yaffo, Tel-Aviv 64044, Israel
gideon@mta.ac.il

Abstract. This work presents a new approach to analysis of shapes represented by finite set of landmarks, that generalizes the notion of Procrustes distance - an invariant metric under translation, scaling, and rotation. In many shape classification tasks there is a large variability in certain landmarks due to intra-class and/or inter-class variations. Such variations cause poor shape alignment needed for Procrustes distance computation, and lead to poor classification performance. We apply a general framework to the task of supervised classification of shapes that naturally deals with landmark distributions exhibiting large intra class or inter-class variabilty. The incorporation of Procrustes metric and of a learnt general quadratic distance inspired by Fisher linear discriminant objective function, produces a generalized Procrustes distance. The learnt distance retains the invariance properties and emphasizes the discriminative shape features. In addition, we show how the learnt metric can be useful for kernel machines design and demonstrate a performance enhancement accomplished by the learnt distances on a variety of classification tasks of organismal forms datasets.

1 Introduction

The mathematical notion of shape is an equivalence class under certain type of group of transformations. The most common transformations are: translation, scaling, and rotation. This definition refers only to the question whether two shapes are identical, but in many cases we want to measure shape similarity or shape distance. Shape definitions in statistics were given by Bookstein [1] and Kendall [2], whose attitudes assume that correspondences between the two shapes are known. These latter approaches make sense while assuming that the two shapes are similar and have homologues features. A common and useful representation of planar shape is by landmark points. This approach can be easily extended to 3D shapes.

Parsimonious representation by landmarks has its advantages from the computational point of view and is very useful in many computer vision applications. Indeed, geometric features can represent the shape and location of facial components and are used in face analysis and synthesis [3]. Landmark analysis is also being used in medical imaging, robotics, dental medicine, anthropology, and many more applications.

In a supervised learning setting a desirable metric for shape classification should not only satisfy certain invariance properties but also capture the discriminative properties of the inputs. In this paper we present a learning algorithm which produces a metric, that satisfies these demands. Moreover, we show how this metric can be used for the design of kernel machines for shape classification.

2 Shape Space and Distances

A natural choice of landmarks is a finite set of particularly meaningful and salient points which can be identified by computer and humans. Several types of landmarks were suggested in previous works (see [1]). In the general case, there is a considerable loss of information by extracting only landmarks, and the transformed shape cannot be restored exactly from the landmarks. Yet, many essential characteristics may remain in such representation. A set of k ordered landmark points in 2D plane can be represented as a $2k$-dimensional vector. Comparing two shapes is usually based on corresponding landmarks which are termed *homologies*.

The general notion of distance (or similarity) between two shapes is quite vague. This term can be easily defined when using $2k$-dimensional vectors by taking only their coordinates as attributes. It is obvious that the order of the landmarks matters. Another convenient representation is called planar-by-complex and uses complex values to represent each 2-dimensional landmark point, so the whole shape is represented as an $k \times 1$ complex vector. The configuration matrix is a $k \times m$ matrix of real Cartesian coordinates of k landmarks in an m-dimensional Euclidian space. In a planar-by-complex representation the configuration is a k dimensional column vector of complex entries. From now on we will assume that all the shapes we deal with are two-dimensional and are given in the planar-by-complex representation.

2.1 Shape Metric

A desired distance measure between two planar landmark based shapes should be insensitive to translation, scaling and rotation. Consider a configuration $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k) \in \mathcal{C}^k$, a centered configuration $\mathbf{x}$ satisfies $\mathbf{x}^* \mathbf{1}_k = 0$, which is accomplished by: $\mathbf{x} \rightarrow \mathbf{x} - \mathbf{1}_k^T \mathbf{x} \mathbf{1}_k$, where $\mathbf{x}^*$ denotes the complex conjugate of $\mathbf{x}$.

The *full Procrustes fit* of $\mathbf{x}$ onto $\mathbf{y}$ is: $\mathbf{x}^P = (\widehat{a} + i\widehat{b})1_k + \widehat{\beta} e^{i\widehat{\vartheta}} \mathbf{x}$ where the parameters values $(\widehat{a}, \widehat{b}, \widehat{\beta}, \widehat{\vartheta})$ are chosen to minimize the Euclidean distance between $\mathbf{y}$ and the transformed configuration of $\mathbf{x}$, and their values are (see [4]):

$$\widehat{a} + i\widehat{b} = 0, \ \widehat{\vartheta} = \arg(\mathbf{x}^* \mathbf{y}), \ \widehat{\beta} = \frac{(\mathbf{x}^* \mathbf{y} \mathbf{y}^* \mathbf{x})^{\frac{1}{2}}}{\mathbf{x}^* \mathbf{x}} \tag{1}$$

Removing the similarity operations from a k-landmark planar configuration space leaves a $2k - 4$ dimensional shape space manifold (2 dimensions for translation, one dimension for scaling, and one dimension for rotation).

The *full Procrustes distance* between two configurations $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$d_F(\mathbf{x},\mathbf{y}) = \inf_{\beta,\vartheta,a,b} \left\| \frac{\mathbf{y}}{\|\mathbf{y}\|} - \frac{\mathbf{x}}{\|\mathbf{x}\|}\beta e^{i\vartheta} - a - bi \right\| = \left(1 - \frac{\mathbf{y}^*\mathbf{x}\mathbf{x}^*\mathbf{y}}{\mathbf{x}^*\mathbf{x}\mathbf{y}^*\mathbf{y}}\right)^{\frac{1}{2}}. \qquad (2)$$

The *full Procrustes mean shape* $\widehat{\boldsymbol{\mu}}$ of a set of configurations $\{\mathbf{w}_{i=1}^n\}$ is the one that minimizes the sum of square full Procrustes distances to each configuration in the set, i.e.

$$\widehat{\boldsymbol{\mu}} = \arg\inf_{\boldsymbol{\mu}} \sum_{i=1}^{n} d_F^2(\mathbf{w}_i, \boldsymbol{\mu}). \qquad (3)$$

It can be shown that the full Procrustes mean shape, $\widehat{\boldsymbol{\mu}}$, is the eigenvector corresponding to the largest eigenvalue of the following matrix:

$$S = \sum_{i=1}^{n} \frac{\mathbf{w}_i\mathbf{w}_i^*}{\mathbf{w}_i^*\mathbf{w}_i}. \qquad (4)$$

(see [5]). The eigenvector is unique (up to rotations - all rotations of $\widehat{\boldsymbol{\mu}}$ are also solutions, but these all correspond to the same shape) provided there is a single largest eigenvalue of S. In many morphometric studies several configurations are handled and pairwise fitted to a single common consensus in an iterative procedure [6]. This process is called *generalized Procrustes analysis*. Scatter analysis, using generalized Procrustes analysis handles the superimposed configurations in an Euclidean manner and provides good linear approximation of the shape space manifold in cases where the configurations variability is small.

Though Procrustes distances and least-squares superimpositions are very common, they can sometimes give a misleading explanation of the differences between a pair of configurations [6,7], especially when the difference is limited to a small subset of landmarks. The Procrustes superimposition tends to obtain less extreme magnitudes of landmark shifts. The fact that in least-squares superimposition landmarks are treated uniformly irrespective of their variance results in poor estimation, and reaches its extreme when all of the shape variation occurs at a single landmark, which is known as the *Pinocchio effect* [8]. This effect is demonstrated in Fig. 1. Due to proportions conservation, trying to minimize the sum-of-squares differences affects all landmarks and thus tilts the head and diminishes its size. Moreover, such variations affect the configuration's center of mass and thus affect translation as well. Actually, the Procrustes fit does not do what would have been expected from pre-classification alignment to do. A desirable fit would be an alignment that brings together non-discriminative landmarks and separates discriminative landmarks, and, in addition, gives appropriate weights for the features according to their discriminative significance.

2.2 General Quadratic Shape Metric

A general quadratic distance metric, can be represented by a symmetric positive semi-definite $k \times k$ matrix Q (we use the $Q = A^*A$ decomposition and estimate

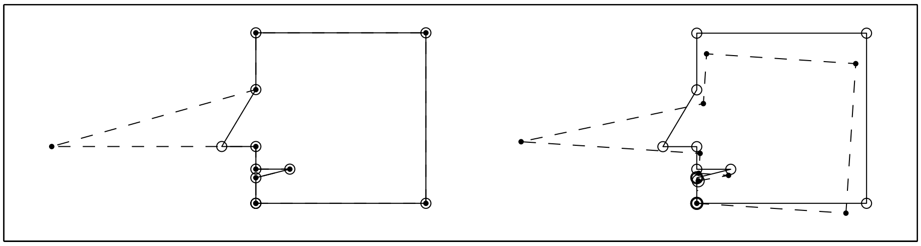


Fig. 1. Pinocchio effect. Two faces which differ only by the tip of their nose are superimposed by their similar features (*left*). Minimization of the sum-of-squares differences affects all landmarks of the longed-nose face: diminishes the head size and tilts it (*right*).

A). Centering a configuration $\mathbf{x}$ according to the metric induced by Q means that $\mathbf{x}^*Q\mathbf{1}_k = 0$, and this is done by: $\mathbf{x} \to \mathbf{x} - \mathbf{1}_k^T Q\mathbf{x}\mathbf{1}_k$. For the rest of this section, we assume that all configurations are centered according to the metric induced by Q.

The *general quadratic full Procrustes fit* of $\mathbf{x}$ onto $\mathbf{y}$ is:

$$\mathbf{x}^P = (\widehat{a}^Q + i\widehat{b}^Q)1_k + \widehat{\beta}^Q e^{i\widehat{\vartheta}^Q}\mathbf{x} \tag{5}$$

where the parameters values $(\widehat{a}^Q, \widehat{b}^Q, \widehat{\beta}^Q, \widehat{\vartheta}^Q)$ are chosen to minimize:

$$D_Q^2(\mathbf{x}, \mathbf{y}) = \left\| A\mathbf{y} - A\mathbf{x}\beta e^{i\vartheta} - A(a+bi)\mathbf{1}_k \right\|^2 . \tag{6}$$

Claim 1. *The minimizing parameters* $(\widehat{a}^Q, \widehat{b}^Q, \widehat{\beta}^Q, \widehat{\vartheta}^Q)$ *values are:*

$$\widehat{a}^Q + i\widehat{b}^Q = 0,\ \widehat{\vartheta}^Q = \arg(\mathbf{x}^*Q\mathbf{y}),\ \widehat{\beta}^Q = \frac{(\mathbf{x}^*Q\mathbf{y}\mathbf{y}^*Q\mathbf{x})^{\frac{1}{2}}}{\mathbf{x}^*Q\mathbf{x}}, \tag{7}$$

(the proof is similar to the Euclidean case).

The *general quadratic full Procrustes distance*, according to matrix $Q = A^*A$, between two configurations $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$\begin{aligned} d_Q^2(\mathbf{x}, \mathbf{y}) &= \inf_{\beta,\vartheta,a,b} \left\| A\frac{\mathbf{y}}{\|\mathbf{y}\|_Q} - A\frac{\mathbf{x}}{\|\mathbf{x}\|_Q}\beta e^{i\vartheta} - a - bi \right\| \\ &= \left(1 - \frac{\mathbf{y}^*Q\mathbf{x}\mathbf{x}^*Q\mathbf{y}}{\mathbf{x}^*Q\mathbf{x}\mathbf{y}^*Q\mathbf{y}}\right)^{\frac{1}{2}}, \end{aligned} \tag{8}$$

where $\|\mathbf{x}\|_Q^2 = \mathbf{x}^*Q\mathbf{x}$ is the square of the generalized norm.

The general quadratic Procrustes mean shape $\widehat{\mu}^Q$, with a matrix $Q = A^*A$, of a set of configurations $\{\mathbf{w}_i\}_{i=1}^n$ is the one that minimizes the sum of square generalized distances to each configuration in the set, i.e.

$$\widehat{\boldsymbol{\mu}}^Q = \arg\inf_{\boldsymbol{\mu}} \sum_{i=1}^n d_Q^2(\mathbf{w}_i, \boldsymbol{\mu}). \tag{9}$$

Claim 2. *The general quadratic Procrustes mean shape is the eigenvector corresponding to the largest eigenvalue of the following matrix:*

$$S^Q = \sum_{i=1}^{n} \frac{A\mathbf{w}_i \mathbf{w}_i^* A^*}{\mathbf{w}_i^* A^* A \mathbf{w}_i}, \tag{10}$$

(the proof is similar to the Euclidean case).

3 Metric Learning

Many pattern recognition algorithms use a distance or similarity measures over the input space. The right metric should fit the task at hand, and understanding the input features and their importance for the task may lead to an appropriate metric. In many cases there is no such prior understanding, but estimating the metric from the data might result in a better performance than that achieved by off the shelf metrics such as the Euclidean [9,10,11]. Fisher Linear Discriminant (FLD) is a classical method for linear projection of the data in a way that maximizes the ratio of the between-class scatter and the within-class scatter of the transformed data (see [12]).

Given a labeled data set consisting of 2D input configurations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ where $\mathbf{x}_i \in \mathcal{C}^k$ and corresponding class labels $c_1, c_2, \ldots, c_n$, we define between-class scatter and within-class scatter both induced by the metric Q. In a similar way to FLD the desired metric Q is the one that maximizes the ratio of the generalized between-class and within-class scatters.
We denote the general quadratic Procrustes mean shape of the members of class j by $\widehat{\boldsymbol{\mu}}_j^Q$, and the full general quadratic Procrustes mean shape of all configurations by $\widehat{\boldsymbol{\mu}}^Q$. Denote

$$\Delta_{k,l}^Q = \left(\frac{\mathbf{x}_l}{\|\mathbf{x}_l\|_Q} \widehat{\beta}_{k,l}^Q e^{i\widehat{\vartheta}_{kl}^Q} \right) - \widehat{\boldsymbol{\mu}}_k^Q \tag{11}$$

and

$$\Delta_k^Q = \widehat{\boldsymbol{\mu}}_k^Q \widehat{\beta}_k^Q e^{i\widehat{\vartheta}_k^Q} - \widehat{\boldsymbol{\mu}}^Q \tag{12}$$

where $\widehat{\beta}_{k,l}^Q, \widehat{\beta}_k^Q$ are the scaling solutions of eq. 8 for the l-th configuration towards the mean of class k, and scaling of the k-th mean configuration towards the global mean respectively. The angles $\widehat{\vartheta}_{kl}^Q, \widehat{\vartheta}_k^Q$ are those which satisfy eq. 8 for rotation the l-th configuration towards the mean of class k, and rotation of the k-th mean configuration towards the global mean correspondently (the translations equal to zero if all configurations are previously centered).

The within class scatter according to a matrix Q is:

$$s_W^Q = \sum_{j=1}^{m} \sum_{i=1}^{n} r_{ij} d_Q^2 \left(\mathbf{w}_i, \widehat{\boldsymbol{\mu}}^Q \right) = \sum_{j=1}^{m} \sum_{i=1}^{n} r_{ij} \left(\Delta_{j,i}^Q \right)^* Q \Delta_{j,i}^Q \tag{13}$$

where

$$r_{kl} = \begin{cases} 1 \ \mathbf{x}_l \in \text{Class } k \\ 0 \ \text{Otherwise} \end{cases} \tag{14}$$

and m is the number of classes.

The between class scatter according to a matrix Q is:

$$s_B^Q = \sum_{k=1}^{m} n_k \left(\Delta_k^Q\right)^* Q \Delta_k^Q, \tag{15}$$

where n_k is the number of samples belong to class k.

The desired metric Q_{opt} is the one that maximizes the ratio of the between-class scatter and within-class scatter:

$$Q_{opt} = \arg\max_Q \frac{s_B^Q}{s_W^Q}. \tag{16}$$

The rank of Q_{opt} is at most $m-1$. Contrary to the standard FLD, the suggested objective function f may have many local maxima. Thus, maximizing the objective function should be carried out carefully, and only a local maximum is guaranteed.

4 Procrustes Distance Based Classifiers

One of the goals of distance learning is the enhancement of the performance of classifiers. In recent years, many studies have dealt with the design and analysis of kernel machines [13]. Kernel machines use inner-products functions where the decision function is not a linear function of the data. Replacing the predefined kernels with ones that are designed for the task at hand and are derived from the data itself, is likely to improve the performance of the classifier considerably, especially when training examples are scarce [14]. In this section we introduce new kernels based on the general quadratic full procrustes distance where the learnt metric can be plugged in to produce new kernels with improved capabilities of shape classification.

4.1 General Quadratic Procrustes Kernels

Certain condition has to be fulfilled for a function to be a dot product in some high dimensional space (see Mercer's theorem [13]). Following the polynomial and radial basis function (RBF) kernels, we propose the following kernels.

Claim 3. *The following function is an inner product kernel for any positive integer p:*

$$k(\mathbf{x}, \mathbf{y}) = \left(\frac{\mathbf{y}^* Q \mathbf{x} \mathbf{x}^* Q \mathbf{y}}{\mathbf{x}^* Q \mathbf{x} \mathbf{y}^* Q \mathbf{y}}\right)^p \tag{17}$$

For proof outline see appendix A.

Claim 4. *The following function is an inner product kernel for any positive semi-definite matrix Q and any positive γ:*

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma\left(1 - \frac{\mathbf{y}^* Q\mathbf{x}\mathbf{x}^* Q\mathbf{y}}{\mathbf{x}^* Q\mathbf{x}\mathbf{y}^* Q\mathbf{y}}\right)\right) \tag{18}$$

For proof see appendix B.

5 Experimental Results

The main role of the general quadratic Procrustes metric described in the previous section is to align configurations in a way that reveals the discriminative features. Most of the datasets we examined were taken from the *shapes package* (http://www.maths.nott.ac.uk/personal/ild/shapes/), and they consist of organismal forms data. We used six datasets where the samples are represented by configurations of 2D landmarks, and each dataset is made of two categories. The datasets are: gorilla midline skull data (8 landmarks 30 females and 29 males), chimpanzee skull data (8 landmarks, 26 females and 28 males), orang utan skull data (8 landmarks 30 females and 30 males), mouse vertebrae (6 landmarks, 23 large mice and 23 small mice), landmarks taken in the near midline from MR images of the brain (13 landmarks 14 subjects diagnosed with schizophrenia and 14 normal subjects). In addition, we used a facial dataset consists of landmarks taken from frontal images of 32 males and 78 females - all with neutral expression. The extraction of the landmarks from the facial images was done by the Bayesian Tangent Shape Model (BTSM) [15].

Figure 2 uncovers discriminative landmarks in facial configurations means. The general quadratic Procrustes mean shape of females' faces is fitted using the learnt metric (general quadratic Procrustes fit) onto the general quadratic Procrustes mean shape of males' faces. It is evident that the learnt metric reveals differences between the two classes. The males' mandibles tend to be larger, and their forehead hairlines tend to be higher than those of females. These differences are not revealed when using the standard Procrustes metric.

The contribution of the Procrustes kernels and the learnt metric was evaluated by the leave-one-out error rate of three classifiers:

- SVM with standard RBF kernel where the input configurations are preprocessed by generalized Procrustes analysis onto the training samples full Procrustes mean shape.
- SVM with full Procrustes distance based RBF kernel ($Q = I$).
- SVM with learnt Procrustes distance based RBF kernel (learnt Q).

The leave-one-out error rates are given in Table 1. The results demonstrate two things: (i) The Procrustes kernel is preferable over the general Procrustes analysis followed by standard Euclidean based kernel (ii) The learnt metric improves the classifier performance.

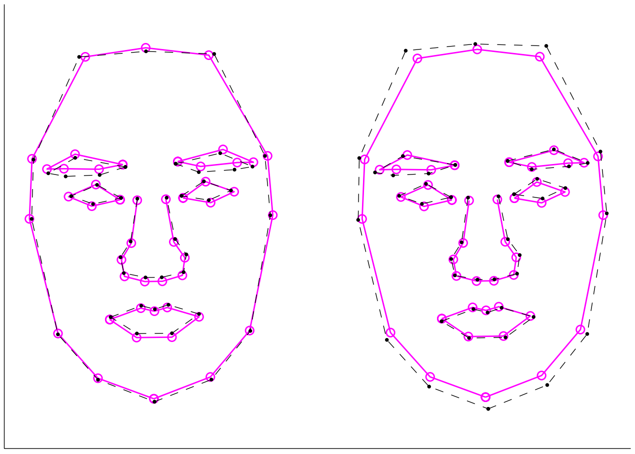


Fig. 2. Superimpositions of mean facial configurations: females (*solid line*) and males (*dashed line*) according to the full Procrustes metric (*left*) and the learnt Procrustes metric (*right*)

Table 1. Leave-One-Out error rates of the SVM classifiers

Dataset	Standard RBF	Procrustes Kernel ($Q = I$)	Learnt Procrustes Kernel
Gorilla Skulls	3.39%	3.39%	**0%**
Mouse Vertebrae	6.52%	4.35%	**2.17%**
Orang Utan Skulls	11.11%	5.56%	**3.70%**
Faces	12.73%	11.82%	**10.91%**
Chimpanzee Skulls	31.48%	31.48%	**25.93%**
Schizophrenia	32.14%	32.14%	**28.57%**

6 Discussion and Conclusions

We have presented an algorithm for learning shape distances, generalizing the Procrustes distance. In the two-classes case, the learnt metric induces a configuration superimposition where weights are assigned to the landmarks according to their discriminative role. Aligning configurations according to the learnt metric enables a visualization that uncovers the discriminative landmarks. Substantial improvement in classification performance was demonstrated by using Procrustes kernel (which keeps the pairwise full Procrustes distances between shapes, where generalized Procrustes analysis does not) and became even more pronounced when plugging in the learnt metric. The main contribution of the learnt metric is the meaningful alignment - it is of particular importance in cases where the training sets are small. Euclidean related kernels cannot learn translation, scaling, and rotation invariants from small data sets. Many types of devices for measuring 3D coordinates are in a wide-spread use: computed tomography

(CT), optical scans of surfaces (laser scanners), etc. All the methods discussed here can easily be extended to handle 3D configurations.

Acknowledgements

This work was supported in part by the Israel Science Foundation and NSF grant DMS-0605236.

References

1. Bookstein, F.: Morphometric Tools for Landmark Data: Geometry and Biology. Cambridge University Press, Cambridge (1991)
2. Kendall, D.: Shape manifolds, procrustean metrics, and complex projective spaces. Bull. London Math. Soc. 16, 81–121 (1984)
3. Li, S., Jain, A. (eds.): Handbook of Face Recognition. Springer, Heidelberg (2005)
4. Dryden, I., Mardia, K.: Statistical Shape Analysis, 1st edn. Wiley, Chichester (1998)
5. Kent, J.: The complex bingham distribution and shape analysis. Journal of the Royal Statistical Society Series B 56, 285–299 (1994)
6. Rholf, F., Slice, D.: Extensions of the procrustes method for the optimal superimposition of landmarks. Syst. Zool. 39, 40–59 (1990)
7. Siegel, A., Benson, R.: A robust comparison of biological shapes. Biometrics 38, 341–350 (1982)
8. Chapman, R.: Conventional procrustes approaches. In: Proceedings of the Michigan Morphometrics Workshop, pp. 251–267 (1990)
9. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning, with application to clustering with side-information. Advances in Neural Information Processing Systems 18 (2004)
10. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. Advances in Neural Information Processing Systems 18 (2004)
11. Globerson, A., Roweis, S.: Metric learning by collapsing classes. Advances in Neural Information Processing Systems 19 (2005)
12. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. John Wiley & Sons, Chichester (2001)
13. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
14. Aha, D.: Feature weighting for lazy learning algorithms. In: Liu, H., Motoda, H. (eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer, Norwell, MA (1998)
15. Zhou, Y., Gu, L., Zhang, H.J.: Bayesian tangent shape model: Estimating shape and pose parameters via bayesian inference. In: CVPR (2003)

Appendix A

Proof Outline: First we show that the following function is an inner product kernel for any positive integer p:

$$k(\mathbf{x},\mathbf{y}) = \left(\frac{\mathbf{y}^*\mathbf{x}\mathbf{x}^*\mathbf{y}}{\mathbf{x}^*\mathbf{x}\mathbf{y}^*\mathbf{y}}\right)^p \tag{19}$$

We have to show that this kernel satisfies Mercer's theorem. This is done by proving that:

$$\int\int\left(\frac{\mathbf{y}^*\mathbf{x}\mathbf{x}^*\mathbf{y}}{\mathbf{x}^*\mathbf{x}\mathbf{y}^*\mathbf{y}}\right)^p g(\mathbf{x})g^*(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0 \tag{20}$$

for any function g with finite l_2 norm.

Each term of the multinomial expansion has a non-negative value:

$$(r_1, r_2, \ldots, l_1, l_2, \ldots,)! \left\| \int \left(\frac{x_1^{r_1}, x_2^{r_2} \cdots \overline{x}_1^{l_1}, \overline{x}_2^{l_2} \cdots}{\|\mathbf{x}\|^{2p}} \right) g\left(\mathbf{x}\right) d\mathbf{x} \right\|^2 \geq 0 \tag{21}$$

and hence the integral is non-negative.

Showing that:

$$k(\mathbf{x},\mathbf{y}) = \left(\frac{\mathbf{y}^*Q\mathbf{x}\mathbf{x}^*Q\mathbf{y}}{\mathbf{x}^*Q\mathbf{x}\mathbf{y}^*Q\mathbf{y}}\right)^p \tag{22}$$

Satisfies Mercer's theorem is done in a similar way by using eigen-decomposition the non-negativity of Q's eigenvalues. ■

Appendix B

Proof

$$k(\mathbf{x},\mathbf{y}) = \exp\left(-\gamma\left(1 - \frac{\mathbf{y}^*\mathbf{x}\mathbf{x}^*\mathbf{y}}{\mathbf{x}^*\mathbf{x}\mathbf{y}^*\mathbf{y}}\right)\right) = \exp\left(-\gamma\right)\exp\left(\gamma\frac{\mathbf{y}^*\mathbf{x}\mathbf{x}^*\mathbf{y}}{\mathbf{x}^*\mathbf{x}\mathbf{y}^*\mathbf{y}}\right). \tag{23}$$

The first factor on the right side is positive and the second factor can be arbitrarily close approximated by polynomial of the exponent with positive coefficients, thus using claim 3 we have a sum of semi-definite functions, which is also a semi-definite function. ■

Probabilistic Combination of Visual Cues for Object Classification

Roman Filipovych and Eraldo Ribeiro

Computer Vision and Bio-Inspired Computing Laboratory
Department of Computer Sciences
Florida Institute of Technology
Melbourne, FL 32901, USA
rfilipov,eribeiro@fit.edu
http://www.cs.fit.edu/~eribeiro

Abstract. Recent solutions to object classification have focused on the decomposition of objects into representative parts. However, the vast majority of these methods are based on single visual cue measurements. Psychophysical evidence suggests that humans use multiple visual cues to accomplish recognition. In this paper, we address the problem of integrating multiple visual information for object recognition. Our contribution in this paper is twofold. First, we describe a new probabilistic integration model of multiple visual cues at different spatial locations across the image. Secondly, we use the cue integration framework to classify images of objects by combining two-dimensional and three-dimensional visual cues. Classification results obtained using the method are promising.

1 Introduction

The classification (and categorization) of objects and scenes from visual information is one of the most challenging problems in computer vision. Recent advances in image-based object classification have focused on statistical approaches modeling both the appearance of discriminative object parts [3,8] and the spatial relationship among these parts [6,8]. Such methods represent the state-of-the-art in both general object classification and object categorization. However, most object classification methods rely on measurements obtained from a single visual cue. While these methods work remarkably well for specific classes of images, they assume that information about a specific cue is always available. On the other hand, psychophysical evidence suggests that natural vision-based classification tasks are performed better when multiple visual cues can be combined to help reduce ambiguity [14].

In this paper, we address the problem of integrating multiple visual cues using a probabilistic framework. Our contribution is twofold. First, we describe a new model for the integration of a set of distinct visual cues using a Bayesian framework. We model both the cues' appearance information and their spatial structure. As a result, our model allows us to determine the different contributions of each cue in the classification process at different spatial locations in the object. Secondly, we use our integration framework to classify images of objects by combining two-dimensional and three-dimensional visual cues. Here, we use a robust shape-from-shading method [15] to estimate surface normals maps (i.e., needlemaps) of the objects in the images. Shape measurements obtained from the estimated needlemaps provide an approximate description of the 3–D geometry of the object's surface. Finally, our results show that the proposed method is able to obtain improvements in classification that go beyond the best rates obtained by individual cues.

The remainder of this paper is organized as follows. In Section 2, we commence by providing a review of the related literature. In Section 3, the details of our cue combination approach are described. Section 4 provides experimental results on two natural image databases. Finally, Section 5 presents our conclusions and plans for future investigation.

2 Related Literature

Recently, there has been considerable developments in part-based classification methods that model the spatial arrangement of object parts [8,7,5]. These methods are inspired by the original ideas proposed by Fischler and Elschlager [10]. For example, Fergus *et al.* [8] proposed a fully-connected part-based probabilistic model for object categorization. The approach is based on the constellation model proposed in [3]. Fergus' approach uses joint probability densities to describe an object's appearance, scale, shape, and occlusion. Shortcomings of the approach include a computational costly parameter estimation as well as a restrictively small number of object parts that can be modeled. The computational cost, in this case, can be addressed by representing the object's spatial structure using tree-structured graphical models [7,6,5].

Yet, most classification methods are based on measurements obtained from a single visual cue. Such an information may not always be available. This limitation can be addressed by combining information from multiple visual cues [13,4,1]. In this paper, we will focus on cue combination methods for object classification. For example, Nilsback and Caputo [13] proposed a cue integration framework based on the linear combination of margin-based classifiers such as vector machines. In another approach, Carbonetto *et al.* [4] combines local image features and region segmentation cues using semi-supervised learning. They also address the problem of selecting reliable local features for classification. However, no spatial structure for the cues is provided (i.e., the visual cues are assumed to be both available and reliable across the entire image). Probabilistic graphical models have also been used for visual cue integration [1], and applied to the problem of direction of figure (DOF) detection and disambiguation. Next, we describe the details of our probabilistic cue combination method.

3 Cue Integration Model

In the description that follows, we drawn our inspiration from recent work on constellation modeling of objects [3] and part-based object classification [6]. In

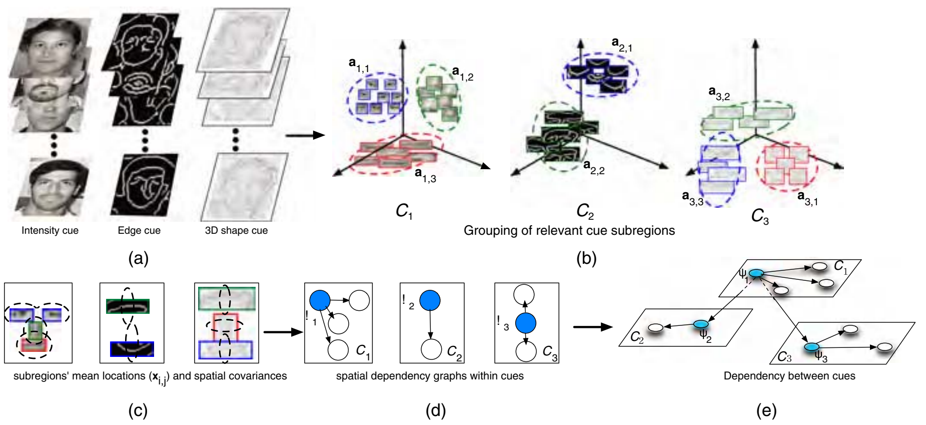

Intensity cue
Edge cue
3D shape cue
(a)
a1,1
a1,2
a1,3
C1
a2,1
a2,2
C2
a3,2
a3,3
a3,1
C3
Grouping of relevant cue subregions
(b)
subregions' mean locations (xi,j) and spatial covariances
(c)
spatial dependency graphs within cues
(d)
Dependency between cues
(e)

Fig. 1. Visual cue integration

this paper, we show how similar ideas can be applied to the problem of multiple visual cues integration.

We begin by defining the main components of our cue integration model. Let $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ represent a set of K visual cues extracted from an image $\boldsymbol{\mathcal{I}}$ of an object (e.g., edge maps, surface normals, color). The goal of our approach is twofold. First, we aim at integrating the information provided by multiple visual cues in a principled manner. Secondly, we will use this integration model for the classification objects in images. Probabilistically, the likelihood of observing a particular image given that an object is at some location can be represented by the distribution $p(\boldsymbol{\mathcal{I}}|\boldsymbol{\mathcal{X}})$ where $\boldsymbol{\mathcal{X}}$ represents a particular spatial configuration of the visual cues associated with the object. From the Bayes' theorem, we obtain:

$$p(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{I}}) \propto \underbrace{p(\boldsymbol{\mathcal{I}}|\boldsymbol{\mathcal{X}})}_{\text{likelihood}} \underbrace{p(\boldsymbol{\mathcal{X}})}_{\text{prior}} \propto \underbrace{p(\boldsymbol{\mathcal{C}}|\boldsymbol{\mathcal{X}})}_{\text{appearance}} \underbrace{p(\boldsymbol{\mathcal{X}})}_{\text{spatial configuration}} \tag{1}$$

In (1), $\boldsymbol{\mathcal{I}}$ was substituted by $\boldsymbol{\mathcal{C}}$ to indicate that the image information will be represented by a set of visual cues. Our cue integration model follows the factorization of Equation 1 suggested by Felzenszwalb and Huttenlocher [7]. The underlying idea in this factorization is that the spatial arrangement of object parts can be encoded into the prior probability distribution while the likelihood distribution encodes the appearance of the object (and its parts). In this paper, we focus ourselves on the representation of both the appearance and the spatial configuration of multiple visual cues.

3.1 Spatial Prior Model

The prior distribution in Equation 1 is described as follows. We assume that each available visual cue $\mathcal{C}_i$ can be subdivided into a number of non-overlapping subregions such that $\mathcal{C}_i = \{(\mathbf{a}_1^{(i)}, \mathbf{x}_1^{(i)}), \ldots, (\mathbf{a}_{N_{\mathcal{C}_i}}^{(i)}, \mathbf{x}_{N_{\mathcal{C}_i}}^{(i)})\}$, where each pair $(\mathbf{a}_j^{(i)}, \mathbf{x}_j^{(i)})$

represents both the local appearance $\mathbf{a}$ and the spatial location $\mathbf{x}$ of the subregion j for the cue $\mathcal{C}_i$. Here, $N_{\mathcal{C}}$ is the total number of subregions within a cue. At this point, we would like to introduce the concept of reliability of a visual cue. Cue reliability can be interpreted as the significance (i.e., contribution) of the cue in the integration process [13]. It can be measured, for example, in terms of the consistency of a cue with respect to the overall recognition rates of the integrated model. In classification problems, it is often the case that some cues are more reliable than others. Moreover, visual cues may not be consistently available and reliable across the entire image. For instance, edge maps may be more reliable at certain image regions while pixel intensity at others.

In this paper, we model the cue integration process and cue reliability using a tree-structured directed acyclic graph [2] in which conditional dependencies represent the implicit cue reliability in the spatial global integration model. Figure 1 illustrates our cue integration concept. Here, the arrows in the graph indicate the conditional dependence between the connected vertices (i.e., less-reliable cues are conditioned to the most reliable one). For simplicity, our model uses the star graph structure suggested by Fergus *et al.* [9]. Here, a particular vertex is assigned to be a landmark vertex $(\mathbf{a}_r^{(i)}, \mathbf{x}_r^{(i)})$. The remaining vertices are conditioned on the landmark vertex. Thus, the dependence between cues in our model is explicitly modeled through the connections between landmark vertices. It should be noted that the dependence between cue regions is based solely on their spatial location as we assume that visual cues are independent with respect to their appearance. We will discuss this assumption in more detail later.

Bayesian Network Factorization. We now introduce the factorization for the prior distribution term in (1). This probability distribution models the spatial interaction among extracted visual cues. When modeling the dependency relationships between cues, we first assume that cues are ordered based on their reliability values. This assumptions allows us to arrange the cues' landmark vertices as a star-shaped tree structure in which landmark vertices of less-reliable cues are conditioned on the landmark vertex of the most reliable cue. For simplicity, we consider the location of the landmark subregion to represent the center of the underlying image cue. Figure 1 illustrates an example of a Bayesian network describing this modeling. Accordingly, the joint distribution for the cues spatial interaction can be derived from the graphical model shown in Figure 1, and is given by:

$$p(\boldsymbol{\mathcal{X}}) = p(\psi_r) \prod_{i \neq r} p(\psi_i | \psi_r) \tag{2}$$

where ψ_i corresponds to spatial configuration of the i-th cue, and the probability distributions that compose Equation 2 are:

$$p(\psi_r) = p(\mathbf{x}_r^{(1)}) \prod_{j \neq r} p(\mathbf{x}_j^{(1)} | \mathbf{x}_r^{(1)}) \tag{3}$$

$$p(\psi_i | \psi_r) = p(\mathbf{x}_r^{(i)}) \prod_{k \neq r} p(\mathbf{x}_k^{(i)} | \mathbf{x}_r^{(i)}) \tag{4}$$

Appearance Model. We now focus ourselves on the appearance term of Equation 1. We assume that the appearance of the visual cues and as well as their corresponding non-overlapping subregions are independent. Under the independence assumption, the appearance likelihood of the combined cues can be factorized into the product of the individual subregions' likelihoods. The likelihood function in (1) becomes:

$$p\left(\boldsymbol{\mathcal{C}}|\boldsymbol{\mathcal{X}}\right) = \prod_{i}^{K} p\left(\mathcal{C}_i|\boldsymbol{\mathcal{X}}\right) = \prod_{i}^{K}\prod_{j}^{N_{\mathcal{C}_i}} p(\mathbf{a}_j^{(i)}|\mathbf{x}_j^{(i)}) \tag{5}$$

Appearance independence is a reasonable assumption for non-overlapping subregions within a single visual cue. However, this independence might be sometimes difficult to achieve. For instance, for visual cues such as color and edges, abrupt changes in color distribution may induce the occurrence of edges on the same image. We plan to study the effects of this assumption in our future work. Next, we describe the learning and recognition stages of our method.

3.2 Learning

The factorization described in Equation 2 and Equation 5 allows for the learning process to be performed in a modular fashion given a set of training images $\{\boldsymbol{\mathcal{I}}_1, \ldots, \boldsymbol{\mathcal{I}}_M\}$. The learning process is divided into two main steps. First, the algorithm estimates the appearance model parameters for each representative subregion in each visual cue layer. Secondly, the parameters representing the spatial configuration of cues are determined. The main learning steps of our algorithm are detailed as follows. For simplicity, we make use of Gaussian densities for conditional probabilities in the model.

Learning the Appearance of Visual Cues' Subregions. In this step, the parameters of the appearance model (Equation 5) are estimated. We commence by extracting a set of visual cues from the training images (e.g., surface normals, edge maps, color, and geometric measurements).

1. **Detect and extract representative subregions in each visual cue.** In this step, each visual cue image is divided into a number of subregions centered at locations provided by an interest feature detector. It should be noted that each visual cue conveys a different type of visual information. Consequently, a cue-specific feature detector should be used for each cue type. For simplicity, we first locate regions of interest in the gray-level image cue using the feature detector described in [12]. We then use these locations to extract multiscale subregions of interest in the remaining visual cues using a Gaussian pyramid approach.
2. **Grouping similar subregions.** The subregions obtained in the previous step are subsequently processed by a semi-supervised learning algorithm to determine the most representative non-overlapping subregions in each cue

map. Here, the method requires two types of input. The first one is a set of positive (i.e., images containing the target object) and a set of negative (i.e., background images) training images. The second input of the method is the number of representative parts to be learned in each cue layer (i.e., $N_{\mathcal{C}_i}$). In our current implementation, we use a modified K-Means clustering method for grouping the representative object parts while giving preference to non-overlapping image subregions. The centroid of the largest K-Means clusters are chosen to be the representative parts of the object for the underlying visual cue. Representative parts of an object are the ones that do not appear frequently in background images.

3. **Appearance likelihood parameter estimation.** Under the Gaussian assumption, the parameters of Equation 5 can be directly estimated by simple calculations of the sample mean vectors and sample covariance matrices of each representative part learned during the clustering step. This step of the learning process is illustrated in Figures 1–(a) and 1–(b).

Learning the Spatial Prior. In this step, our main goal is to estimate the parameters of the spatial prior of the cue integration model (Equation 2). Our method for learning the spatial prior is divided into two main steps. First, for each visual cue, landmark vertices of each cue are chosen to be the ones that can be consistently located in the positive training images dataset. The remaining subregions are conditioned to the landmark ones. The conditioning step is illustrated by the graphs in Figure 1–(d). Secondly, we determine the best global cue dependency configuration using exhaustive search based on the overall classification rate (Figure 1–(e)). We estimate the most likely spatial configuration of the learned parts' locations by selecting the joint probability distribution that allows for the maximum overall recognition rate. The main steps of this stage are described as follows.

1. **Learn the location uncertainty of learned subregions.** The goal of this step is to determine possible locations of the parts previously learned by the algorithm. This is equivalent to a template matching operation. In our implementation, we simply select the image location with maximum likelihood of the part appearance $p(\mathbf{a}_j^{(i)}|\mathbf{x}_j^{(i)})$. The mean location and covariance matrix of each part location is estimated.
2. **Determine the joint Gaussian probability of part locations.** Here, the conditional probabilities of the subregion locations in Equation 2 are estimated using Gaussian joint probability distributions. It can be shown that the conditional distributions relating independent Gaussian distributions are also Gaussian. As a result, the terms $p(\mathbf{x}_k^{(i)}|\mathbf{x}_r^{(i)})$ in (2) take a particularly simple form [2]. Figure 1–(c) shows elliptical shapes illustrates the spatial location uncertainty of each representative part in the model.
3. **Cue probabilities and global model probability.** Once the model parameters for the spatial configuration of subregions within each visual cue

are at hand, the integration of all available visual cues is accomplished by estimating the parameters of Gaussian joint probabilities as described in Equations 3 and 4.

3.3 Recognition and Detection

Once the parameters of the cue integration framework are estimated, the problem of recognizing (and in this case, also locating) an object in an image can then be posed as follows: we seek the location in the image that maximizes the posterior probability of the location of the object given a set of visual cues as given in (1):

$$\boldsymbol{\mathcal{X}}^* = \arg \max_{\boldsymbol{\mathcal{X}}} p(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{C}}) \tag{6}$$

An exact inference using the model described in (2) is computationally intractable. A possible way to overcome this problem is to make use of approximate inference methods (i.e., belief propagation, expectation-maximization). In this paper, the approach is to first detect every cue individually and then obtain the probability of object configuration using the global relationship among cues. Here, we follow the inference approach suggested in [6]. The recognition stage is accomplished using the following main steps.

1. **Determine part locations.** We begin by extracting a set of image subregions located at interest points in a similar fashion as in the first step of the appearance learning procedure.
2. **Appearance and spatial configuration.** Calculate the appearance likelihood based on the appearance of the parts as described by Equation 5. Determine the probability of spatial configuration of cue parts as in Equation 2. Select the model with the maximum overall posterior probability.

4 Experimental Results

In this section, we assess the potential of our cue combination model for the problem of object recognition. Here, we describe the experimental results performed on two sets of real-world images. The first of these consists of a dataset of marine biofouling organisms (i.e., barnacles). These organisms are usually found attached to the hull of ships and have a dome-like shape. The second dataset used in our experiments consists of images from the Caltech face database. Images samples from both datasets are shown in Figure 2. The choice of the class of images is aimed at demonstrating the ability of our model to integrate both 2–D and 3–D visual cues. The images used in our experiments were divided into subsets of training, validation, and test images. The sizes of the subsets were 100, 100, and 300 images, respectively. Each subset contained an equal amount of object images and background images.

We commenced by processing all images to obtain a set of visual cues represented by maps of pixel intensity, edges, and 3–D shape information. The pixel

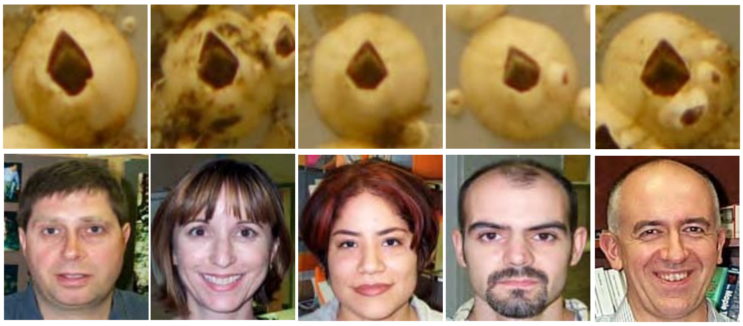


Fig. 2. Sample of the images used in our experiments. Row 1: images of barnacles (i.e., marine biofouling organisms). Row 2: face images from the Caltech face database.

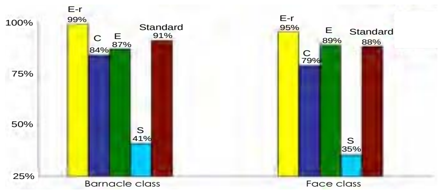


Fig. 3. Recognition rates obtained for barnacles and faces. (E-r) Our probabilistic cue integration model; (C) Classification rate using gray-level intensity only; (E) Classification rate using edge-map only; (S) Classification rate using estimated surface normals only; (Standard) Simple linear combination of cues.

intensity map consisted of simple gray level versions of the images. The edge map information was obtained using the Canny edge detector. Finally, the third visual cue was obtained from surface normals (needlemaps) estimated using the robust shape-from-shading method proposed by Worthington and Hancock [15]. Surface normals from shape-from-shading as a single visual cue have been recently used for object recognition [15].

In our experiments, we created two sets of graphical models. For the barnacle class, we created two, four, and one part star-graph models to represent the gray-level intensity, edge, and 3–D shape cues, respectively. The choice of spatial configuration was determined automatically to maximize the classification rates for each individual visual cue.

We also experimented our method on images from the Caltech face database. For these images, the edge cue was the most reliable one. For the face class, we used two-vertex graphs for both the gray-level pixel intensity and the edge map cues. The 3–D information was represented by a single landmark vertex graph.

Finally, our experiments include a comparison between our cue integration framework and the one described in [11]. In [11], the final recognition score is calculated as the linear combination of the individual cues' maximum posterior probabilities. Additionally, we provide a comparison between the recognition results obtained using our model and the results obtained using each cue individually. Figure 3 shows the results of this comparative study. The results indicate that the combined model provides significantly better recognition rate when compared to results obtained based on single cues only. Our preliminary results also show that our method outperforms the cue integration method based solely on linear combination for the class of images used in this paper.

5 Conclusions and Remarks

In this paper, we proposed a probabilistic model for the integration of visual cues for object recognition. We drew our motivation from recent probabilistic part-based models for object recognition. Here, we derived a Bayesian framework for multiple cue integration. Our model was able to represent the different contributions of each cue in the recognition process at different spatial locations in the object. We also combined information from 2–D and 3–D modalities. Finally, our experiments showed the effectiveness of our method for object recognition.

The work presented in this paper represents an attempt to accomplish cue integration in a principled way. The use of a probabilistic framework that describes cue dependencies consists of a natural integration approach that allows for the inclusion of prior information while permitting the learning of models from training data.

The proposed method have many avenues for improvement. The experiments presented in this paper are preliminary. However, they serve to show the potential of the proposed cue integration method. The use of only three cues is limiting and a larger number of cues should be added to assess the behavior of the method. A study of the selection of difference dependences and cue would also be helpful. Additionally, the use of only two object classes is somehow insufficient. We plan to extend the experiments to a larger number of object classes.

The use of 3-D shape information is clearly a desirable feature in a cue integration framework. In this paper, we added shape information in the form of surface normals provided by a shape-from-shading algorithm. Unfortunately, using surface normals directly does not provide an appropriate treatment of the 3-D information due to often low quality of the measured data. Improvements can probably be obtained by using alternative shape representations such as shape-index and spin-images.

Finally, we have arbitrarily selected the types of cues to use in our model. In principle, any visual cue can be used by the method. However, for simplicity, we have selected a set of commonly used cues in image analysis. A more comprehensive study of effective cues and their representation within the cue integration framework is needed. Work addressing the above issues is currently underway and it will be published in due course.

References

1. Baek, K., Sajda, P.: A probabilistic network model for integrating visual cues and inferring intermediate-level representations. In: IEEE Workshop on Statistical and Computational Theories of Vision, Nice, France, IEEE Computer Society Press, Los Alamitos (2003)
2. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics), Secaucus, NJ, USA. Springer, New York (2006)
3. Burl, M.C., Weber, M., Perona, P.: A probabilistic approach to object recognition using local photometry and global geometry. In: Burkhardt, H., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1407, pp. 628–641. Springer, Heidelberg (1998)
4. Carbonetto, P., Dorkó, G., Schmid, C., Kück, H., de Freitas, N.: A semi-supervised learning approach to object recognition with spatial integration of local features and sementation cues. In: Towards category-level object recognition, Springer, Heidelberg (2006)
5. Carneiro, G., Lowe, D.: Sparse flexible models of local features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3953, pp. 29–43. Springer, Heidelberg (2006)
6. Crandall, D., Felzenszwalb, P.F., Huttenlocher, D.P.: Object recognition by combining appearance and geometry. In: Toward Category-Level Object Recognition, pp. 462–482 (2006)
7. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. Int. J. Comput. Vision 61(1), 55–79 (2005)
8. Fergus, R., Perona, P., Zisserman, A.: Weakly supervised scale-invariant learning of models for visual recognition. Int. J. Comput. Vision 71(3), 273–303 (2007)
9. Fergus, R., Perona, P., Zisserman, A.: A sparse object category model for efficient learning and exhaustive recognition. In: CVPR 2005. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (June 2005)
10. Fischler, M., Elschlager, R.: The representation and matching of pictorial structures. IEEE Transactions - Computers 22, 67–92 (1977)
11. Hayman, E., Eklundh, J.-O.: Probabilistic and voting approaches to cue integration for figure-ground segmentation. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 469–486. Springer, Heidelberg (2002)
12. Kadir, T., Zisserman, A., Brady, M.: An affine invariant salient region detector. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 228–241. Springer, Heidelberg (2004)
13. Nilsback, M., Caputo, B.: Cue integration through discriminative accumulation. In: Conf. on Computer Vision and Pattern Recognition, pp. II: 578–585 (2004)
14. Tanaka, K., Saito, H., Fukada, Y., Moriya, M.: Coding visual images of objects in the inferotemporal cortex of the macaque monkey. Journals of Neurophysiology 66(1), 170–189 (1991)
15. Worthington, P.L., Hancock, E.R.: Object recognition using shape-from-shading. IEEE Trans. Pattern Anal. Mach. Intell. 23(5), 535–542 (2001)

Hill Climbing Algorithm for Random Sample Consensus Methods

Timo Pylvänäinen[1] and Lixin Fan[2]

[1] Nokia Research Center
timo.pylvanainen@iki.fi
[2] Nokia Research Center
fanlixin@ieee.org

Abstract. We propose a modification of RANSAC that performs guided search of the sample space. The sampling is applicable to any of the sample consensus methods, such as MAPSAC or MLESAC. We give simulation results which show that the new method can reduce the number of required iterations to find a good model by orders of magnitude.

1 Introduction

Many machine vision problems boil down to model fitting to certain data points, or parameter estimation. In the presence of outliers, the standard least squares fitting does not produce the desired results. Several methods can be used in the presence of outliers. For line fitting, a common approach is the Hough Transform[1,2]. For higher dimensional problems, the most widely used solution may be the RANSAC algorithm[3].

The computational complexity of Hough Transform and the purely random nature of RANSAC has lead some people to seek for more systematic searches of the parameter space. Few notable solutions are an application of Tabu-search[4] and the use of genetic algorithms[5]. Both methods, in essence, are based on the idea of iteratively improving a current solution. The current solution is used to limit the search space in some fashion.

In the context of multiple view matching, improvements have been proposed to deal with drawbacks of the basic RANSAC algorithm. In terms of evaluation object function, RANSAC inliers score nothing while outliers are given a constant penalty. Better performance was obtained by using MAPSAC or MLESAC where each inlier is scored based on how well they fit the estimated model [6,7]. MAPSAC and MLESAC, in essence, extend the discrete binary like scoring of RANSAC to a continuous range of scores.

To improve the random model sampling stage, different methods have been proposed to take advantage of information other than original data points. Tordoff and Murray proposed to use matching priors from the previous matching stage to *guide* the sampling of models [8]. A similar idea is pursued in PROSAC where the model sampling is based on a subset of "good" data points, based on domain specific matching score at a previous processing step[9]. The notion of applying a local optimization to so-far-the-best model was proposed in [10]. The local search, however, was a disjointed step from RANSAC model sampling. Myatt et al. proposed a biased sampling of the minimal set where only one point is selected from the uniform distribution and consecutive points are weighted relative to the Euclidean distance to the initial point[11]. P. Meer et al. addressed the problem of estimation of inlier noise scales and multiple structure fitting [12,13].

In this paper we give a few remarks on the standard analysis of RANSAC and show that its performance can be dramatically improved by using more informed model sampling strategy. To this end, we propose a simple data point weighting method, which favors a local hill climbing search instead of blind random sampling. In contrast to existing guided sampling approaches in [8][9], the proposed method is completely based on original data points and does not require any information from the matching stage. The algorithmic and computational cost of including this approach is minimal and thus it can be used for any RANSAC-like – such as MLESAC – approaches. Finally, we give simulation results that compare traditional RANSAC and the new hill climbing strategy.

2 Terms and Definitions

We define a model as a parametrized function $f(\boldsymbol{x} \mid \theta)$, where $\boldsymbol{x}$ is a data point and θ are the model parameters. All possible data points form a set D. A point $\boldsymbol{x} \in D$ is said to perfectly fit the model when $f(\boldsymbol{x} \mid \theta) = 0$. Given a parameter ε we say that the inlier set for model θ is

$$I_\varepsilon(\theta) = \{\boldsymbol{x} \in D \mid f(\boldsymbol{x} \mid \theta) < \varepsilon\}. \tag{1}$$

A minimal subset is a subset of D that contains just enough data points to define a model. For instance, in line fitting any two points in D would constitute a minimal subset.

3 RANSAC

The basic principle of RANSAC is to randomly select a minimal subset of s points from D, find the model θ defined by these s points and compute

$$c(\theta) = |I_\varepsilon(\theta)| \tag{2}$$

This is repeated for some number of iterations while keeping track of the model $\hat{\theta}$ for which $c(\hat{\theta})$ is the highest thus far.

The basic analysis of RANSAC focuses on the fraction ω of inliers (to the optimal model). Given p, the required probability of finding the right model, the required number of iterations is supposedly given by

$$N = \frac{\log(1-p)}{\log(1-\omega^s)}. \tag{3}$$

For somewhat more elaborate discussion on this analysis, see e.g. [14]. We will come back to this analysis in the following sections.

MAPSAC and MLESAC generalize RANSAC so that $c(\theta)$ can be seen as the posterior probability or the likelihood of the model θ in certain probabilistic formulation[6,7]. This way, the definition of a good model can take into account possible feature information associated to the data points. Nevertheless, both approaches adopt the same sampling strategy as standard RANSAC.

4 Random Sampling

The basic principle in all the xSAC methods is the random sampling of the model space. The minimal subsets of D actually define a proposal distribution where certain models are more likely than others.

We are looking for the point which maximizes $c(\theta)$. Usually there are more than one model that will give the maximal value for $c(\theta)$, so we define

$$G = \{\theta \in R_m \mid \forall \gamma \in R_m : c(\gamma) \leq c(\theta)\}, \tag{4}$$

the set of optimal models. An optimal model proposal distribution should put all probability mass on G, but clearly uniform sampling of minimal sets of D does not produce this distribution.

The probability of selecting a model θ is given by

$$p_\theta(\theta) = \begin{cases} \frac{H_D(\theta)!(N-s)!}{N!(H_D(\theta)-s)!} & \text{when } H_D(\theta) \geq s, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $H_D(\theta) = |\{\boldsymbol{x} \in D \mid f(\boldsymbol{x} \mid \theta) = 0\}|$. That is, H_D is the Hough transform for the data D, and $H_D(\theta)$ the number of data points that fit the model θ exactly.

When the total number of data points N and the values of $H_D(\theta)$ are large, this can be approximated by

$$p_\theta(\theta) \approx \begin{cases} \left(\frac{H_D(\theta)}{N}\right)^s & \text{when } H_D(\theta) \geq s, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

For RANSAC to succeed, it must find some model in G. The probability of this is simply

$$P(\theta \in G) = \sum_{\theta \in G} p_\theta(\theta). \tag{7}$$

The typical analysis of RANSAC assumes that any s inlier points will produce a model in G. This assumption is flawed. For instance, two inlier points can be on opposing sides of the line and would thus define a model orthogonal to the desired model. The simulations in Section 6 will verify that, indeed, the true probability of finding the desired model with RANSAC can be significantly lower than the standard analysis would indicate.

5 Hill Climbing Random Sampling

Observations in the previous section motivate a new method of sampling for the xSAC family. While many models based on inlier points are not optimal, they nevertheless capture many inlier points within their threshold. Some of these inlier points will lead to yet a better solution and it is possible to do a kind of randomized hill climbing algorithm.

For each model, we define a neighborhood $U_D(\theta_i) = \{\theta_k \mid H_{I_\varepsilon(\theta_i)}(\theta_k) \geq s\}$. That is, the set of models that can be generated by the points in the inlier set of θ_i. Instead of randomly jumping around the model space, we focus our attention to the neighborhood set and find a good model there. We then go on to study the neighborhood of this new, better model and in this way step by step find the optimal model.

To this end, we propose a new weighting strategy to increase the probability of drawing models from $U_D(\theta_i)$. To each data point $\boldsymbol{x}_i$ associate a weight $w_\theta(\boldsymbol{x}_i)$ such that

$$w_\theta(\boldsymbol{x}_i) = \begin{cases} 1 & \boldsymbol{x}_i \notin I_\varepsilon(\theta) \\ |I_\varepsilon(\theta)| & \boldsymbol{x}_i \in I_\varepsilon(\theta). \end{cases} \tag{8}$$

Models are generated by selecting s points from D with probability proportional to $w_\theta(\boldsymbol{x}_i)$ instead of the uniform distribution. In other words, the probability of selecting an inlier to the current model is increased proportional to the number of inliers for the current model. A positive weight is also assigned to outlier points so that it is always possible to move to any state from any other state.

The probability of selecting model θ_k is now given by the neighborhood distribution $\mathcal{N}(\theta_k \mid \theta_i)$. Define

$$\omega_{\theta_i}(\theta_k) = \frac{\sum_{\boldsymbol{x}\in S} w_{\theta_i}(\boldsymbol{x})}{\sum_{\boldsymbol{x}} w_{\theta_i}(\boldsymbol{x})}, \tag{9}$$

where $S = \{\boldsymbol{x} \mid f(\boldsymbol{x} \mid \theta_k) = 0\}$ and we have

$$\mathcal{N}(\theta_k \mid \theta_i) \approx \begin{cases} \omega_{\theta_i}(\theta_k)^s & \text{when } H_D(\theta_k) \geq s, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The neighborhood distributions also define a transition kernel of a Markov chain.

Since we expect a path of good but non-optimal models that lead to the optimal model, we add a rejection sampling step to the above Markov chain. Namely, after drawing a sample from $\mathcal{N}(\theta_k \mid \theta_i)$ we accept the new state θ_k only if $c(\theta_k) > c(\theta_i)$. In this new Markov chain, any state in G is an absorbing state. The algorithm is summarized in Algorithm 1.

Just as gradient descent or standard hill climbing algorithms require sufficient continuity from the objective function, the proposed sampling works well only when the data contains points that define suboptimal models that will nevertheless cover subsets of the optimal inlier set. Experience tells us that in many applications data exhibits this kind of continuity property and the method can significantly improve the efficiency of xSAC methods. The simulations in the following section also validate the method.

Algorithm 1. Hill Climbing RANSAC

```
1. Initialize $w(\boldsymbol{x}_i)$ to 1 for all points $\boldsymbol{x}_i$.
2. Draw $s$ data points with probabilities proportional to $w(\boldsymbol{x}_i)$ and compute the
   model $\theta$ defined by these points.
3. If $c(\theta) > c(\hat{\theta})$ set $\hat{\theta} = \theta_k$ and recompute $w(\boldsymbol{x}_i)$ according to Equation 8 for each
   data point $\boldsymbol{x}_i$.
4. Repeat 2–3 until appropriate stopping criterion has been reached.
```

6 Experimental Results

6.1 Line Fitting

To analyze the differences in performance between the traditional RANSAC sampling and the hill climbing algorithm, we used a simple line fitting example.

First a random line segment of length one was selected. Inlier points were generated by randomly selecting points on the line and adding a normally distributed noise component. Outliers were generated from an uniform distribution over a 4×4 square.

For the first simulation, 200 inliers and 200 outliers were generated. The noise component for the inliers had a standard deviation of 0.004. The inlier threshold was set to 0.02.[1]

A total of 250 different random data sets were tested, and both algorithms run for each data set for 400 times. Both algorithms were run until a model was found that had at least as many inlier points as the known true model and the number of iterations required were collected. It is possible to generate a data set where no two points define a model that would produce the desired inlier set. To avoid this each generated data set was tested with RANSAC to produce the desired model under 10000 iterations. If this was not the case, then the data set was replaced until a proper data set was obtained.

For the hill climbing algorithm, a rejected model is, of course, also counted as an iteration. We are therefore fairly counting the number of models that must be evaluated against the objective function for each algorithm. The computational overhead of one hill climbing iteration over RANSAC iteration is negligible. A histogram of the number of iterations required was collected from these 100000 test runs. The results are shown in Figure 1.

The standard analysis of RANSAC would indicate that for 99 percent chance of finding the right model, RANSAC should run for a little over 16 iterations. From the histogram, however, we can compute that the 99-percentile is 59.0 iterations. At the same time, the 99-percentile for the hill climbing algorithm is only 18 iterations. The mean number of iterations required were 9.8 and 4.9 respectively.

[1] Different inlier thresholds lead to different results, but for reasonable thresholds Hill-Climbing always seems to perform better than RANSAC in this test.

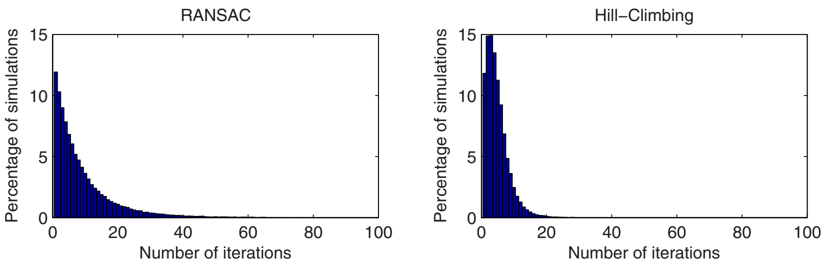


Fig. 1. Histograms of the number of iterations required to find the correct model for RANSAC and the hill climbing algorithm. At each x, the height of the bar indicates the percentage of simulation runs that found the true model at exactly x iterations.

6.2 Hyperplane Fitting

The performance of RANSAC is well known to dramatically depend on dimension. Many practical problems are multidimensional, such as ellipse fitting and finding the fundamental matrix. Each has some special characteristics and they are not directly comparable. To test the effect of dimension alone, we extended the line fitting example to hyperplane fitting in an arbitrary dimension d.

First inlier points were randomly generated on a $d-1$ dimensional hypercube with side length of 1 and at least one point within the unit ball at origin. White noise with standard deviation $\sigma = 0.004$ or $\sigma = 0.0001$ on each component was added to the inliers. Outliers were generated from an even distribution over the d dimensional hypercube which spans from -2 to 2 on each axis. Inlier threshold was set to 0.03 to keep the required iterations reasonable even at higher dimensions.

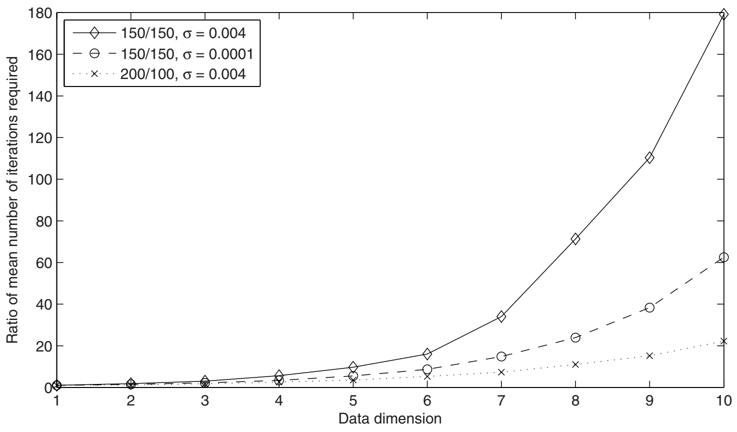


Fig. 2. The ratio of the mean number of iterations required for RANSAC and the hill climbing algorithm for different inlier/outlier and noise combinations as a function of dimension. The legend reads as follows: inliers / outliers, σ = noise standard deviation.

For each dimension from 1 to 10, 50 data sets were generated and both algorithms tested on each data set 50 times and the mean of the required iterations was computed for each dimension and algorithm. The test was run for different number of inliers and outliers. The ratio of the mean number of iterations for each test are shown in Figure 2.

We can clearly observe that when the ratio of outliers and noise in inliers increases, the benefit of using the hill climbing strategy becomes very significant. While for one dimensional data RANSAC can actually perform slightly better, for 10 dimension data with 50 percent outliers and reasonable amount of noise the hill climbing strategy requires only 1 iteration for every 180 iterations of RANSAC. In practical computer vision problems, such as fitting the fundamental matrix to extracted corner points, 50 percent of outliers can be quite common.

6.3 Epipolar Geometry Estimation

The proposed hill climbing algorithm was tested on the problem of estimating the epipolar geometry from image point correspondences. Five pairs of images shown in Figure 3 were tested. For each pair of images, SIFT descriptors [15] were first extracted and then matched by minimizing the total cost of one-to-one assignment, i.e. by solving the Assignment Problem with Hungarian algorithm [16]. Tentative matching results are rectified by running both RANSAC and Hill-climbing algorithms. It is observed that the speedup of Hill-climbing over RANSAC ranges between 4 up to over 100. The results are summarized in Table 1. We can observe that the speedup ratio increases dramatically as outlier ratio increases.

A possible stopping criterion for the Hill-climbing algorithm is to stop after

$$N = \frac{\log(1-p)}{\log(1-\omega_{\hat{\theta}}(\hat{\theta})^s)} \tag{11}$$

iterations, where $\hat{\theta}$ is the best model found so far. This is a a modified version of Equation 3, taking into account of the proposed weighting scheme. This stopping criterion was used for the Hill-climbing algorithm in this test.

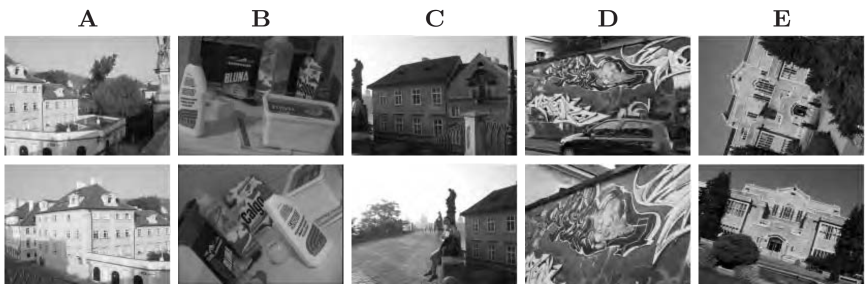


Fig. 3. Image pairs used in epipolar geometry estimation. Image pairs A,B,C are from [17]. D and E are from INRIA image dataset.

Table 1. Epipolar geometry fitting, using hill-climbing and RANSAC algorithms. AVGI is the number of inliers found when the algorithm stops averaged over 100 tests. Iter is the average number of iterations the algorithm required before stopping. Speedup is the ratio of required iterations.

	Pair A	**Pair B**	**Pair C**	**Pair D**	**Pair E**
Num. points	378	259	308	206	672
Num. inliers	242	115	133	71	202
RANSAC AVGI	237.3	109.5	129.4	68.1	190.3
H-C AVGI	238.9	113.4	130.9	70.1	197.4
RANSAC Iter	191.5	5380.5	5066.5	27048.0	94710.4
H-C Iter	41.2	231.0	221.9	604.0	908.1
Speedup	**4.7**	**23.3**	**22.3**	**44.8**	**104.3**

6.4 Ellipse Fitting

The simulations above were all for linear models. Intuitively, such models have no local optima and the hill climbing is expected to work well. A standard computer vision problem, to which RANSAC is applied, is ellipse fitting. Five points are required to determine a conic on the plane.When these points are noisy and represent only a small segment of the full curve of the ellipse, then small local ellipses can result even from a set of all inliers, as depicted in Figure 4.

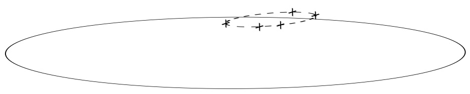


Fig. 4. Models based on points on one side of the desired ellipse often lead to small "parasite" ellipses, due to noise

As expected, local optima are a problem for a hill climbing algorithm, and hill climbing RANSAC is no exception. Simulations show that the hill climbing algorithm for the ellipse fitting can perform as much as three times worse than standard RANSAC.

It is possible, however, to remedy the situation. To obtain a good model, it is crucial that the defining points are from opposing sides of the target ellipse. Many local optima, however, only cover one side of the ellipse. By slightly altering the sampling, so that three of the five points are selected using the weighing scheme described above and the remaining two points from the uniform distribution over all points, the performance is significantly improved. An ellipse fitting using this kind of hybrid sampling exceeds RANSAC performance by a factor of two. This kind of sampling is not beneficial for the linear models.

A simulation similar to the linear case was run for ellipse fitting, for 150 inliers and 150 outliers. The inlier set thresholding was based on algebraic distance. In this test RANSAC required approximately 57 iterations on average, the hill climbing algorithm over 150 iterations, but the hybrid sampling only needed about 30 iterations on average.

The nature of a particular problem defines the extrapolation properties of local inlier neighborhoods. Local inlier sets of ellipses do not extrapolate well to the whole ellipse and the method requires some domain specific adjustments. While the method may not be suitable for every application, augmenting existing xSAC implementations with the hill climbing strategy requires so little effort that simply testing the effect in a particular application is a viable option.

7 Conclusions and Future Work

We have shown that the hill climbing algorithm clearly has the potential to dramatically improve, by orders of magnitude, the probability of selecting a good model. It is also extremely easy to implement as it only replaces the point selection phase in RANSAC. Compared with other modifications of RANSAC, the proposed algorithm has some favorable features:

– The notion of guided sampling was proposed in [8][9], where efficient sampling was achieved by using information from matching stage. It is, however, unclear how these approaches can be applied to domains other than image matching (e.g. line fitting), in which matching scores are not readily available. The hill climbing algorithm, on the other hand, is completely based on original data points and lends itself to being used in different domains, as demonstrated in simulation section.
– The proposed hill climbing algorithm can be used to augment existing xSAC algorithms, such as MAPSAC and MLESAC. Since the replacement of sampling strategy is independent of evaluation object functions, it is straightforward to combine the hill climbing weighting scheme with different evaluation functions used.

Finally, an interesting problem remains open. The Markov chain for the hill climbing algorithm forms a stochastic process with a stopping time τ_G defined as the first iteration the process reaches G. τ_G is a random variable and the properties of this random variable may be of interest. Unfortunately, the properties are based on the data set D and it is unclear what kind of statements we can make about D. Such an analysis would, however, lay a solid theoretical foundation for the hill climbing RANSAC algorithm.

References

1. Goldenshluger, A., Zeevi, A.: The hough transform estimator. The Annals of Statistics 32, 1908–1932 (2004)
2. Kultanen, P., Xu, L., Oja, E.: Randomized hough transform. In: Proceedings of the International Conference on Pattern Recognition, pp. 631–635 (1990)
3. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24, 381–395 (1981)
4. Jiang, T.: Geometric primitive extraction by the combination of tabu search and subpixel accuracy. Journal of Computer Science and Technology 14, 74–80 (1999)

5. Roth, G., Levine, M.D: Geometric primitive extraction using a genetic algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 16, 901–905 (1994)
6. Torr, P.H., Zisserman, A.: MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding 78, 138–156 (2000)
7. Torr, P.H., Davidson, C.: IMPSAC: Synthesis of importance sampling and random sample consensus. IEEE Transactions on Pattern Analysis and Machine Intelligence 25, 354–364 (2003)
8. Tordoff, B.J., Murray, D.W.: Guided-MLESAC: Faster image transform estimation by using matching priors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1523–1535 (2005)
9. Chum, O., Matas, J.: Matching with PROSAC - progressive sample consensus. In: Proceedings of the Computer Vision and Pattern Recognition, pp. 220–226 (2005)
10. Chum, O., Matas, J., Kittler, J.: Locally optimized RANSAC. In: Michaelis, B., Krell, G. (eds.) Pattern Recognition. LNCS, vol. 2781, pp. 236–243. Springer, Heidelberg (2003)
11. Myatt, D., Bishop, J., Craddock, R., Nasuto, S., Torr, P.H.: NAPSAC: High noise, high dimensional robust estimation — it's in the bag. In: Proceedings of the Britsh Machine Vision Conference, pp. 458–467 (2002)
12. Chen, H., Meer, P., Tyler, D.: Robust regression for data with multiple structures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1069–1075. IEEE Computer Society Press, Los Alamitos (2001)
13. Subbarao, R., Meer, P.: Beyond RANSAC: User independent robust regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop, IEEE Computer Society Press, Los Alamitos (2006)
14. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2003)
15. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, pp. 1150–1157 (1999)
16. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Inc., Upper Saddle River, NJ. USA (1982)
17. Matas, J.: `http://cmp.felk.cvut.cz/~wbsdemo/demo/wbsmain.php`

FPGA Implementation of a Feature Detection and Tracking Algorithm for Real-time Applications

Beau Tippetts, Spencer Fowers, Kirt Lillywhite, Dah-Jye Lee, and James Archibald

Dept. of Electrical and Computer Eng., Brigham Young University, Provo, UT USA

Abstract. An efficient algorithm to detect, correlate, and track features in a scene was implemented on an FPGA in order to obtain real-time performance. The algorithm implemented was a Harris Feature Detector combined with a correlator based on a priority queue of feature strengths that considered minimum distances between features. The remaining processing of frame to frame movement is completed in software to determine an affine homography including translation, rotation, and scaling. A RANSAC method is used to remove mismatched features and increase accuracy. This implementation was designed specifically for use as an onboard vision solution in determining movement of small unmanned air vehicles that have size, weight, and power limitations.

1 Introduction

As high computational requirements are the nature of computer vision and image processing applications, real-time performance is not a trivial accomplishment. It is even more difficult when the nature of a specific application limits the size of the computing system, or the amount of power that can be consumed. With small unmanned air vehicles, or micro UAVs, there are a multitude of applications for onboard image processing, but the ability of the craft to maneuver in the air is severely hindered if it has to be tethered to a power supply or overloaded with the weight of large processing systems.

In order to provide image processing solutions onboard micro UAVs, a balanced combination of software algorithms and hardware implementations needs to be determined. This paper details work to provide an onboard image processing solution of detecting and tracking features in order to estimate the movement or homography from one frame to the next for micro UAV applications.

Feature detection forms the basis of many UAV applications. It is the initial step in developing computer understanding of video sequences. Detected features are tracked and classified as obstacle/non-obstacle in order to implement basic obstacle avoidance. Obstacle avoidance can be used to help a UAV safely avoid trees, power lines, and other obstacles. Features can also be identified as a desired target and then used to maintain a specified distance from the target, effectively tracking any identifiable object. Tracked features may also be combined with line detection or color segmentation in order to implement path or lane following, allowing the UAV to map out potential routes for ground vehicles. These features can be used to obtain information about the UAVs surroundings, such as height above ground, pitch, roll, direction of movement, and speed.

Image processing solutions involving feature detection and homography calculation currently exist for micro UAV applications, but are all done remotely on a ground station computer. Most noise introduced into image processing for micro UAV applications occurs during transmission to the ground station. Transmitting video that will be processed in order to modify UAV commands introduces a latency that can limit functionality and affect the effectiveness of certain algorithms. Implementing an onboard solution obviously removes the negative attributes of transmitting video for ground-based image processing.

First, work will be cited to support the choice of the Harris Feature Detection algorithm [2], as also described in [6], as the feature detection algorithm for this implementation. As a micro UAV moves across a scene, its motion can be estimated using an affine model. [4] Therefore, it is important that the selected feature detector can find the same features in a scene as the scene is translated, rotated, or scaled with reference to the UAV. Although an onboard image processing solution has considerably less noise than a majority of current micro UAV solutions which transmit video and process information on the ground, the feature detector also needs to be robust to noise that is experienced on micro UAV platforms.

Second, the advantages of using a RANdom SAmple Consensus or RANSAC algorithm to calculate the homography are discussed. The target platform for this implementation will then be presented, after which the details of the algorithm implementations are given. In conclusion, results are discussed and a few optimizations that will be considered for future work are mentioned.

2 Background

Different feature detection algorithms possess varying strengths and weaknesses; therefore, a closer look is required to determine the appropriate algorithm for any given implementation. The following works describe and compare the strengths and weaknesses of a variety of feature detection algorithms.

Schmid, Mohr, and Bauckhage found that the Improved Harris Feature detection algorithm had a better repeatability rate than the Foerstner, Cottier, Heitger, and Horaud algorithms when images were modified by rotation, warping, scaling, lighting, etc.[8] The results of their work show that the Harris Feature detector algorithm is robust even when these changes to the images are severe, except in the case of scaling. In [4], Johansen discusses feature detection algorithms that are better for micro-UAV applications. He found that Harris Feature detection worked better than Canny Edge detection, Foerstner interest operator, Binary Corner detector, and gradient differencing using a Sobel kernel. In another study of feature detection algorithms, Tissainayagama and Suter also found that Harris Feature and KLT feature tracking algorithms performed better

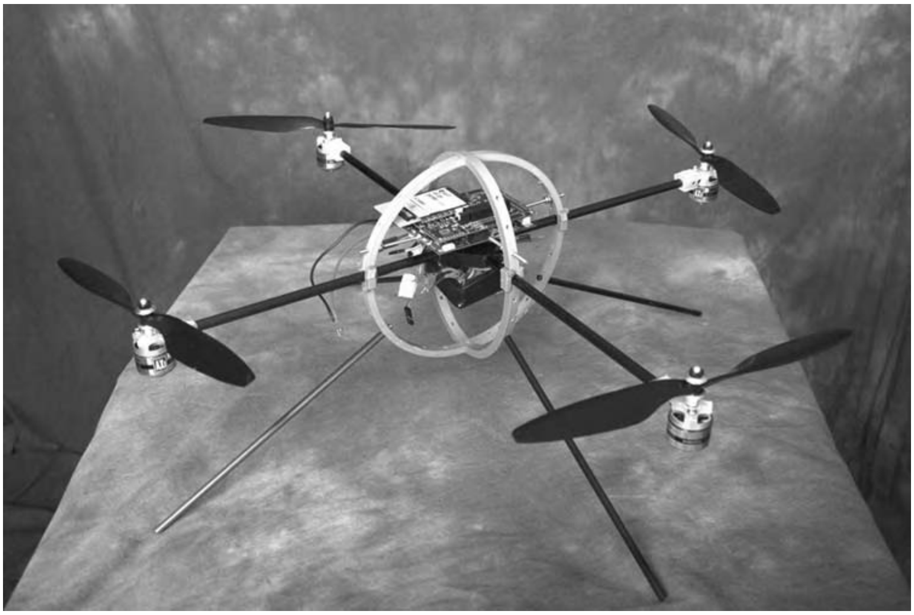


Fig. 1. Quadrotor helicopter platform

than Kitchen-Roselfield and Smith corner detection both in general performance measurements and performance under noisy conditions.[10]

Hartley and Zisserman in [3] showed that performing the RANSAC algorithm for the appropriate number of samples could guarantee a 99% probability of removing outliers. Perez and Garcia also show excellent results from a feature-based algorithm using RANSAC to estimate an homography in their efforts to mosaic images [7]. In [9], Thunuguntla and Gunturk show that a feature-based implementation of the RANSAC algorithm is robust for finding accurate homographies under rotation and scaling changes even in the presence of moving objects in the scene. Work has also been done using the Harris Feature Detector and RANSAC algorithms together to generate accurate homographies, as seen in [5]. Their results show that these algorithms are robust even when large disparities between images exist.

3 Target Platform and Applications

This feature detection and movement tracking algorithm implementation is targeted towards micro UAV platforms. It has specifically been tested on a quadrotor helicopter platform. This platform is of dimensions 30 x 30 x 9 tall. It carries a custom board that includes among other things, a Xilinx FX20 FPGA, SDRAM, USB interface, serial ports, and camera headers. This board communicates with a ground station wirelessly, and to a Kestral autopilot [1] on the quadrotor that handles low level control of the aircraft. This setup allows vision algorithms to be implemented on the FPGA and resulting information to be passed on to the autopilot to modify flight commands. Overall mission objectives may be transmitted from a ground station and status information of the micro UAV is sent back. For the quadrotor application used in this implementation, the results from feature tracking are used to detect motion. A camera located on the underside of the platform and pointed at the ground is used to track features on the ground and detect movement of the micro UAV. The autopilot on the micro UAV is equipped with an inertial measurement unit (IMU) to detect changes in pitch, roll, and yaw. Although this IMU is quite sensitive, it lacks the resolution to be able to detect horizontal drift that is caused by slight differences in motors and propellers or by outside forces such as wind gusts. The feature tracking system can detect this movement and send appropriate commands to the rotors in order to correct and maintain a stable hovering position. Figure 1 shows the platform used to test the implemented feature detection and homography algorithms.

4 Algorithm Implementations

The feature detection and homography algorithms were divided into hardware cores and software functions. The hardware cores that were implemented were an xy gradient, a Harris Feature detector, and a priority queue. The input to the xy gradient core is a 3x3 pixel matrix from a row buffer fed by the camera, allowing a pipelined processing of images. The gradient core outputs two one-pixel gradient values for each 3 pixel by 3 pixel input matrix (one value for the x gradient and another for the y gradient), which is then buffered for three rows of pixels in preparation for the Harris feature detection core. The Harris core calculates feature strengths and matches strengths with an x and y value representing the pixel location in the image. The feature strength, x, and y indices are concatenated to create a single 32 bit value.

The feature priority queue operates on the 32 bit values, prioritizing them based on feature strength. The current implementation uses a queue that contains a sorted list of twenty features. When the processor receives the interrupt signifying that the frame is complete, then it dequeues the queue, writing the values to memory. The processor then reads the lists of two images and pairs features from each of them. RANSAC is run on the resulting paired list, and a homography is calculated. A block diagram of this design can be seen in Figure 2.

In order to perform Harris feature detection in hardware a few simplifications were made to the original algorithm [2], such as scaling down values to ensure that no overflow occurred. In this implementation the top 20 features were identified in the image which was accomplished by using a priority queue. As each feature strength and indices pair is produced by the Harris Feature core it is compared with existing pairs in the queue. The new pair is inserted in the appropriate location in the queue only if there are no pairs with a higher feature strength already in the queue that is within a specified minimum distance of the new pair. This process in the priority queue is accomplished in hardware by making two passes through the pairs in the queue. The first pass compares the

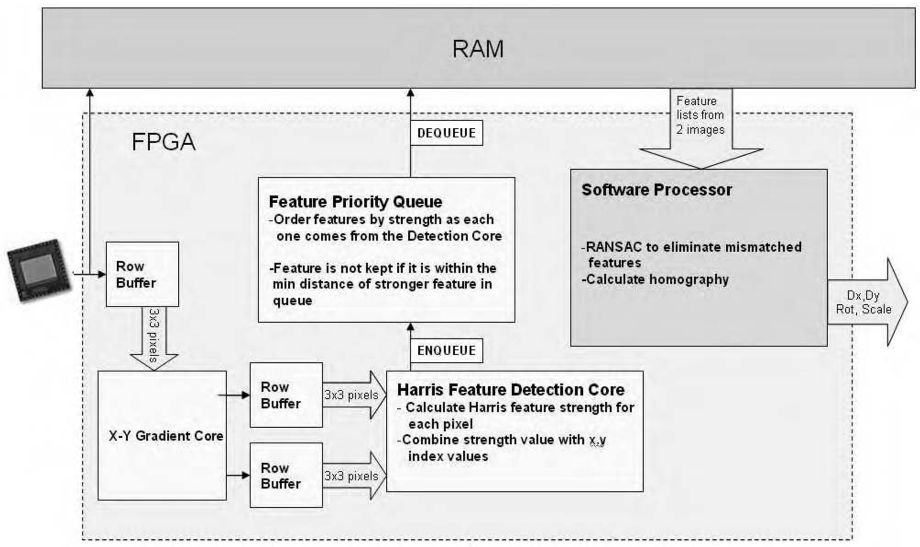

RAM
FPGA
DEQUEUE
Feature lists from 2 images
Feature Priority Queue
-Order features by strength as each one comes from the Detection Core
-Feature is not kept if it is within the min distance of stronger feature in queue
Software Processor
-RANSAC to eliminate mismatched features
-Calculate homography
Dx,Dy Rot, Scale
Row Buffer
3x3 pixels
ENQUEUE
X-Y Gradient Core
Row Buffer
3x3 pixels
Row Buffer
3x3 pixels
Harris Feature Detection Core
- Calculate Harris feature strength for each pixel
-Combine strength value with x,y index values

Fig. 2. Block diagram of hardware/software co-design

x, y indices of the new pair with the x, y indices of the pairs in the queue using a Manhattan distance. When a pair is encountered that is within the minimum distance then the pair with the lower strength value is set to zero. The second pass compares feature strengths allowing the new feature strength index pair to be inserted into the queue if necessary and the end of the queue to shift down.

Limiting the selection of features using a minimum distance minimizes the grouping of features and provides a simple matching scheme across sequential images. Using features that are spread out across the image decreases the probability of a moving object in the scene to negatively affect the homography that is calculated. If a minimum distance threshold is selected that corresponds to the maximum movement that is expected from one frame to the next, then the matching scheme is as simple as correlating the two features that are within the minimum distance of each other. The maximum lateral velocity of the quadrotor helicopter platform and the distance it maintains from the scene under standard operation allowed a minimum distance to be selected for this implementation.

Once good features have been found and correlated in two images, a RANSAC homography algorithm is used to determine the translation, rotation, and scaling difference between the images. A least squares approach to calculating homographies described in [4] is used to compare against the results of the RANSAC algorithm for a performance measure, and the results are discussed in section 5. The homography equations used in the RANSAC algorithm are derived from the affine model shown in Eq 1 and matrix equations 2, 3, 4 and 5, where (x_1, y_1) and (x_2, y_2) are two feature indices and $(x_1^{'}, y_1^{'})$ and $(x_2^{'}, y_2^{'})$ are the matching feature indices in a previous image.

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{1}$$

$$Ax = b \tag{2}$$

$$A = \begin{bmatrix} x_1' & -y_1' & 1 & 0 \\ y_1' & x_1' & 0 & 1 \\ x_2' & -y_2' & 1 & 0 \\ y_2' & x_2' & 0 & 1 \end{bmatrix} \tag{3}$$

$$x = \begin{bmatrix} \alpha & \beta & \gamma & \xi \end{bmatrix}^T \tag{4}$$

$$b = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 \end{bmatrix}^T \tag{5}$$

Solving for α and β results in Equations 6 and 7 where $det(A)$ is shown in 8. Equations 8 and 12 can then be used to find the angle of rotation θ, and the scale factor λ between the two images.

$$\alpha = \frac{x_1(x_1^{'} - x_2^{'}) + y_1(y_1^{'} - y_2^{'}) + x_2(x_2^{'} - y_1^{'}) + y_2(y_2^{'} - y_1^{'})}{det(A)} \tag{6}$$

$$\beta = \frac{x_1(y_2^{'} - y_1^{'}) + y_1(x_1^{'} - x_2^{'}) + x_2(y_1^{'} - y_2^{'}) + y_2(x_2^{'} - x_1^{'})}{det(A)} \tag{7}$$

$$det(A) = x_1^{'2} + y_1^{'2} + x_2^{'2} + y_2^{'2} - 2y_1^{'}y_2^{'} - 2x_1^{'}x_2^{'} \tag{8}$$

$$\theta = -\arctan\left(\frac{\beta}{\alpha}\right) \tag{9}$$

$$\lambda = \frac{\cos(\theta)}{a} \tag{10}$$

Translation along the two perpendicular axes, T_x and T_y, are calculated using Equations 11 and 12, which are derived by solving for γ and ξ in Equation 4. Variables δ, ζ, φ, and ψ are defined in Equations 13, 14, 15, 16, respectively.

$$T_x = \frac{\delta x_1 + \zeta y_1 + \varphi x_2 + \psi y_2}{det(A)} \tag{11}$$

$$T_y = \frac{-\zeta x_1 + \delta y_1 + \psi x_2 + \varphi y_2}{det(A)} \tag{12}$$

$$\delta = x_2^{'2} + y_2^{'2} - y_1^{'}y_2^{'} - x_1^{'}x_2^{'} \tag{13}$$

$$\zeta = y_2^{'}x_1^{'} - y_1^{'}x_2^{'} \tag{14}$$

$$\varphi = x_1^{'2} + y_1^{'2} - y_1^{'}y_2^{'} - x_1^{'}x_2^{'} \tag{15}$$

$$\psi = y_1^{'}x_2^{'} - y_2^{'}x_1^{'} \tag{16}$$

Just like a traditional RANSAC algorithm, random sets of feature indices are selected and the homography results that are calculated from them are compared with the first set using a Euclidean distance measure. A variation made in the algorithm for this implementation is that instead of using the whole set of inliers to calculate the result, only one pair is chosen. Of the set of inliers that are produced from the RANSAC voting, the one pair of points that has the most votes is selected as the pair that most closely represents the whole set of inliers. The homography generated by this pair is used for the helicopter command. This variation reduces the amount of computation needed to generate a homography by reducing the overconstrained problem of multiple points to a problem with an exact solution. The next section discusses the results of the two methods.

5 Results

After completing the implementation of the Harris feature detector hardware and the RANSAC homography software, multiple tests were performed to ascertain performance results. Figures 3(a) and 3(b) show a typical image scene transmitted to a desktop computer from the platform where boxes representing the minimum distance around the resulting features are drawn on the image. Figure 3(a) is the original image and Figure 3(b) is the top eight bits of the Harris Feature strengths of the image.

Comparing the sample scene and the resulting feature strength image generated by the Harris Feature Detection core shows that feature strengths are calculated as would be expected. With the current resources on the Xilinx FX20

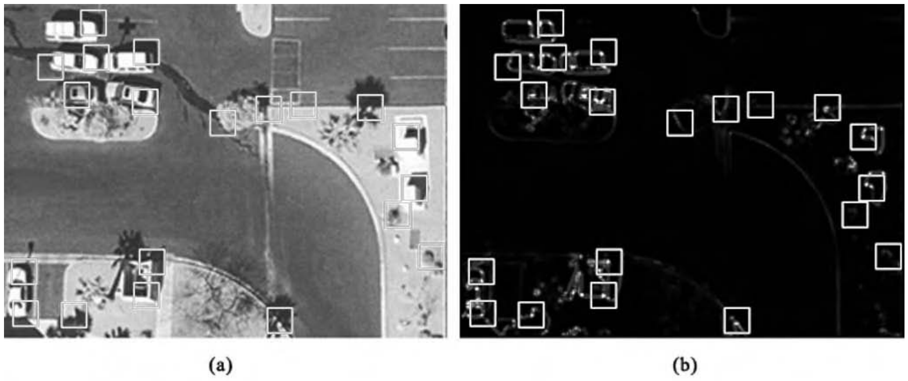


Fig. 3. (a) Typical scene for micro UAVs with minimum distance boxes centered on features. (b) Feature strengths of same scene.

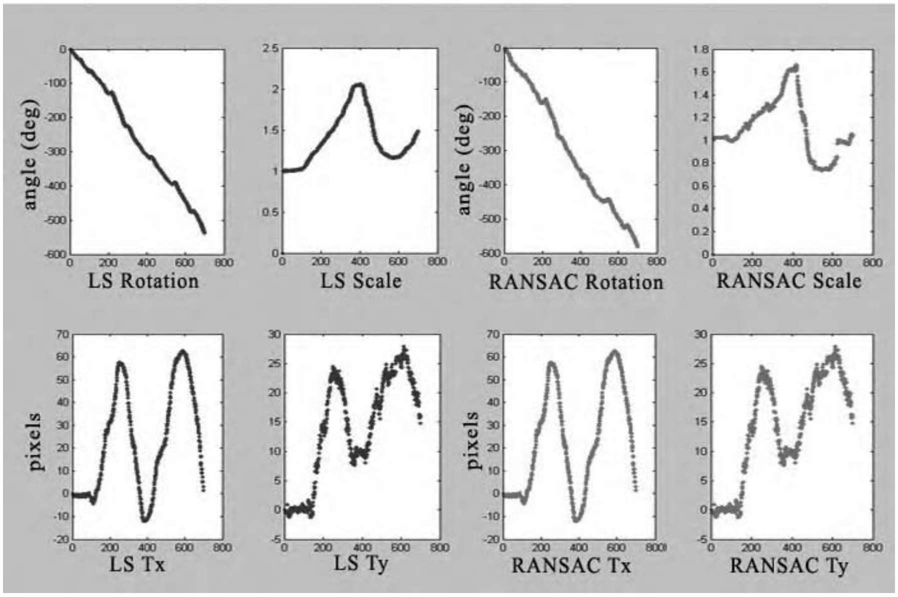


Fig. 4. Right four graphs are results of rotation, scale, x translation, and y translation tests using least squares solution on all matched features. Left four graphs are results of same four tests using modified RANSAC.

FPGA, the feature detector and priority queue detect and correspond the 20 strongest features at 30 fps. Given the design of this implementation it would be possible to track more features, the only limitation being the time to sort them in the priority queue. The minimum distance correspondence scheme gave excellent results. It was found that features were mostly mismatched in just one case where two features that were within the selected minimum distance of each other in a scene produced very similar Harris Feature strengths. Sometimes these features could vary in strength enough to switch their order, changing which one stayed in the queue. If one is selected in the first frame and the other in the second frame then an incorrect homography would be found between the two frames. On average this occurred with between 5-15% of the features on the various scenes and lighting conditions tested. As discussed, performing RANSAC on the resulting matched pairs for the appropriate number of iterations would allow the correctly matched features to outvote these outlier pairs in calculating the homography.

A sequence of images and resulting feature points were captured of controlled scene movements and evaluated in MATLAB. The homography calculations for each pair of images were accumulated in order to compare estimated scene movement over the sequence with the actual movement. Rotation angle, scale factor, and translation along two axis were calculated using all points by the least squares approach and graphed in the left four graphs of Figure 4. In this figure the graphs for four different tests are shown, the first compares rotation results when the scene was just rotated clockwise, the second scaling results when the scene was moved towards the camera, away, and then back towards the camera. The third and fourth tests consisted of translation back and forth, and up and down along the x and y axes, respectively. The four graphs on the right of Figure 4 show the results for the same test generated by using just two sets of points selected by the RANSAC algorithm. As can be seen the accumulated error is very small throughout most of the tests. Since only relative information will used in the control of the helicopter platform then this accumulated error is of little concern, and the RANSAC results are considered equivalent to a least squares solution using all the features.

Although equipment was not available to obtain an exact measurement of actual movement of the scene for the tests performed, estimates showed that the homographies were quite accurate. For example, the actual total rotation of the scene was approximately 610 degrees, whereas Figure 4 shows that the calculated total rotation added up to about 550 degrees, which is less than a 10% error over 800 frames.

Throughout testing it was observed that inaccuracies in the homography were evident when too few features existed in the scene. Also, the accuracy of a homography is obviously dependent on the resolution of the camera used since the homography is calculated using pixel indices.

6 Future Enhancements

The platform used in this implementation did not require a homography to be calculated for every frame, but optimizations could be made to the homography calculation code to ensure that it met 30 fps speed. The current code uses the arctangent and cosine functions in the standard C++ math library. Since homographies are generated every couple of frames the range of angles of rotation that will be observed with a quadrotor helicopter platform is very small. If the performance velocity to calculate a homography after every frame is increased then the range would be even smaller. This would make calculation of trigonometric functions using lookup tables more feasible than using the standard math library trigonometric functions. Small lookup tables would not require much more space in memory, but would increase the speed to only a couple of cycles.

Other work to be done is to use the same feature tracking system with a forward-facing camera mounted on the UAV to detect and track objects of interest. By using the detected changes in position and scaling of the object of interest, the helicopter will be able to maintain a specified distance from the object.

Acknowledgment

This work was supported in part by David and Deborah Huber.

References

1. Beard, R., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., Goodrich, M.: Autonomous vehicle technologies for small fixed wing uavs. AIAA Journal of Aerospace Computing, Information, and Communication 2(1), 92–108 (2005)
2. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference, vol. 15 (1988)
3. Hartley, R., Zisserman, A.: Multiple view geometry in computer visiond. Cambridge University Press, New York, NY, USA (2000)
4. David, L.: Johansen. Video Stabilization and Object Localization Using Feature Tracking with Small UAV Video. PhD thesis, Brigham Young University (2006)
5. Kanazawa, Y., Kanatani, K.: Robust image matching under a large disparity. Workshop on Science of Computer Vision, 46–52 (2002)
6. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An Invitation to 3-D Vision: From Images to Geometric Models. In: Interdisciplinary Applied Mathematics, 1st edn., vol. 26, Springer, Heidelberg (2004)
7. Perez, P., Garcia, N.: Robust and accurate registration of images with unknown relative orientation and exposure. In: Image Processing, 2005. ICIP 2005. IEEE International Conference on, vol. 3, pp. 1104–1107. IEEE Computer Society Press, Los Alamitos (2005)
8. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. International Journal of Computer Vision 37(2), 151–172 (2000)
9. Thunuguntla, S.S., Gunturk, B.K.: Feature-based image registration in log-polar domain. In: ICASSP 2005. Proceedings IEEE International Conference, Acoustics, Speech, and Signal Processing, March 18-23, 2005, vol. 2, pp. 853–856. IEEE Computer Society Press, Los Alamitos (2005)
10. Tissainayagam, P., Suter, D.: Assessing the performance of corner detectors for point feature tracking applications. Image and Vision Computing 22(8), 663–679 (2004)

Utilizing Semantic Interpretation of Junctions for 3D-2D Pose Estimation

Florian Pilz[1], Yan Shi[1], Daniel Grest[1], Nicolas Pugeault[2], Sinan Kalkan[3], and Norbert Krüger[4]

[1] Medialogy Lab, Aalborg University Copenhagen, Denmark
[2] School of Informatics, University of Edinburgh, United Kingdom
[3] Bernstein Center for Computational Neuroscience, University of Göttingen, Germany
[4] Cognitive Vision Group, University of Southern Denmark, Denmark

Abstract. In this paper we investigate the quality of 3D-2D pose estimates using hand labeled line and point correspondences. We select point correspondences from junctions in the image, allowing to construct a meaningful interpretation about how the junction is formed, as proposed in e.g. [1], [2], [3]. We make us of this information referred as the semantic interpretation, to identify the different types of junctions (i.e. L-junctions and T-junctions). T-junctions often denote occluding contour, and thus do not designate a point in space. We show that the semantic interpretations is useful for the removal of these T-junction from correspondence sets, since they have a negative effect on motion estimates. Furthermore, we demonstrate the possibility to derive additional line correspondences from junctions using the semantic interpretation, providing more constraints and thereby more robust estimates.

1 Introduction

The knowledge about the motion of objects in a scene is crucial for many applications such as driver assistant systems, object recognition, collision avoidance and motion capture in animation. One important class of motion is the 'Rigid Body Motion' (RBM) which is defined as a continuous movement of the object, such that the distance between any two points of the object remains fixed at all times. The mathematical structure of this motion has been studied for a long while (see e.g., [4]). However, the problem of visual based motion estimation is far from being solved. Different methods for RBM estimation have been proposed [5] and can be separated into feature based, optic flow based and direct methods, where this work concerns a feature based method. In feature based RBM estimation, image features (e.g., junctions [6] or lines [7]) are extracted and their are correspondences defined. The process of extracting and matching correspondences suffers from high ambiguity, and is even more severe than the correspondence problem for stereopsis, since the epipolar constraint is not directly applicable. A Rigid Body Motion consisting of translation $\boldsymbol{t}$ and rotation $\boldsymbol{r}$ is described by six parameters, three for the translation $\boldsymbol{t} = (t_1, t_2, t_3)$ and three for rotation

$\boldsymbol{r} = (r_1, r_2, r_3)$. This allows for the formulation of the transformation between a visual entity in one frame, and the same entity in the next frame.

$$RBM^{(\boldsymbol{t},\boldsymbol{r})}(e) = e' \tag{1}$$

The problem of computing the RBM from correspondences between 3D objects and 2D image entities is referred as 3D-2D pose estimation [8,9]. The 3D entity (3D object information) needs to be associated to a 2D entity (2D knowledge of the same object in the next image) by the perspective projection P.

$$P(RBM^{(\boldsymbol{t},\boldsymbol{r})}(e)) = e' \tag{2}$$

There exist approaches (in the following called projective approaches) that formalize constraints directly on equation (2) (see e.g., [10]). An alternative is, instead of formalizing the pose estimation problem in the image plane, to associate a 3D entity to each 2D entity: A 2D image point together with the optical center of the camera spans a 3D line (see figure 1a) and an image line together with the optical center generates a 3D plane (see figure 1b). In case of a 2D point p we denote the 3D line that is generated in this way by $\boldsymbol{L}(p)$. Now the RBM estimation problem can be formulated for 3D entities

$$RBM^{(\boldsymbol{t},\boldsymbol{r})}(\boldsymbol{p}) \in \boldsymbol{L}(p) \tag{3}$$

where $\boldsymbol{p}$ is the 3D Point. Such a formulation in 3D has been applied by, e.g., [11,9], coding the RBM estimation problem in a twist representation that can be computed iteratively on a linearized approximation of the RBM.

In this paper we study how multiple correspondence types of visual entities influence RBM estimation performance. We identify T-junctions by making use of the semantic interpretation [3] and study their effect on RBM estimation performance. Furthermore we make use of the semantic interpretation to derive additional line correspondences for junctions.

The paper is structured as following: In section 2, we describe the formulation of the constraint equations that are used in the 3D-2D pose estimation algorithm. In section 3, we introduce the scenario in which our technique is applied. In section 4 we describe the concept of a semantic interpretation for junction and explain in more detail how this applies to the 3D-2D pose estimation problem.

2 Constraint Equations

We now want to formulate constraints between 2D image entities and 3D object entities, where a 2D image point together with the optical center of the camera spans a 3D-line (see fig. 1a) and an image line together with the optical center generates a 3D-plane (see fig. 1b).

A 3D line. $\mathbf{L}$ can be expressed as two 3D vectors $\mathbf{r}, \mathbf{m}$. The vector $\mathbf{r}$ describes the direction and $\mathbf{m}$ describes the moment which is the cross product of a point

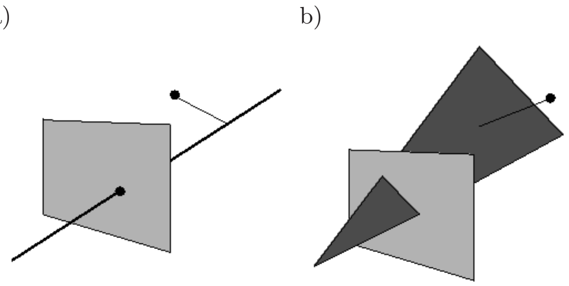


Fig. 1. Geometric Interpretation of constraint equations[12], a) Knowing the camera geometry a 3D-line can be generated from an image point and the optical center of the camera. The 3D- -point-3D-line constraint realizes the shortest Euclidian distance between the 3D-point and the 3D-line. b) From an image line a 3D-plane can be generated. The 3D-point-3D-plane constraint realizes the shortest Euclidian distance between the 3D-point and the 3D-plane.

$\mathbf{p}$ on the line and the direction $\mathbf{m} = \mathbf{p} \times \mathbf{r}$. The vectors $\mathbf{r}$ and $\mathbf{m}$ are called Plücker coordinates. The null space of the equation $\mathbf{x} \times \mathbf{r} - \mathbf{m} = \mathbf{0}$ is the set of all points on the line, and can be expressed in matrix form (see eq. 4). The creation of the 3D-line $\mathbf{L}$ from the 2D-point p together with the 3D-point $\mathbf{p}$ allows the formulation of the 3D-point-3D-line constraint [13] (eq.5),

$$\mathbf{F}^{\mathbf{L}}(\mathbf{x}) = \begin{pmatrix} 0 & r_x & -r_y & -m_x \\ -r_z & 0 & r_x & -m_y \\ r_y & -r_x & 0 & -m_z \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4}$$

$$\mathbf{F}^{\mathbf{L}(p)} \left((I_{4\times 4} + \alpha\tilde{\xi})\mathbf{p} \right) = 0. \tag{5}$$

where $\tilde{\xi}$ is the matrix representing the linearisation of the RBM and α is a scale value 7 [12]. Note, that the value $||\mathbf{F}^L(\mathbf{x})||$ can be interpreted as the Euclidian distance between the point (x_1, x_2, x_3) and the closest point on the line to (x_1, x_2, x_3) [14,9]. Note that although we have 3 equations for one correspondence the matrix is of rank 2 resulting in 2 constraints.

A 3D-plane. $\mathbf{P}$ can be expressed by defining the components of the unit normal vector $\boldsymbol{n}$ and a scalar (Hesse distance) δ_h. The null space of the equation $\boldsymbol{n} \cdot \mathbf{x} - \delta_h = \mathbf{0}$ is the set of all points on the plane, and can be expressed in matrix form (see eq. 6). The creation of the 3D-plane $\mathbf{P}$ from the 2D-line l together with the 3D-point $\mathbf{p}$ allows the formulation of the 3D-point-3D-plane constraint (eq.7) [13].

$$F^{\mathbf{P}}(\mathbf{x}) = \begin{pmatrix} n_1 & n_2 & n_3 & -\delta_h \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = 0 \tag{6}$$

$$F^{\mathbf{P}(p)}\left((I_{4\times 4} + \alpha\tilde{\xi})\mathbf{p}\right) = 0. \tag{7}$$

Note that the value $||F^P(\mathbf{x})||$ can be interpreted as the Euclidian distance between the point (x_1, x_2, x_3) and the closest point on the plane to (x_1, x_2, x_3) [14,9]. These 3D-point-3D-line and 3D-point-3D-plane constraints result in a system of linear equations, which solution becomes optimized iteratively (for details see [12]).

3 Ego-motion Estimation from Stereo Image Sequences

We apply the pose estimation algorithm for estimating the motion in stereo sequences. Here we do not have any model knowledge about the scene. Therefore the 3D entities need to be computed from stereo correspondences. We provide hand picked 2D-point and line correspondences in two consecutive stereo frames. From stereo correspondences (fig.2) we compute a 3D-point. The corresponding 2D-point or line in the next left frame (fig.3 a-f) provides either a 3D-line or 3D-plane. For each, one constraint can be derived (see eq.(5 and (7).

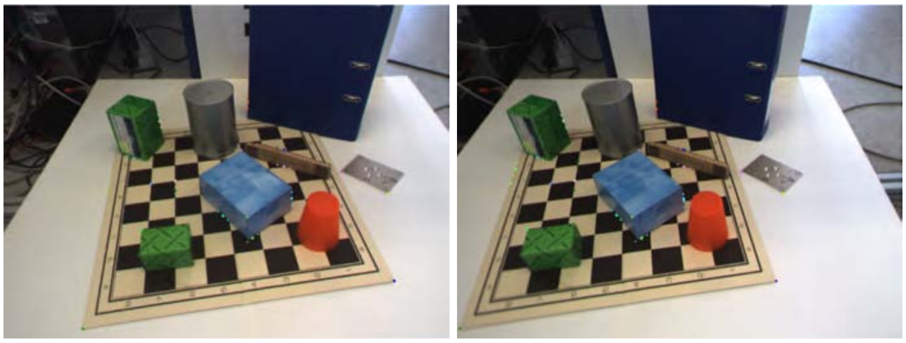


Fig. 2. Stereo image pair (left and right) used for computation of 3D entities

A 3D-point-2D-line correspondence leads to one independent constraint equation and a point-point (3D-point-2D-point) correspondence leads to two independent constraint equations [12]. Therefore, it is expected that 3D-point-2D-point correspondences produce more accurate estimates with smaller sets than 3D-point-2D-line correspondences [15].

4 Using Semantic Interpretation of Junctions

A junction is represented by a set of rays corresponding to the lines that intersect, each defined by an orientation [1] (see fig. 4). In [3] a more elaborate description can be found, where methods for junction detection and extraction

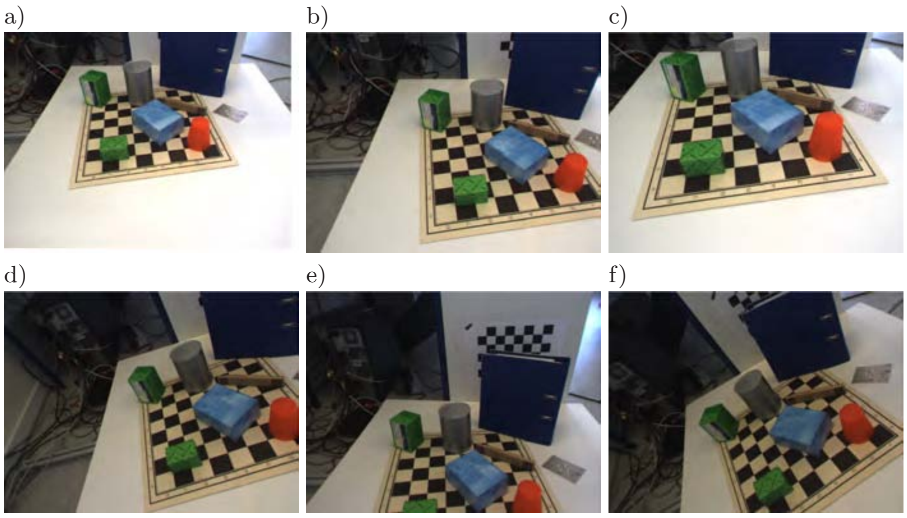


Fig. 3. Image Set: a) translation on x-axis (150 mm), b) translation on y-axis (150 mm), c) translation on z axis (150mm), d) rotation around x-axis (+20 deg), e) rotation around y-axis (-15 deg), f) rotation around z-axis (+30 deg)

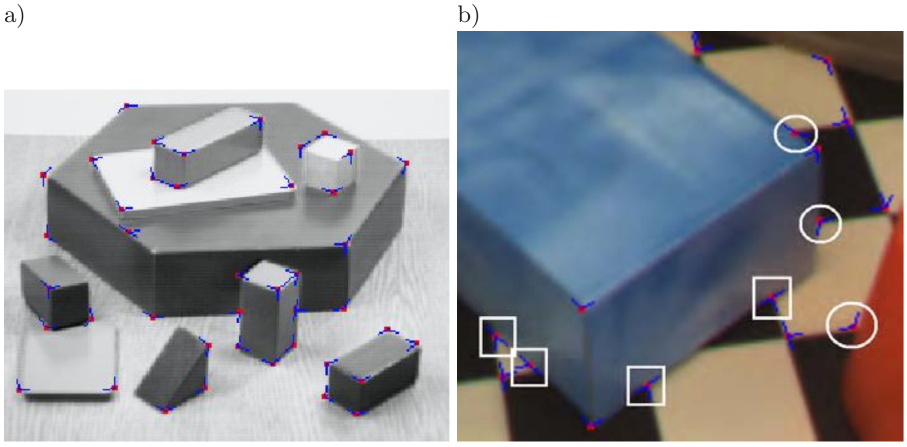
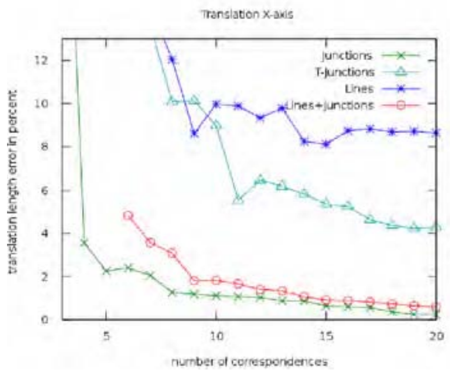
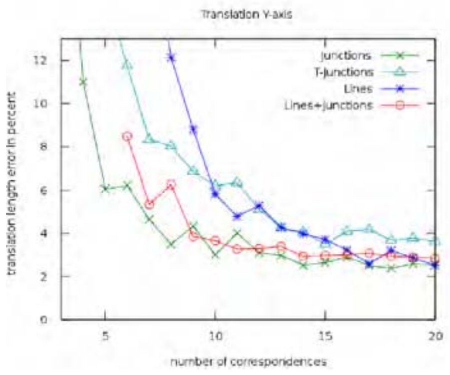


Fig. 4. a) Detected junctions with extracted semantic interpretation [3]. b) Examples of T-junctions (partly detected) - rectangles: valid T-junctions, circles: invalid T-junctions due depth discontinuity.

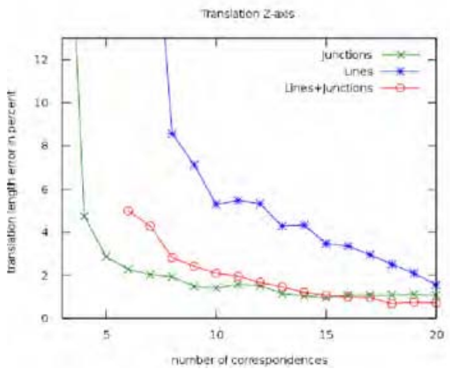
their semantic information are proposed. Since 3D-point-2D-point correspondences are junctions often, the additional semantic information can be extracted and utilized in the context of 3D-2D pose estimation. Feature-based motion estimation makes assumptions on the visual entities used as correspondences. One kind of uncertainty arises through the assumption, that correspondences a)



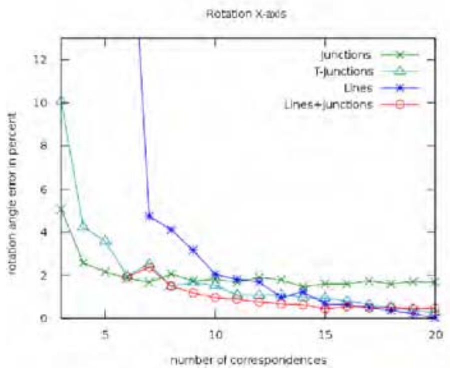
b)



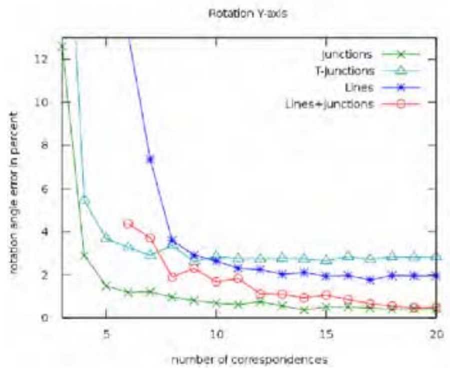
c)*



d)



e)



f)

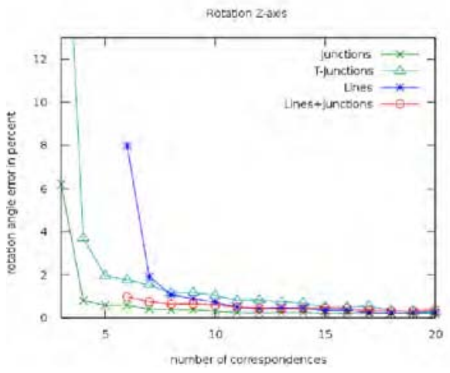
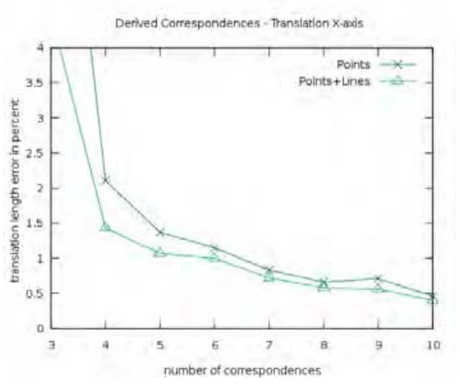
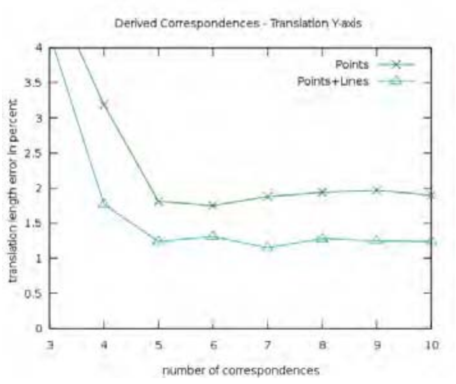


Fig. 5. translation length error in percent: a) translation on x-axis (150 mm), b) translation on y-axis (150 mm), c) translation on z axis (150mm), * here the error for T-junctions is too high to be shown. d) rotation around x-axis (+20 deg), e) rotation around y-axis (-15 deg),f) rotation around z-axis (+30 deg).

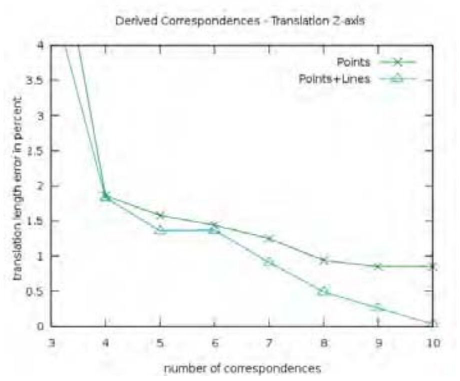
a)



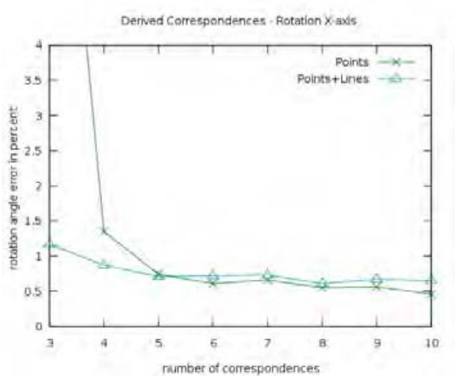
b)



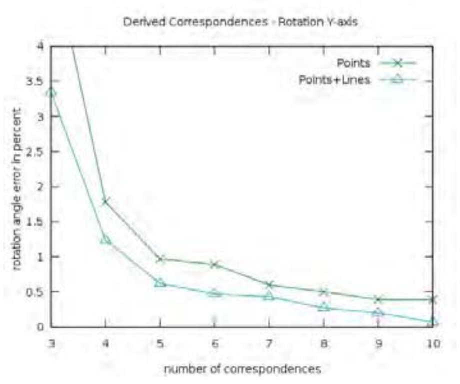
c)



d)



e)



f)

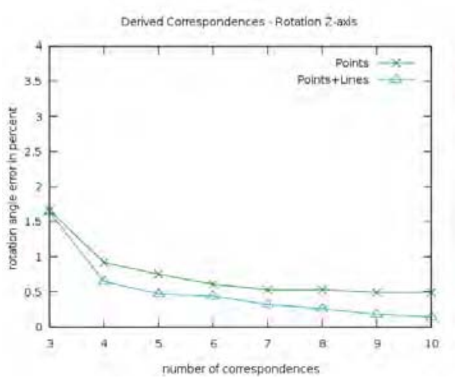


Fig. 6. translation length error in percent for derived correspondence: a) translation on x-axis (150 mm), b) translation on y-axis (150 mm), c) translation on z axis (150mm), d) rotation around x-axis (+20 deg), e) rotation around y-axis (-15 deg),f) rotation around z-axis (+30 deg)

are temporally stable. Certain kinds of visual entities such as T-junctions (see Fig.4b) may be an indication of depth discontinuity, having a negative effect on motion estimates. The use of the semantic interpretation allows us to identify and remove constraints based on these T-junction correspondences. Temporally unstable correspondences introduce error in two ways. First, the computed 3D point is errorenous and second the corresponding 2D point does not reflect the motion between to consecutive frames. Furthermore, if a semantic interpretation of a junction is known, further constraints can be derived for one location. The idea is to use the information of the lines to build up additional 3D-point 2D-line correspondences. In the case of point and line correspondences, it allows deriving line correspondences from existing point correspondences (e.g., for an L-junction this leads to one point and two line correspondences). For testing, we handpick 10 point correspondences (see figure 2 and 3a-f), estimate the pose and compare it with the estimation based one the very same 10 point correspondences and their intersecting edges.

5 Results

The images are recorded with a calibrated camera system mounted on a robotic arm. The error is computed as the difference between the estimated translation length and rotation angle compared to the ground, truth provided by the robot. The data set consist of a set of stereo images (Fig.2) at different time steps and six images for different arm motion (Fig.3a-f). The process of manually picking correspondences introduces a small position error, where lines and points are not picked with the same accuracy. To reduce this error difference between line and point correspondences, we add noise ($n \in [0, 1]$) to the 2D correspondences pixel positions. In Fig. 5 and Fig. 6 the error between estimated motion and ground truth is shown for different types of correspondences. The experiments were repeated 50 times for random subsets of correspondences with with a specific set size as shown on the x-axis. Results show that the error for 2D-point (junction) correspondences decreases faster than for 2D-line (edge) correspondences (see fig. 5a-f). Moreover it shows that more than 8 point or 15 line correspondence do not further minimize the error. Motion estimates from correspondence sets based on a combination of junctions and edges converge to nearly the same error than junction correspondence sets (see Fig.5a-f). In cases where lines perform better (see Fig. 5d), the error for mixed correspondence sets converges to a value similar to the best of both individual correspondence sets. Moreover, estimations are independent of motion direction or length and provide even for minimal solution sets accurate estimates.

We can make the general observation that correspondences from T-junctions have a negative influence on motion estimates. For translational motions the error is related to the motion direction. T-junctions introduce a noticeably error for forward/backward motions (x-axis) (Fig. 5a), while the error for sideways motions (y-axis) is insignificant (Fig. 5b). For motions along the z-axis the error introduced by T-junctions is enormous (around 250%) and therefore not displayed in figure 5c. For rotational motions the error introduced by T-junctions in this scenario is minimal (see fig. 5d-f).

If a semantic interpretation of junctions is available, further correspondences can be generated for these junctions. This provides more constraints and should therefore result in better RBM estimates. For a given L/Y junction the semantic interpretation provides the information of the intersecting lines. Each line is defined by its orientation and junctions position. In order to derive an additional line correspondence a point is constructed on the lines in some distance from the junction. Figure 6a-f shows that using the intersecting lines of a junction indeed results in more accurate estimates.

6 Discussion

The evaluation of line and point correspondences has shown, that 2D-point correspondences provide more valuable constraints than 2D-line, since they lead to more constraints. The results show that T-junctions may introduce errors and should therefore be avoided as point correspondences for motion estimation. The semantic interpretation is a way to identify and disregard these temporally inconsistent image entities, providing correspondence sets leading to more accurate and robust RBM estimations. It is questionnable whether line contraints derived from junctions provide additional information compared to those junctions. However, in the presence of noise, it is expected these additional constraints further reduce the estimation error. The results in Fig. 6 clearly confirm this expectation.

To summarize, the contribution shows that a combination of 2D-Points and 2D-Lines correspondences result in more accurate and robust motion estimations. The semantic interpretation of junctions (2D-points) allows us to disregard T-junctions and to derive additional line correspondences from junctions, providing more robust correspondences sets.

References

1. Parida, L., Geiger, D., Hummel, R.: Junctions: detection, classification and reconstruction. In: CVPR 1998. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 20, pp. 687–698 (1998)
2. Rohr, K.: Recognizing corners by fitting parametric models. International Journal of Computer Vision 9, 213–230 (1992)
3. Kalkan, S., Yan, S., Pilz, F., Krüger, N.: Improving junction detection by semantic interpretation. In: VISAPP 2007. International Conference on Computer Vision Theory and Applications (2007)
4. Ball, R.: The theory of screws. Cambridge University Press, Cambridge (1900)
5. Steinbach, E.: Data driven 3-D Rigid Body Motion and Structure Estimation. Shaker Verlag (2000)
6. Phong, T., Horaud, R., Yassine, A., Tao, P.: Object pose from 2-D to 3-D point and line correspondences. International Journal of Computer Vision 15, 225–243 (1995)

7. Krüger, N., Jäger, T., Perwass, C.: Extraction of object representations from stereo imagesequences utilizing statistical and deterministic regularities in visual data. In: DAGM Workshop on Cognitive Vision, pp. 92–100 (2002)
8. Grimson, W. (ed.): Object Recognition by Computer. MIT Press, Cambridge (1990)
9. Rosenhahn, B., Sommer, G.: Adaptive pose estimation for different corresponding entities. In: Van Gool, L. (ed.) Pattern Recognition. LNCS, vol. 2449, pp. 265–273. Springer, Heidelberg (2002)
10. Araujo, H., Carceroni, R., Brown, C.: A fully projective formulation to improve the accuracy of lowe's pose–estimation algorithm. Computer Vision and Image Understanding 70, 227–238 (1998)
11. Rosenhahn, B., Perwass, C., Sommer, G.: Cvonline: Foundations about 2d-3d pose estimation. In: Fisher, R. (ed.) CVonline: On-Line Compendium of Computer Vision [Online] (2004), `http://homepages.inf.ed.ac.uk/rbf/CVonline/`
12. Krüger, N., Wörgötter, F.: Statistical and deterministic regularities: Utilisation of motion and grouping in biological and artificial visual systems. Advances in Imaging and Electron Physics 131, 82–147 (2004)
13. Wettegren, B., Christensen, L., Rosenhahn, B., Granert, O., Krüger, N.: Image uncertainty and pose estimation in 3d euclidian space. In: DSAGM 2005. Proceedings DSAGM 13th Danish Conference on Pattern Recognition and Image Analysis, pp. 76–84 (2005)
14. J.M., S.: Some remarks on the statistics of pose estimation. Technical Report SBU-CISM-00-25, South Bank University, London (2000)
15. Liu, Y., Huang, T., Faugeras, O.: Determination of camera location from 2-d to 3-d line and point correspondence. In: CVPR 1989. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 12, pp. 28–37 (1989)

Shape from Texture of Developable Surfaces Via Fourier Analysis

Fabio Galasso and Joan Lasenby

University of Cambridge, Cambridge, UK

Abstract. Shape from texture has received much attention in the past few decades. We propose a computationally efficient method to extract the 3D shape of developable surfaces from the spectral variations of a visual texture. Under the assumption of homogeneity, the texture is represented by the novel method of identifying ridges of its Fourier transform. Local spatial frequencies are then computed using a minimal set of selected Gabor filters. In both orthographic and perspective projection cases, new geometric equations are presented to compute the shape of developable surfaces from frequencies. The results are validated with semi-synthetic and real pictures.

1 Introduction

Shape from Texture was first introduced by Gibson 50 years ago. In [1] he suggests that texture can provide an important shape cue. However, for a machine the solution to this problem is ill-posed. Shape from texture is generally about measuring the texture distortion in an image, and then reconstructing the surface 3D coordinates in the scene ([2], [3], [4], [5]). The model for the texture can be either deterministic or stochastic. The second allows a wider variety of textures ([4], [5], [6]) and implies local spectral measurements, usually with the Fourier transform ([5]), or more recently, wavelets ([3], [7]).

An initial assumption about the texture is always necessary, and a restrictive one can preclude applicability to real surfaces. [8] deals with texels, which are seldom found in nature, while [9] assumes isotropy, rarely the case. Homogeneity is more frequently used ([4], [2], [7]), and is the one we choose here. For deterministic textures it can be seen as periodicity, for stochastic textures it can be formalized as stationarity under translation ([5]). Under this condition we assume that all texture variations are produced only by projective geometry.

We consider the cases of an orthographic camera model, as in [6] and [10], and of a perspective camera model, as in [11] and [12]. In both cases advantages and disadvantages are presented and results are compared.

The present work takes its motivation from [12]. The texture is analyzed using Gabor filters to produce distortion information based on *local spatial frequency* (LSF). Unlike [12], we do not just rely on a dominant LSF, but we consider groups of LSFs. This extends [12] to exploit the multi-scale nature of textures. To our knowledge the algorithm presented here is the first to consider the multi-scale nature of texture to the extent of using all main LSFs, most of the related work uses only two preferred directions in the spectral domain (e.g. [6],[13]).

Furthermore we present here new geometric equations to compute the 3D coordinates of a developable shape from its LSFs, both in the orthographic and in the perspective case.

Section 2 explains in detail how the texture is analyzed to produce distortion information. Section 3 presents the projective geometry. Section 4 shows how we can recover the 3D coordinates from the measured texture distortion. Finally, section 5 presents results.

2 Texture Description

Here we describe how to set 2D Gabor functions and their first derivatives from the information on texture supplied by the Fourier transform. The former provide local analysis to compute instantaneous frequencies, which are used to measure distortion and reconstruct the 3D coordinates of the texture surface.

2.1 Estimating the Instantaneous Frequencies

We can analyse an image $I(\mathbf{x})$ using a band-pass filter $h(\mathbf{x}, \mathbf{u})$, a function of a point $\mathbf{x}$ and of a central frequency $\mathbf{u}$, which is convolved with the image to provide the local spectrum. As in [12] we choose 2D Gabor functions:

$$h(\mathbf{x}, \mathbf{u}) = g(\mathbf{x})e^{2\pi j \mathbf{x}\cdot\mathbf{u}} \quad \text{where} \quad g(\mathbf{x}) = \frac{1}{2\pi\gamma^2} e^{\frac{-(\mathbf{x}\cdot\mathbf{x})}{2\gamma^2}} \ , \tag{1}$$

with j the unit imaginary and $g(\mathbf{x})$ a 2D Gaussian function with variance γ^2.

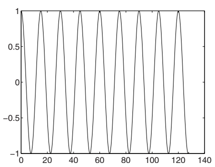
For a 2D cosine $f(\mathbf{x}) = \cos(2\pi\Omega(\mathbf{x}))$ the instantaneous frequency is given by

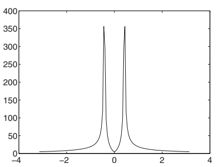
$$\tilde{\mathbf{u}}(\mathbf{x}) = (\tilde{u}(\mathbf{x}), \tilde{v}(\mathbf{x})) = \left(\frac{\partial\Omega}{\partial x}, \frac{\partial\Omega}{\partial y}\right) . \tag{2}$$

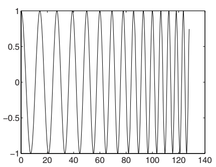
Our goal is to measure $\tilde{\mathbf{u}}(\mathbf{x})$. [14] shows that this can be done by considering a Gabor function $h(\mathbf{x}, \mathbf{u})$, and its two first order derivatives, $h_x(\mathbf{x}, \mathbf{u})$ and $h_y(\mathbf{x}, \mathbf{u})$:

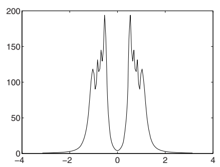
$$|\tilde{u}(\mathbf{x})| = \frac{|h_x(\mathbf{x}, \mathbf{u}) * I(\mathbf{x})|}{2\pi|h(\mathbf{x}, \mathbf{u}) * I(\mathbf{x})|}$$
$$|\tilde{v}(\mathbf{x})| = \frac{|h_y(\mathbf{x}, \mathbf{u}) * I(\mathbf{x})|}{2\pi|h(\mathbf{x}, \mathbf{u}) * I(\mathbf{x})|} . \tag{3}$$

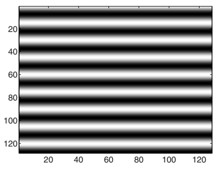
This estimate can be assumed to be correct if the measured frequency is in the pass-band of the filter. This method implies that we choose the central frequencies $\mathbf{u}$ and the spatial constants γ of the Gabor functions, i.e. their centers and width. The filters have constant *fractional bandwidth* (bandwidth divided by center frequency). This allows us to measure higher frequencies more locally than lower frequencies and is computationally less expensive. Moreover, as all filters so derived are geometrically similar it is simpler to compare their outputs.

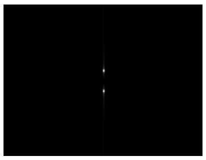

(a) 1D cosine at frequency $\tilde{u} \approx 0.42$ rad/s

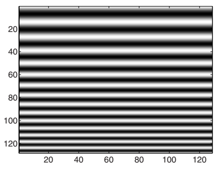

(b) Amplitude of the spectrum of the cosine in 1(a)

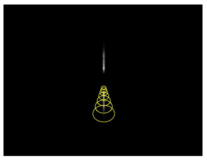

(c) 1D cosine. Frequency varies from $\tilde{u} \approx 0.42$ rad/s to $\tilde{u} \approx 1.27$ rad/s

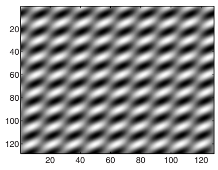

(d) Amplitude of the spectrum of the cosine in 1(c)

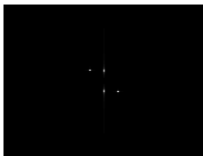

(e) 2D cosine at frequency $|\tilde{\mathbf{u}}| \approx 0.42$ rad/s

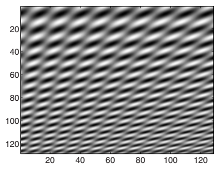

(f) Amplitude of the spectrum of the cosine in 1(e)


(g) 2D cosine. Frequency varies from $|\tilde{\mathbf{u}}| \approx 0.42$ rad/s to $|\tilde{\mathbf{u}}| \approx 1.27$ rad/s


(h) Spectrum amplitude of 1(g) and the chosen set of Gabor filters (circles) on it


(i) Two 2D cosines superposed, at frequencies $|\tilde{\mathbf{u}}_1| \approx 0.42$ rad/s and $|\tilde{\mathbf{u}}_2| \approx 0.63$ rad/s, 45° apart


(j) Amplitude of the spectrum of the image in 1(i)


(k) Result of slanting the image in 1(i) by 38°

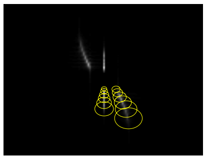

(l) Spectrum amplitude of 1(k) and the chosen set of Gabor filters (circles) on it

Fig. 1. Setting the Gabor filters' parameters

We set the Gabor functions using the information from the Fourier transform of the texture. Unlike Super and Bovik ([12]), who sample the whole 2D frequency plane, we make a selection of Gabor filters using ridges in the Fourier transform of the image. In our algorithm every ridge determines a set of Gabor filters that covers the corresponding values of frequencies. Every ridge therefore determines different instantaneous frequencies and thus different distortion measures.

2.2 Setting the Gabor Filter Parameters

Let us consider a 1D cosine (figure 1(a)). The signal has length of 128 samples and frequency $\tilde{u} \approx 0.42$ rad/s (where π rad/s is by convention the biggest admissible frequency). Figure 1(b) represents its spectrum amplitude, two symmetric spikes at the corresponding frequencies ($\approx \pm 0.42$ rad/s). A chirp is shown in figure 1(c), i.e. a cosine with frequency varying from $\tilde{u} \approx 0.42$ rad/s to $\tilde{u} \approx 1.27$ rad/s. Figure 1(d) illustrates its spectrum, where significant non-zero values span that range.

Analogously we show a 2D image generated by a 2D cosine with frequency $|\tilde{\mathbf{u}}| \approx 0.42$ rad/s (figure 1(e)) and its spectrum (figure 1(f)), given by symmetric spikes in the frequency plane. We then compare this to figures 1(g) and 1(h), the image of a 2D cosine with frequency ranging from $|\tilde{\mathbf{u}}| \approx 0.42$ rad/s to $|\tilde{\mathbf{u}}| \approx 1.27$ rad/s and its spectrum (circles are fully explained later). In the latter, significant non-zero values form a ridge corresponding to the range of frequencies. Figures 1(g) and 1(h) were actually generated by slanting (see section 3) image 1(e) through 38°. The ridges of the amplitude of the Fourier transform of the image represent the 2D frequencies contained in the texture.

The algorithm we propose analyzes the spectrum of the texture to determine its ridges, and then uses this information to define the sets of Gabor functions used. Figure 1(h) shows the chosen set of central frequencies $\mathbf{u}$ (the centers of the circles) and the set of spatial constants γ (their radii); half of the spectrum is considered because of its redundancy. There is significant overlapping (50%) to produce a robust LSF estimation. However, unlike in [12], where 63 central frequencies and spatial constants sample the whole 2D frequency plane, here the number used varies with the image. 7 $\mathbf{u}$'s and γ's are used in figure 1(h). This implies a significant reduction of the computational expense: in [12] 63 $\mathbf{u}$'s and γ's correspond to 378 convolutions (the Gabor filter and its first order derivatives and an equivalent number of post-smoothing filters); in this case 7 $\mathbf{u}$'s and γ's mean 42 convolutions, with a computational saving of about 89%.

We now consider multiple frequencies. Figure 1(i) shows a superposition of the cosine from the previous example ($|\tilde{\mathbf{u}}_1| \approx 0.42$ rad/s) on another with frequency $|\tilde{\mathbf{u}}_2| \approx 0.63$ rad/s, separated by 45° degrees from the first in the frequency plane. In this case the spectrum amplitude (figure 1(j)) has four peaks, corresponding to the values of the two frequencies of the cosines. In fact we can associate two instantaneous frequencies to each point, which coexist at every pixel. Figure 1(k) shows the result of applying the same slant as in 1(g): each cosine now has a continuously-varying frequency. Moreover the two LSFs change independently from each other, i.e. the first assumes the same values as in figure 1(h) and the second forms corresponding ridges in the direction of the slant. This is illustrated in figure 1(l), where the two ridges are functions of the original frequency and of the distortion due to the slant.

Our algorithm detects the two ridges and sets two groups of Gabor filters. In each group the central frequencies, $\mathbf{u}$, and the spatial constants, γ, are defined so that the filters cover the respective ridge area (figure 1(l)). Every set of filters is processed as in the previous example, i.e. as if the texture contained only one corresponding LSF. Thus each set of filters reconstructs an instantaneous frequency for each pixel. LSFs measure distortion and are combined to reconstruct the shape (see section 4). In this sense we exploit the multi-scale nature of the texture, because all different-scale frequencies are considered in the final result.

3 Projection of Texture

Here we describe the viewing geometry and a projection model, to provide a relationship between the surface texture and the image plane frequencies as a function of the shape of the surface. We then present a surface texture model.

3.1 Viewing Geometry and Projection Model

We adopt the viewing geometry and projection model of [12], and from that we take our motivation to extend the theory to general developable surfaces. They deal with the case of a planar surface and assume a pin-hole camera model. Their coordinate systems are given in figure 2. In this figure, the origin of the world coordinate system $\mathbf{x}_w = (x_w, y_w, z_w)$ coincides with the focal point and the optical axis coincides with the $-z_w$-direction. The image plane coordinate system $\mathbf{x}_i = (x_i, y_i)$ is placed at $z_w = f < 0$, with $|f|$ being the focal length, such that $x_i = x_w$ and $y_i = y_w$. The orientation of the surface is described using the slant-tilt system: the slant σ is the angle between the surface normal and the optical axis, with values ranging from 0° to 90°; the tilt τ is the angle between the x_i-axis and the projection on the image plane of the surface normal, with values between −180° and 180°. The surface is described by the coordinate system $\mathbf{x}_s = (x_s, y_s, z_s)$: the x_s-axis is aligned with the perceived tilt direction, the z_s-axis is aligned with the surface normal, y_s forms a right handed orthogonal coordinate system and the origin of $\mathbf{x}_s$ is on the intersection of the surface with the z_w-axis, at $z_w = z_0 < 0$.

[12] illustrates the equations for transforming 2D surface to 2D image coordinates, and vice versa, under perspective projection. Most importantly, they derive the relationship between the instantaneous frequencies on the image plane $\mathbf{u}_i = (u_i, v_i)$ and those on the surface plane $\mathbf{u}_s = (u_s, v_s)$:

$$\mathbf{u}_s = J^t(\mathbf{x}_i, \mathbf{x}_s) \cdot \mathbf{u}_i. \tag{4}$$

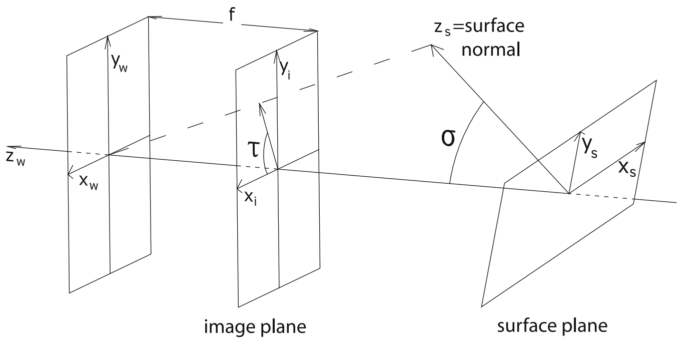


Fig. 2. Viewing geometry and projection model

J^t, the transpose of the Jacobian of the transformation, is

$$J^t(\mathbf{x}_i) = \frac{\sin\sigma}{z_w}\begin{bmatrix} x_i & y_i \\ 0 & 0 \end{bmatrix} + \frac{f}{z_w}\begin{bmatrix} \cos\sigma\cos\tau & \cos\sigma\sin\tau \\ -\sin\tau & \cos\tau \end{bmatrix}, \tag{5}$$

where z_w is the corresponding coordinate, in the $\mathbf{x}_w$ coordinate system, of the surface point which projects to the image point $\mathbf{x}_i$.

The above described coordinate systems can be easily extended to general developable surfaces: assuming that the surface is smooth and that at any point it can be locally approximated with the corresponding tangent plane, the equations from [12] then apply to the tangent plane of the considered point.

Given the homogeneity assumption in section 1, frequencies $\mathbf{u}_s$ corresponding to the same LSF must be the same. However the $\mathbf{x}_s$ coordinate system changes from point to point on the surface texture according to the tilt-direction, because the x_s axis is aligned with it. In particular, between two image points $\mathbf{x}'_i$ and $\mathbf{x}''_i$ with measured frequencies $\mathbf{u}'_i$ and $\mathbf{u}''_i$, if the difference of tilt-direction of the corresponding points on the surface is $\varDelta\tau = \tau'' - \tau'$, we have:

$$\mathbf{u}'_s = \mathbf{u}''_s \tag{6}$$

$$J^t(\mathbf{x}'_i)\cdot\mathbf{u}'_i = \begin{bmatrix} \cos\varDelta\tau & \sin\varDelta\tau \\ -\sin\varDelta\tau & \cos\varDelta\tau \end{bmatrix} J^t(\mathbf{x}''_i)\cdot\mathbf{u}''_i \tag{7}$$

where the matrix rotates the $\mathbf{x}_s$ coordinate systems by $\varDelta\tau$ to align them.

Eq. 7 also applies in cases of orthographic projection: the relation between the surface tangent and the image coordinates in [12] reduces to

$$\mathbf{x}_i = \begin{bmatrix} \cos\sigma\cos\tau & -\sin\tau \\ \cos\sigma\sin\tau & \cos\tau \end{bmatrix}\mathbf{x}_s, \tag{8}$$

and the J^t in eq. 7 becomes

$$J^t = \begin{bmatrix} \cos\sigma\cos\tau & \cos\sigma\sin\tau \\ -\sin\tau & \cos\tau \end{bmatrix}. \tag{9}$$

3.2 Surface Texture Model

We model textures as due to variations of surface reflectance, the proportion of reflected light. We assume surfaces with Lambertian reflection, and that the texture is therefore 'painted' on them, without roughness or self-occlusion.

Surface reflectance, $t_s(\mathbf{x}_s)$, and image reflectance, $t_i(\mathbf{x}_i)$, are related by

$$t_i(\mathbf{x}_i) = k(\mathbf{x}_i)\cdot t_s[\mathbf{x}_s(\mathbf{x}_i)], \tag{10}$$

where $\mathbf{x}_s(\mathbf{x}_i)$ represents the perspective backprojection, while $k(\mathbf{x}_i)$ is a multiplicative shading term. [11] shows how to estimate and remove k. However, if the scale of variation of t_s is small compared to the scale of variation of the shading term, then the latter can be assumed to be constant in any small neighborhood. Moreover, our method automatically normalizes for slow variations in illumination, shading and surface texture contrast, because it uses frequencies rather than amplitudes. Also no assumption is made about the textural nature of $t_s(\mathbf{x}_s)$, thus it might apply to various patterns, e.g. lines, blobs, etc.

4 Computing Surface Orientation

We explain here how our algorithm processes the image texture to produce the shape of the surface texture.

After the LSFs are computed as indicated in section 2, eq. 7 is used to estimate the corresponding σ's and τ's on the surface (the orientation parameters), results are then combined and used to reconstruct the shape.

In the orthographic case, substituting for J^t from eq. 9 in eq. 7 we get two equations in 4 unknowns: σ', τ', σ'', τ''. Assuming we know the orientation (σ', τ') of the tangent plane at point P', the surface texture point which projects to $\mathbf{x}'_i$, we can solve eq. 7 for σ'' and τ'' (the orientation parameters at point P'', respectively corresponding to $\mathbf{x}''_i$). The analytical solution for τ'' is

$$\tau'' - \tau' = \tan^{-1} \frac{u' \sin\tau' - v' \cos\tau' - u'' \sin\tau' + v'' \cos\tau'}{\cos\sigma'(u' \cos\tau' + v' \sin\tau') + u'' \cos\tau' + v'' \sin\tau'} = \Delta\tau, \quad (11)$$

which provides two values in the $[0, 2\pi]$ range, as for the well-known tilt-ambiguity:

$$\tau'' = \begin{cases} \tau' + \Delta\tau \\ \tau' + \Delta\tau + \pi \end{cases} . \quad (12)$$

σ'' is given by

$$\sigma'' = \cos^{-1} \left[\frac{\cos\sigma'(u' \cos\tau' + v' \sin\tau') + u'' \cos\tau' + v'' \sin\tau'}{\cos(\tau'' - \tau')(u'' \cos\tau'' + v'' \sin\tau'')} - 1 \right] . \quad (13)$$

The perspective equations are obtained by substituting for J^t from eq. 5 in eq. 7. In this case, besides the σ's and τ's, z' and z'' are unknown, the z_w coordinates at the surface points P' and P''. As in the orthographic case, we assume z', σ' and τ' are known, i.e. the position and orientation of P' are assumed known. Furthermore z'' is inferred from the assumed orientation parameters, and then refined once the values of σ'' and τ'' are computed: in the first instance z'' is assigned a value as if P'' lied on the tangent plane at P' (its orientation is given by σ' and τ'), this produces two equations in two unknowns and enables us to compute σ'' and τ'', which are then used to refine z''. Experimental results show that assuming P'' initially on the tangent plane at P' gives accurate results for σ'' and τ'' if P' and P'' correspond to close points on the image plane. Our algorithm therefore uses a known point P to compute the orientation parameters of its neighboring points. Below is the closed form solution for τ''

$$\tau'' - \tau' = \tan^{-1} \left(-\frac{b}{a} \right) = \Delta\tau, \quad (14)$$

$$\text{where} \quad a = \frac{\sin\sigma'}{z'}(u'x' + v'y') + \frac{f}{z'} \cos\sigma'(u' \cos\tau' + v' \sin\tau') + \frac{f}{z''}(u'' \cos\tau' + v'' \sin\tau'), \quad (15)$$

$$b = \frac{f}{z'}(v' \cos\tau' - u' \sin\tau') + \frac{f}{z''}(u'' \sin\tau' - v'' \cos\tau'), \quad (16)$$

giving two solutions (eq. 12) corresponding to the well-known tilt-ambiguity. σ'' is then given by

$$\sigma'' = \sin^{-1} \frac{-c(b+d) \pm |d|\sqrt{c^2 - 2bd - b^2}}{c^2 + d^2}, \tag{17}$$

$$\text{where } b \text{ is as above and } c = \frac{1}{z''} \sin(\tau'' - \tau')(u''x'' + v''y''), \tag{18}$$

$$d = \frac{f}{z''} \sin(\tau'' - \tau')(u'' \cos\tau'' + v'' \sin\tau''). \tag{19}$$

The two sets of equations show how a single LSF can be used to compute the σ's and τ's, point after point, at all image points, and hence reconstruct the shape. However we choose to use the LSFs combined in pairs in order to get a more robust estimation: for each image point each of the two LSFs gives a solution (σ, τ), then the one that best approximates the equation defined by the other is chosen. Experimental results show that this produces more robust estimates than if finding a solution numerically by combining the equations for the two, or even more, LSFs. Note that this does not reduce the applicability of our method because most real textures have at least two LSFs.

For computing the shape, our algorithm needs a starting point with known orientation. As in [13], we consider the following alternatives: the orientation of a point can be manually input, or we can assume that the image displays a frontal point (for which $\sigma = 0$), which can be detected using the method of [6]. The latter easily integrates with our algorithm, because it uses the same Gabor filters which we adopted for computing the LSFs, and is therefore convenient because most of the computation is shared with the LSF estimation. As described in section 2.2, we do not use the whole lattice of filters, but we choose to separate and process only the relevant LSFs which we determine via the Fourier transform. In particular, the spectral second-order moments defined in [6] are here computed using two LSFs, as if the spectrum of the image contained only them:

$$F(\mathbf{u}) = \delta(\mathbf{u} - \mathbf{u}_1) + \delta(\mathbf{u} - \mathbf{u}_2). \tag{20}$$

The product of the canonical moments $\sqrt{mM}$ (see [6]) reduces therefore to

$$\sqrt{mM} = |u_1 v_2 - u_2 v_1|. \tag{21}$$

It can be shown that the above is the inverse of the area gradient (see [2]), computed using the two frequencies. As in [6], the minimum gives the frontal point ($\sigma = 0$), while the gradient direction gives the initial τ value.

With the method described, we get two reconstructed shapes for each pair of LSFs. By using eq. 4 and 7 and the determined σ's and τ's, we can backproject the frequencies $\mathbf{u}_i$ to the surface texture and align them. The homogeneous assumption states that the frequencies computed in this way should be the same and that their variance should therefore be zero. Hence we choose the shape associated with the lowest variance, assuming that lower values, closer to the ideal zero value, correspond to better estimates.

Below is the proposed algorithm:

- The spectrum amplitude of the image is analyzed and ridges are detected.
- Each ridge determines a set of Gabor functions and their first derivatives, so that the filters cover the frequencies pertaining to the particular ridge.
- For each set of filters the following steps are repeated:
  - the image is convolved with the Gabor filters and their derivatives, and the outputs are smoothed with a Gaussian to reduce noise;
  - the Gabor filter with largest amplitude output is selected at each point;
  - the instantaneous frequencies are computed at each point (eq. 3).
- For each pair of LSFs the orientation of a point is manually input or a frontal point is found (eq. 21).
- For each pair of LSFs a shape is reconstructed (eq. 7) and the variance of backprojected and aligned image frequencies is computed.
- The best shape is chosen according to the lowest variance.

We plan to use the backprojection of frequencies, and their theoretical equal values, in order to identify and discard possible outliers. Also, as many of the reconstructed shapes from the pairs of LSFs are accurate, future work might address combining these to produce better estimates.

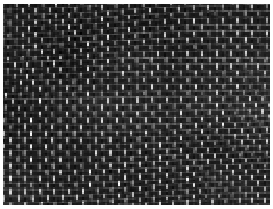
Finally, the algorithm lends itself to parallel implementation, because ridges and filters can be processed independently and implemented by different units.

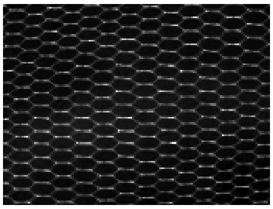
5 Results

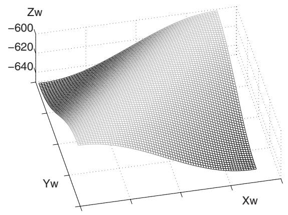
We demonstrate our method on two sets of images. The first images (figures 3(a),(b)) were synthesized by mapping real textures from the Brodatz database ([15]) onto cosine surfaces (figures 3(c),(f)) and then rendering as a new image. The second (figures 4(a),(d),(g)) are real images acquired with a Pulnix TM-6EX camera. They are the central parts of 640x480 pictures of textured objects laid on a cylinder. Being real images, they are affected by variations in illumination, self shadowing (4(a)) and imperfections (4(g)). The images were not retouched before processing, and are 128x128 pixels with 256 levels of gray, with the exception of 4(g), which shows the case of a bigger extracted image (180x180).

Table 1. Estimation errors for σ and τ for orthographic and perspective cases (degrees)

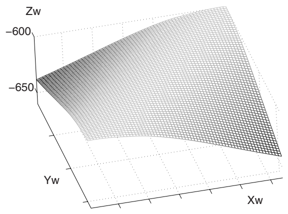
	Average error - Orthographic		**Average error - Perspective**	
Image	$\lvert\epsilon_\sigma\rvert$	$\lvert\epsilon_\tau\rvert$	$\lvert\epsilon_\sigma\rvert$	$\lvert\epsilon_\tau\rvert$
D6	5.24	3.10	1.37	3.10
D34	4.80	1.98	2.88	1.93
sponge	1.55	4.30	1.21	4.27
trousers	1.48	1.92	1.86	1.87
rubber rug	4.28	0.44	4.39	0.62


(a) D06

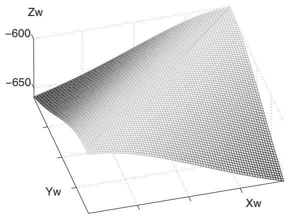

(b) D34



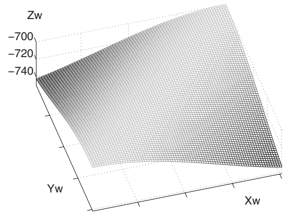
(c) D6 - original shape



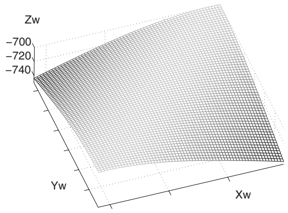
(d) D6 - orthographic reconstruction



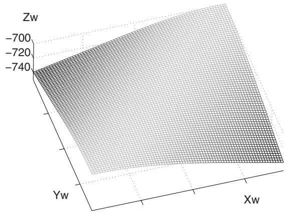
(e) D6 - perspective reconstruction



(f) D34 - original shape



(g) D34 - orthographic reconstruction



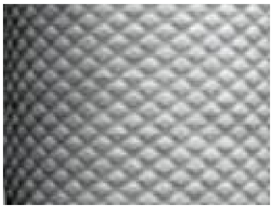
(h) D34 - perspective reconstruction

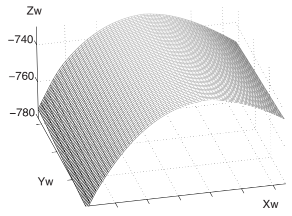
Fig. 3. Images synthesized from Brodatz textures, original shapes and reconstructions

Figures 3(d),(g) and 4(b),(e),(h) show the shapes reconstructed via the orthographic equations (9,7), while figures 3(e),(h) and 4(c),(f),(i) show those reconstructed via the perspective equations (5,7). In all cases the shapes were achieved by using the values of the orientation parameters (σ,τ), which were computed for all corresponding pixels of the original images. Although the estimates were not smoothed, the surfaces do look smooth and close to the original (or to part of a cylinder for real images). Table 1 shows the average errors of the estimated orientation parameters. In fact the ground truth was known for the first set of images, and it could be extracted from the images for the second set because the diameters of the cylinders and their distances from the camera were known. The errors confirm both the accuracy and robustness of our algorithm.

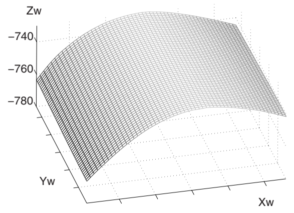
As stated in section 1, the homogeneity assumption requires some sort of periodicity/stationarity: the algorithm worked well up to cases when the periodicity of the texture was just one order of magnitude bigger than the main frequency associated with the shape.

Furthermore, we address the possibility that ridges might superimpose. This may be the case when a texture composed of close frequencies is slanted. The

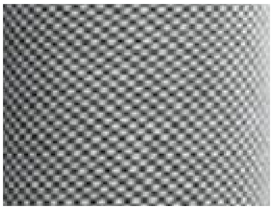

(a) Sponge

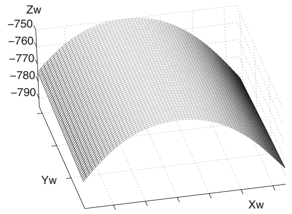


(b) Sponge - orthographic reconstruction

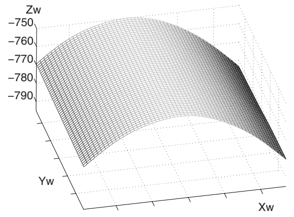


(c) Sponge - perspective reconstruction

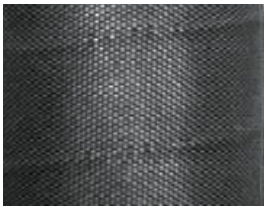

(d) Trousers

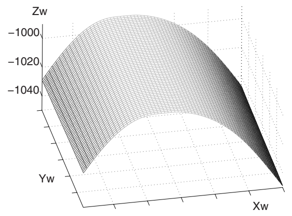


(e) Trousers - orthographic reconstruction

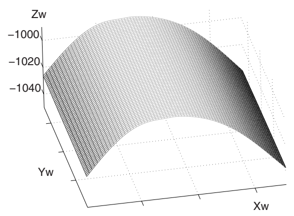


(f) Trousers - perspective reconstruction


(g) Rubber rug



(h) Rubber rug - orthographic reconstruction



(i) Rubber rug - perspective reconstruction

Fig. 4. Real images of textures and reconstruction

superposition can be spotted as it results in gaps in the frequency estimation. By considering smaller patches of the image (e.g. 96x96 instead of 128x128), the range of variation of frequencies analyzed by the Fourier transform is smaller and hence there is less chance of observing the superposition.

The pictures in this paper show the application of the algorithm to relatively simple developable shapes, where the tilt-ambiguity is solved by assuming convexity. However we plan to extend our simple and robust method to deal with general shapes, using a technique such as the EM algorithm to disambiguate the tilt. Finding a robust solution to shape from texture will open the way to possible applications such as the rendering and the retexturing of clothing ([10]).

6 Conclusions

The study presented here has characterized the variations of the dominant LSFs in textures via the ridges of their Fourier transforms, and used those to estimate the shapes of developable and convex surface textures. Examples of reconstruction of both semi-synthetic and real images have been given, and errors have been computed in both cases. Our algorithm is accurate, simple to implement, and has the potential to be extended to general surfaces, by addressing non-developable surfaces and implementing an EM-type algorithm to solve the tilt-ambiguity.

To our knowledge, the proposed algorithm is the first to consider the multi-scale nature of texture to the extent of exploiting all main LSFs. Furthermore, it is robust against shading, variations in illuminations, and occlusions. Finally, it is based on the Fourier transform of the image and on a minimal number of convolutions, results are therefore computationally fast.

Acknowledgements

Fabio Galasso is grateful to St. John's college and Giovanni Aldobrandini for the 'A Luisa Aldobrandini' Scholarship, supporting his graduate studies.

References

1. Gibson, J.J.: The Perception of the Visual World. Houghton Mifflin, Boston, Massachusetts (1950)
2. Gårding, J.: Shape from texture for smooth curved surfaces in perspective projection. JMIV 2, 327–350 (1992)
3. Hwang, W.L., Lu, C.S., Chung, P.C.: Shape from texture: Estimation of planar surface orientation through the ridge surfaces of continuous wavelet transform. IEEE Trans. Image Processing 7, 773–780 (1998)
4. Kanatani, K., Chou, T.C.: Shape from texture: general principle. Artificial Intelligence 38, 1–48 (1989)
5. Malik, J., Rosenholtz, R.: Computing local surface orientation and shape from texture for curved surfaces. IJCV 23, 149–168 (1997)
6. Super, B.J., Bovik, A.C.: Shape from texture using local spectral moments. IEEE Trans. Patt. Anal. Mach. Intell. 17, 333–343 (1995)
7. Clerc, M., Mallat, S.: Shape from texture through deformations. Int. Conf. Comp. Vision, 405–410 (1999)
8. Loh, A.M., Hartley, R.: Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. In: BMVC, pp. 69–78 (2005)
9. Witkin, A.P.: Recovering surface shape and orientation from texture. Artificial Intelligence 17, 17–45 (1981)
10. Lobay, A., Forsyth, D.A.: Shape from texture without boundaries. IJCV 67, 71–91 (2006)
11. Clerc, M., Mallat, S.: The texture gradient equation for recovering shape from texture. IEEE Trans. Patt. Anal. Mach. Intell. 24, 536–549 (2002)
12. Super, B.J., Bovik, A.C.: Planar surface orientation from texture spatial frequencies. Pattern Recognition 28, 729–743 (1995)
13. Loh, A.M., Kovesi, P.: Shape from texture without estimating transformations. Technical report, UWA-CSSE-05-001 (2005)
14. Bovik, A.C., Havlicek, J.P., Desai, M.D.: Theorems for discrete filtered modulated signals. In: ICASSP, Minneapolis, MN, pp. 805–810 (1993)
15. Brodatz, P.: Textures: A Photographic Album for Artists and Designers. Dover, New York (1966)

Skeleton-Based Data Compression for Multi-camera Tele-Immersion System

Jyh-Ming Lien[1], Gregorij Kurillo[2], and Ruzena Bajcsy[2]

[1] George Mason University, Fairfax, VA
[2] University of California, Berkeley, CA

Abstract. Image-based full body 3D reconstruction for tele-immersive applications generates large amount of data points, which have to be sent through the network in real-time. In this paper we introduce a skeleton-based compression method using motion estimation where kinematic parameters of the human body are extracted from the point cloud data in each frame. First we address the issues regarding the data capturing and transfer to a remote site for the tele-immersive collaboration. We compare the results of the existing compression methods and the proposed skeleton-based compression technique. We examine robustness and efficiency of the algorithm through experimental results with our multi-camera tele-immersion system. The proposed skeleton-based method provides high and flexible compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality (peak signal-to-noise ratio from 28 to 31 dB).

1 Introduction

Tele-immersion (TI) is aimed to enable users in geographically distributed sites to collaborate and interact in real time inside a shared simulated environment as if they were in the same physical space [1]. In addition, the virtual environment can include different synthetic objects (e.g. three-dimensional models of buildings) or captured data (e.g. magnetic resonance image of a brain) which can be explored by the users through three-dimensional (3D) interaction. The TI technology combines virtual reality for rendering and display purpose, computer vision for image acquisition and 3D reconstruction, and various networking techniques for transmitting the data between the remote sites in real-time with the smallest delays possible. TI is aimed at different network applications, such as collaborative work in industry and research, remote evaluation of products for ergonomics, remote learning and training, coordination of physical activities (e.g. dancing [2], rehabilitation), and entertainment (e.g. games, interactive music videos). In addition, 3D data captured locally could be used for kinematic analysis of body movement (e.g. in medicine and rehabilitation) or in computer animation [3].

To render realistic model of the TI user inside the virtual space, real-time 3D capturing of the human body is needed. Human body can be captured using multi-camera system which allows extraction of depth information. Different approaches for real-time processing of depth information have been proposed. One of the first TI systems was presented by the researchers at University of Pennsylvania [4]. Their system consisted of several stereo camera triplets used for the image-based reconstruction of the upper body, which allowed a local user to communicate to remote users while sitting behind a desk. A simplified version of the desktop tele-immersive system based on reconstruction from silhouettes was proposed by Baker et al. [5]. Blue-c tele-immersion system presented by Wurmlin et al. [6] allowed full-body reconstruction based silhouettes obtained by several cameras arranged around the user. Reconstruction from the silhouettes provides faster stereo reconstruction as compared to image-based methods; however, this approach lacks accuracy and discrimination of several persons or objects inside the system. The TI system presented in this paper introduces 360-degree full-body 3D reconstruction from images using twelve stereo triplets [7]. The captured 3D data are transferred using TCP/IP protocol to the rendering computer which reconstructs and displays point clouds inside a virtual environment at the local or remote sites in real time. The presented system has been successfully used in remote dancing applications [2] and learning of tai-chi movements [8][1].

One of the major bottlenecks of our current system for tele-immersion is the real-time transfer of large amount of data points, which are generated by the stereo reconstruction from multiple images, to the remote site. Current image/video-based compressor [2] in our TI system only allows transmission of data with the rate of 5 or 6 frames per second (fps). In the future we plan to improve the resolution of the captured images which will additionally increase the bandwidth requirements for transfer of data in real time. In this paper, we focus on the data compression using motion estimation for TI applications. The proposed compression method provides high and flexible compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality (peak signal-to-noise ratio from 28 to 31 dB). We describe a novel technique of skeleton-based compression of 3D point data captured by our system. Robustness and efficiency of the algorithm is examined theoretically and experimentally. We discuss how to compress the data using the estimated articulated motions and how to encode non-rigid motions using regular grids (residual maps). Finally, we propose several efficient ways to detect temporal coherence in our residual maps including accumulating residuals from a few frames and compress them using established video compression techniques.

2 Overview of the Tele-Immersion Apparatus

Our tele-immersion apparatus consists of 48 Dragonfly cameras (Point Grey Research Inc, Vancouver, Canada) which are arranged in 12 clusters covering 360 degree view of the user(s). The cameras, equipped with 6 and 3.8 mm lenses, are mounted on an aluminum frame with dimensions of about 4.0 x 4.0 x 2.5 m^3 (Fig. 1a). The arrangement of cameras was optimized to increase the usable workspace coverage to about 2.0 x 2.0 x 2.5 m^3. Each cluster consists of three black and white cameras intended for stereo reconstruction and a color camera used for texture acquisition. The four cameras in each cluster are connected through a fire-wire interface to a dedicated personal computer. The cluster PCs are dual or quad CPU (Intel Xeon, 3.06 GHz) machines with 1GB of memory and 1 Gbps connection to Internet 2. To synchronize image acquisition on all 48 cameras, the trigger computer generates an analogue signal sent to all the cameras

[1] Movies and images are available at http://tele-immersion.citris-uc.org

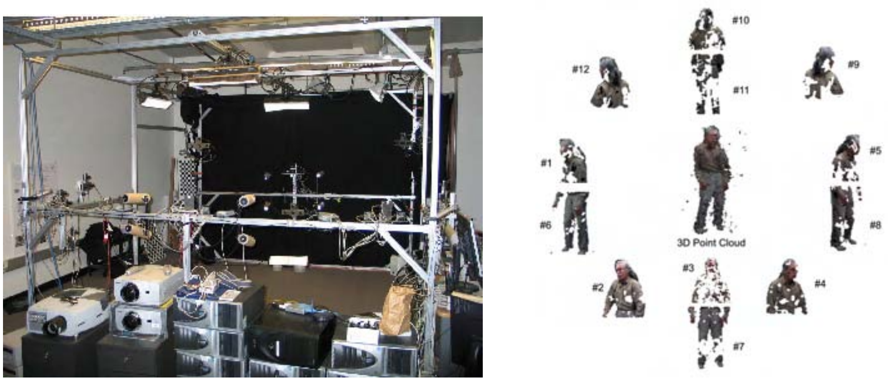


Fig. 1. (a) Multi-camera stereo system for tele-immersion consists of 48 cameras arranged in 12 stereo clusters. The captured images are processed by 12 multi-core computers to provide 360-degree 3D full-body reconstruction in real time. (b) Partial depth image data captured by each of the 12 stereo camera clusters, which are accurately calibrated to a reference coordinate system, is combined inside the renderer into a full-body 3D reconstruction.

in each frame. The cluster computers notify the trigger computer via TCP/IP messaging when the image acquisition and reconstruction have been completed. Although the cameras allow capturing of images with the resolution of 640 by 480 pixels, the images are resized to 320 by 240 pixels to reduce the computational load. Based on the intrinsic and extrinsic parameters of each camera obtained during the calibration, the image is first rectified and distortion of the lens is corrected. The background is subtracted using a modified Gaussian average algorithm [9]. In the next step, edges of foreground objects are extracted and regions with similar features (i.e. texture) are identified to perform region based correlation among the images captured by the stereo triplet [4]. Next, the depth map is computed from the three gray scale images using triangulation method. The reconstruction algorithm has been paralleled to exploit the multi-core technology. The image data is equally split between the CPUs to increase the speed of the 3D reconstruction process. The accuracy of the stereo reconstruction is mainly affected by the quality of calibration. The errors can occur due to lens distortion, misalignment between image and lens plane, and deviations of position and rotation between the stereo cameras [10],[11]. The calibration of our TI system is performed in two steps. The intrinsic calibration of camera parameters (i.e. distortion, optical center) is performed by Tsai algorithm [12] using a checkerboard. Extrinsic parameters of each camera (i.e. position, orientation) are obtained by capturing LED position in about 10,000 points located inside the overlapping volume between the calibrated and reference camera. The captured points are re-projected to the reference camera plane while non-linear optimization is used to reduce the errors. The quality of the stereo reconstruction is further affected by illumination conditions, texture and color of the objects, and arrangement of the camera clusters around the workspace. The illumination inside the tele-immersion apparatus is diffused by filters to reduce shadows and specular highlights which interfere with stereo matching algorithm.

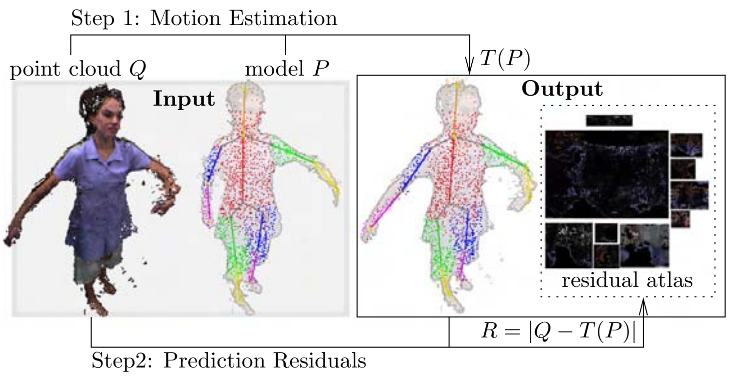


Fig. 2. Skeleton-based compression of the point data Q, where P is the skeleton, and T is the motion parameter (e.g., the skeleton configuration) that transforms P to Q. The prediction residuals R is the difference between $T(P)$ and Q.

To transfer the data to the rendering computer, the acquired depth map (2 bytes per pixel) is combined with the color map (3 bytes per pixel) and compressed by run-length encoding and z-lib [13] loss-less compression algorithm [2]. The compressed data package is sent through the network using TCP/IP protocol. The size of each data package depends on the image coverage and ranges from about 25 to 50 KBs for imaging one person inside the system. Data streams from the twelve calibrated clusters are composed into a 3D image by a point-based rendering application developed using OpenGL. Based on camera calibration parameters the renderer combines the points received from each cluster into the full 3D reconstruction of the tele-immersion user (Fig. 1b). The complexity of the stereo reconstruction is currently limiting the frame rate to about 5 to 6 frames per second (fps). The required network bandwidth for this acquisition rate is below 5 Mbps. With increased frame rate and resolution of the captured data the bandwidth requirements can exceed 1 Gbps, therefore different compression techniques of the 3D data are needed.

3 Skeleton-Based Compression

One of the major bottlenecks of our current system for tele-immersion is the transfer of large amount of data points. For example, data from images of 640x480 pixel depth and color maps from 10 camera clusters with a frame rate of 15 fps would require 220 MB/s network bandwidth. Several methods [6],[2] have been proposed to address this problem using image and video compression techniques, but the data volume remains to be prohibitively large. Currently, our compression method can only reach 5 to 6 fps when connected with another remote TI site. In this section we will discuss our solutions to address the problems in TI data compression from a model driven approach. The main idea of this work is to take advantage of prior knowledge of objects, e.g. human figures, in the TI environments and to represent their motions using smaller number of parameters, e.g., joint positions and angles. The data transfer can be significantly reduced by introducing compression algorithms based on extracted kinematic parameters (i.e. joint angles) of people captured by the stereo cameras.

We propose a new compression method based on human motion estimates. Instead of transmitting the point clouds, we can simply transmit the motion parameters. This approach is based on the assumption that most points move under rigid-body transform along with the skeleton. In reality, point movements may deviate from this assumption, such as muscle movements and hair or cloth deformations; therefore, we further compress the deviations from the rigid movements. As it will be shown in the remainder of the paper, the deviations (which is called "prediction residuals") in most cases are small. An overview of this model driven approach is shown in the figure below (Fig. 2). Our compressor provides (flexible) high compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality. Our method can estimate motions from the data captured by our TI system in real time (10+ fps).

3.1 Skeleton and Motion Estimation

We extend Iterative Closest Points (ICP) [14] to estimate motion of dynamic point data in real time. Our approach assumes the skeleton(s) of the person(s) represented by points in the first frame is given. Several methods exist [15],[16] and can provide us an initial skeleton to start the process. We then apply a real-time tracking method which fits the skeleton to the point cloud data captured from the rest of the movements.

Extensive work has been done to track human motion in images (see surveys [17,18]). Our goal is to estimate motion from 3-D points. Algorithms 3.1 and 3.2 outline our approach. The input parameters of Algorithm 3.1 include the skeleton S and the points Q that we would like S to fit to. The algorithm starts by fitting the points (P_l) associated with the root link (l_{root}) of S to Q using ICP (see Algorithm 3.2) and then fits the rest of the links hierarchically, i.e., from the torso to the limbs. Finally, once all links are roughly aligned with the point cloud Q, we perform a global fitting which tries to minimize the *global* difference between the skeleton S and Q. Details of global fitting can be found in [19].

Algorithm 3.1: ARTICP(S, Q, τ)

```
cluster $Q$
$q$.push($l_{root}$)
while $q \neq \emptyset$
  do
    $l \leftarrow q$.pop()
    $T \leftarrow$ICP($P_l, Q, \tau$)
    for each child $c$ of $l$
      do
        apply $T$ to $c$
        $q$.push($c$)
global fitting
```

Algorithm 3.2: ICP(P, Q, τ)

```
repeat
  find corresponding points $\{(p_i \in P, q_i \in Q)\}$
  compute $error$ and $T$ in Eq. 1
  $P = T(P)$
until $error < \tau$
return ($T$)
```

Given two point sets P and Q, ICP first computes corresponding pairs $\{(p_i \in P, q_i \in Q)\}$. Using these corresponding pairs, ICP computes a rigid-body transform T such that the "matching error" defined in Eq. 1 between P and Q is minimized.

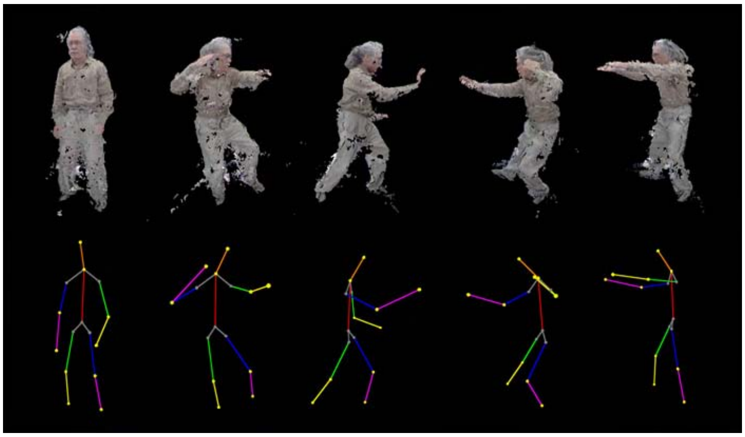


Fig. 3. Top: Stereo reconstruction of our tele-immersion system. Bottom: Real-time skeleton extraction from 3D point cloud data.

$$error = \underset{T}{\operatorname{argmin}} \sum_i |(T(p_i), q_i)|^2 . \tag{1}$$

The main step for minimize the matching error (see details in [20]) is to compute the cross-covariance matrix Σ_{PQ} of the corresponding pairs $\{p_i, q_i\}$,

$$\Sigma_{PQ} = \frac{1}{n} \sum_{i=1}^{n} [(p_i - \mu_p)(q_i - \mu_q)^t] , \tag{2}$$

where μ_p and μ_q are the centers of $\{p_i\}$ and $\{q_i\}$, resp., and n is the size of $\{p_i, q_i\}$. As outlined in Algorithm 3.2, ICP iterates these steps until the error is small enough.

Figure 3 illustrates the result produced by our motion estimation from a sequence of tai-chi movements. The main features of our motion estimation include: (a) Hierarchical fitting for faster convergence, (b) Articulation constraint, (c) Monotonic convergence to local minimum guaranteed, and (d) Global error minimization. Due to the space limitation, we refer interested readers to [19] for detail.

3.2 Prediction Residuals

Motion estimation brings the skeleton S close to the current point cloud Q. However, due to several reasons, e.g., non-rigid movements and estimation errors, our model, the points P_S associated with the skeleton S, may not match Q exactly. We call the difference between P_S and Q ”prediction residuals” (or simply residuals). Because P_S and Q are close to each other, we expect the residuals to be small. In this section, we present a method to compute the prediction residuals.

First, for each link l in the skeleton, we have two point set associated with l, namely, P_l and Q_l which are points from P_S and Q, respectively, and are closest to l than other links of S Then we project both P_l and Q_l to a regular 2-D grid embedded in a cylindrical coordinate system defined by the link l (see Fig. 4). Because P_l and Q_l are now

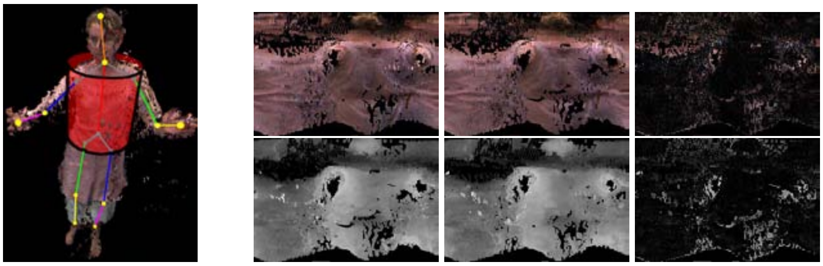


Fig. 4. A skeleton and the torso link is shown as a cylinder. (a) Color and depth maps at time $t-1$ of the torso link. The Y-axis of the maps is parallel to the link. (b) Color and depth maps at time t of the torso link. (c) The differences between the maps at times $t-1$ and t.

Table 1. Efficiency of skeleton-based motion estimation method. On average, our method achieves 11+ frames per second (fps).

motion	dancer 1 (Fig.3)	dancer 2	tai-chi student	tai-chi teacher
average fps	11.9 fps	11.5 fps	12.5 fps	12.6 fps

encoded in regular grids, we can easily compute the difference, which can be compressed using image compression techniques. Because this projection is invariant from a rigid-body transform, we only need to re-sample Q_l at each time step. We determine the size of a grid from the shape of a link l and the size of l's associated points P_l . We make sure that our grid size is at least $2|P_l|$ using the following formulation, i.e., the width and the height of the grid are $2\pi R_l S$ and $L_l S$, resp., where R_l and L_l are the radius and the length of the link l and $S = \sqrt{\frac{|P_l|}{\pi R_l L_l}}$. We call that a grid encodes 100% prediction residuals if the grid has size $2|P_l|$. As we will see later, we can tune the grid size to produce various compression ratios and qualities.

4 Results

In this section, we study the quality of our skeleton-driven compression on the TI data with various levels of prediction residuals. We also compare our model-driven compression to H.264 video compression [21] and Yang et al. method [2]. All the experimental results in this section are obtained using a Pentium4 3.2GHz CPU with 512 MB of RAM. We evaluate the results of our motion estimation method using four motion sequences captured by our TI system. These motions are performed by the dancers, one student and one tai-chi master (Fig.2). The data captured by our TI system have about 75,000 3D points in each frame. Table 1 shows that we can maintain at least 11 fps interactive rate in all studied cases.

4.1 Quality

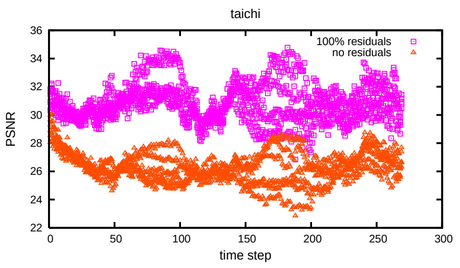


Fig. 5. PSNR values from the tai-chi motion. Each point in the plot indicates a PSNR value from a rendered image. For each time step, there are 12 points, which indicate 12 images rendered from two points with 100% and 0% residuals.

The quality of our compression method was measured as the difference between the point data before and after the model-driven compression. The "peak signal-to-noise ratio" (PSNR) of the images rendered from uncompressed and compressed point data was computed. In our experiments, two sets of images (one for each point set) are rendered from six (60 degree separated) camera views in each frame. We compare the reconstruction quality by computing PSNRs w.r.t the uncompressed data. Typical PSNR values in image compression are between 20 and 40 dB. We considered three compression levels, i.e., compression without residuals and with 50% and with 100% residuals. We see that encoding residuals indeed generate better reconstruction by 4 dB as compared to the compression without residuals. Figure 5 shows that considering residuals always produces better reconstructions in all frames. Another important observation is that the compression quality remains to be the same (around 30 dB) during the entire motion. Figure 6 shows that the difference between the uncompressed and compressed frame is more visible when the prediction residuals are not considered.

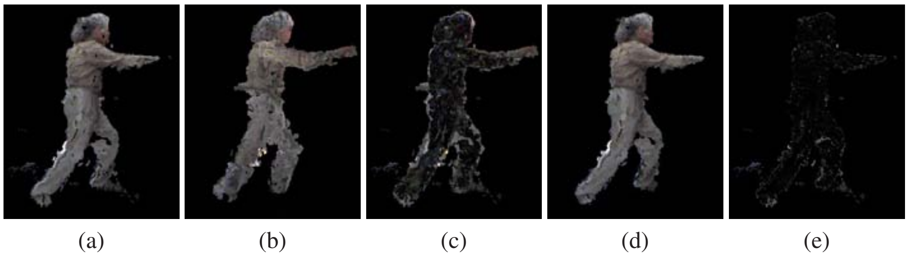


Fig. 6. Reconstructions from the compressed data and their differences with the uncompressed data. (a) Uncompressed data. (b) Compressed without residuals. (c) Difference between (a) and (b). (d) Compressed with residuals. (e) Difference between (a) and (d).

4.2 Compression Ratio

The analysis of different compression ratios showed that our model-driven compression method can achieve 50:1 to 5000:1 compression ratios (Table 2). As we have shown earlier, our compression method can provide different compression ratios by varying the level of residuals considered during the encoding. Significantly higher compression ratio as compared to the other two methods tested is achievable due to a fundamental difference in encoding the data while maintaining reasonable reconstruction quality

Table 2. Compression ratio. Both Yang et al.'s [2] and H.264 (we use and implementation from [22]) compression methods took the color and depths images as their input and output. The compression ratio of H.264 reported in this table is obtained using 75% of its best quality. We use jpeg and png libraries to compress color and depth residuals, respectively.

motion		**dancer 1**	**dancer 2**	**student**	**tai-chi master**
size before compression		142.82 MB	476.07 MB	1.14 GB	988.77 MB
compression ratio	Yang et al. [2]	11.36	11.73	10.23	14.41
	H.264 [22]	64.04	53.78	32.04	49.58
	no residuals	1490.31	3581.52	5839.59	5664.55
	25% residuals	195.62	173.52	183.80	183.82
	100% residuals	66.54	55.33	60.29	61.43

(Fig 6). Both, Yang et al.'s and H.264 algorithms, are image (or video)-based compressions, which take color and depth images as their input and output. Our model-driven compression, however, converts the color and depth images to motion parameters and prediction residuals. Despite high quality and high compression ratio of H.264 algorithm, the processing cannot be performed in real time for the amount of data that we considered in this work.

5 Conclusion

In this paper we have presented a skeleton-based data compression aimed for the use in tele-immersive environments. Data produced by the full-body 3D stereo reconstruction requires high network bandwidth for real-time transmission. Current implementation of the image/video-based compressor [2] in our tele-immersion system only allows transmission of data with the rate of 5 or 6 fps. Using model-based compression techniques can significantly reduce the amount of data transfer between the remote TI sites.

Using the real time (10+ fps) motion estimation technique described in this paper, we can compress data by converting point cloud data to a limited number of motion parameters while the non-rigid movements are encoded in a small set of regular grid maps. Prediction residuals are computed by projecting the points associated with each link to a regular 2D grid embedded in a cylindrical coordinate system defined by the skeleton link. Our experiments showed that our compressor provides adjustable high compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality with peak signal-to-noise ratio from 28 dB to 31 dB.

References

1. Lanier, J.: Virtually there. Scientific American 4, 52–61 (2001)
2. Yang, Z., Cui, Y., Anwar, Z., Bocchino, R., Kiyanclar, N., Nahrstedt, K., Campbell, R., Yurcik, W.: Real-time 3d video compression for tele-immersive environments. In: MMCN 2006. Proceedings of SPIE/ACM Multimedia Computing and Networking, San Jose, CA, ACM Press, New York (2006)

3. Kalra, P., Magnenat-Thalman, N., Moccozet, L., Sannier, G., Aubel, A., Thalman, D.: Real-time animation of realistic virtual humans. IEEE Computer Graphics and Applications 18, 42–56 (1998)
4. Mulligan, J., Daniilidis, K.: Real time trinocular stereo for tele-immersion. In: Proceedings of 2001 International Conference on Image Processing, Thessaloniki, Greece, pp. 959–962 (2001)
5. Baker, H., Tanguay, D., Sobel, I., Gelb, D., Gross, M., Culbertson, W., Malzenbender, T.: The coliseum immersive teleconferencing system. In: Proceedings of International Workshop on Immersive Telepresence, Juan-les-Pins, France (2002)
6. Wrmlin, S., Lamboray, E., Gross, M.: 3d video fragments: dynamic point samples for real-time free-viewpoint video. Computers and Graphics 28, 3–14 (2004)
7. Jung, S., Bajcsy, R.: A framework for constructing real-time immersive environments for training physical activities. Journal of Multimedia 1, 9–17 (2006)
8. Patel, K., Bailenson, J.N., Hack-Jung, S., Diankov, R., Bajcsy, R.: The effects of fully immersive virtual reality on the learning of physical tasks. In: Proceedings of the 9th Annual International Workshop on Presence, Ohio, USA, pp. 87–94 (2006)
9. Piccardi, M.: Background subtraction techniques: a review. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Hague, Netherlands, pp. 3099–3104. IEEE Computer Society Press, Los Alamitos (2004)
10. Zhao, W., Nandhakumar, N.: Effects of camera alignment errors on stereoscopic depth estimates. Pattern Recognition 29, 2115–2126 (1996)
11. Zhang, D., Nomura, Y., Fujii, S.: Error analysis and optimization of camera calibration. In: IROS 1991. Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems, Osaka, Japan, pp. 292–296 (1991)
12. Tsai, R.: A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. IEEE Journal of Robotics and Automation RA3, 323–344 (1987)
13. Zlib: Compression library (2005)
14. Besl, P., McKay, N.: A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14, 239–256 (1992)
15. Herda, L., Fua, P., Plankers, R., Boulic, R., Thalman, D.: Skeleton-based motion capture for robust reconstruction of human motion. In: Proceedings of Computer Animation Conference, pp. 77–93 (2000)
16. Theobalt, C., Magnor, M., Schuler, P., Seidel, H.: Multi-layer skeleton fitting for online human motion capture. In: VMV 2002. Proceedings of 7th International Workshop on Vision, Modeling and Visualization, Erlangen, Germany, pp. 471–478 (2002)
17. Aggarwal, J.K., Cai, Q.: Human motion analysis: a review. Comput. Vis. Image Underst. 73, 428–440 (1999)
18. Gavrila, D.M.: The visual analysis of human movement: a survey. Comput. Vis. Image Underst. 73, 82–98 (1999)
19. Lien, J.M., Bajcsy, R.: Skeleton-based compression of 3-d tele-immersion data. In: ICDSC 2007. Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras, IEEE Computer Society Press, Los Alamitos (2007)
20. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14, 239–256 (1992)
21. Wiegand, T., Sulivan, G., Bjntegaard, G., Luthra, A.: Overview of the h. 264/avc video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 13, 560–576 (2003)
22. Adobe: Qicktime 7.0 h.264 implementation (2006)

A CUDA-Supported Approach to Remote Rendering

Stefan Lietsch and Oliver Marquardt

University of Paderborn
Paderborn Center for Parallel Computing
`slietsch@upb.de`,
`marquard@upb.de`

Abstract. In this paper we present the utilization of advanced programming techniques on current graphics hardware to improve the performance of remote rendering for interactive applications. We give an overview of existing systems in remote rendering and focus on some general bottlenecks of remote visualization. Afterwards we describe current developments in graphics hardware and software and outline how they can be used to increase the performance of remote graphics systems. Finally we present some results and benchmarks to confirm the validity of our work.

1 Introduction

With today's ever increasing computational power and the possibilities to simulate and visualize more and more complex systems, it is obvious that efficient technologies are needed to make the results of such simulations and visualizations available for a broad audience. Remote rendering and remote visualization are techniques to fulfill this need for visual data. Since the early 90s researchers work on the topic of transporting the rendered images or visual data to the clients of remote users to let them analyze and work with this data. There are many systems that focus on several different aspects of remote rendering / visualization. Some allow comfortable administration of remote servers and others focus on interactively showing remote users images that where rendered on powerful workstations. But all of those systems need to deal with vast amounts of data that needs to be transferred. This is because visual data is mostly pixel or voxel based. Therefore even single images which are displayed for only a fraction of a second can be consist of one or more Megabytes of data. To provide a smooth animation, 30 or more frames are needed every second. All have to get from the GPU of the server, over the network, to the GPU and finally to the display of the user. There are already many approaches to minimize the amount of data that needs to be transferred for example through prefetching certain data to the client or by introducing level of detail mechanisms and compression of all kind. But there is still no solution on how to enable users to interact with a remote system as if it was local. However recent developments in graphics hard- and software technology open up new possibilities to access, compress and transfer visual data on and to the graphics cards. The introduction of PCI Express for Graphics (PEG) for example cleared the way to efficiently download rendered images from the graphics card and send it over the network. This was a serious improvement over the asynchronous AGP-Bus used until then, which only granted very slow access to the relevant memory areas on the card. With the so called *Frame Buffer Extension* and the *Render Buffer Objects* of the OpenGL 2.0 specification [1] efficient methods to enable offscreen rendering and the read-back of rendered images were introduced. Besides, the only recently published CUDA framework by NVIDIA [2] opens up new ways on also using graphics hardware for compression or filtering techniques to reduce the size of the data that needs to be transferred already on the graphics card. An example of how all these techniques could be used to improve the abilities of remote rendering is given in this paper.

2 Remote Rendering/Visualization - An Overview

There are different classes of remote visualization for different tasks. In the following we will introduce three classes that group the most common existing systems. This should help to understand what the problems of the different classes are, and to which class the approach proposed in this paper belongs.

2.1 Client-Side Rendering

Systems where the application runs on a server but the graphics data (polygon meshes, textures, volumes) is sent to the client and rendered there belong to this class. The best known system in this class is a remote X-Server which allows to start X-based applications on a remote server as if they where local. The main disadvantage, however, is that the rendering power of the server system is not used at all while all rendering work is done by the client. This is fine for small desktop applications but insufficient for complex visualization applications that require certain rendering power. Another popular representative of this class is the Chromium [3] framework which is able to transfer the whole OpenGL stream from a server to a client. This is only one feature of Chromium but has the same problem as the remote X-Server, namely it does not use the servers rendering power. Chromium is a very flexible framework and is not only designed to do remote rendering. It also offers a lot of features to support all kind of distributed and parallel rendering tasks. But for this work we only consider the remote rendering capabilities and classify them as Client-Side Rendering.

2.2 Server-Side Rendering for 2D and Administration

This class contains all systems that are mainly used for server administration and remote control. They render the images on the server side, compress them and, send them to the client where they can be viewed with simple viewers. They also work with low bandwidth connections and mostly show the whole desktop of the remote server. They perform well with simple 2D and little interactive applications (e.g. administration and settings dialogs) but they are insufficient for highly interactive 3D applications such as a driving simulator or an interactive 3D viewer. The most popular representatives of this class are the Virtual Network Computing Framework (VNC) [4] and Microsoft's Terminal Service architecture [5]. Both offer possibilities to adapt the remote visualization to the available bandwidth and client device. However they are not optimized to achieve high frame rates and low latency for interaction.

2.3 Server-Side Rendering for 3D Application (with and Without Transparent Integration)

The third class of systems is specialized on displaying interactive 3D applications on remote clients. The server (or servers) with a lot of graphics processing power renders a 3D application (e.g. OpenGL-based). Every frame is read back, compressed (lossy or lossless) and send to a client. Mostly, only the window of the 3D application is processed, to save bandwidth and processing power. Popular systems of this class are the SGI Viz-Server [6] and VirtualGL [7]. All of these systems are optimized to provide high frame rates with the available bandwidth. This is supported by the possibility to choose different resolutions and compression methods as well as certain level of detail mechanisms. Most systems allow a transparent usage of existing applications (mostly OpenGL-based). This guarantees a comfortable and universal utilization. On the opposite, there are systems that are tightly coupled to a certain application to further increase performance by adjusting the rendering process dynamically to fit the needs of remote visualization. The Invire prototype presented in this paper belongs to this class and will support both transparent and non-transparent integration. Furthermore it utilizes advanced hard- and software mechanisms to achieve high performance remote visualization.

3 Advances in Computer Graphics Hard- and Software

As mentioned before graphics hardware significantly changed in the last decade. The GPUs evolved from highly specialized and slowly clocked graphic processors to highly parallel, multi-purpose, fast co-processors which in some tasks outperform the CPU by far. This fact inspired a lot of programmers to try to port their application on the graphics hardware to gain significant performance increases. However the main problem was that the general purpose applications needed to be transferred into a graphics domain since the graphics cards could only be programmed through graphics APIs such as OpenGL or DirectX. There where efforts to develop more general APIs, mainly driven by the GPGPU[1] consortium. However one of the most important step towards general usage of graphics hardware was the introduction of the Compute Unified Device Architecture (CUDA) by NVIDIA in 2006 (see [2]).

[1] www.gpgpu.org

3.1 CUDA

CUDA is a combination of software and hardware architecture (available for NVIDIA G80 GPUs and above) which enables data-parallel general purpose computing on the graphics hardware. It therefore offers a C-like programming API with some language extensions. The architecture offers support for massively multi threaded applications and provides support for inter-thread communication and memory access. The API distinguishes between *host* and *device* domains and offers access to fast caches on the device side. The implemented method of thread partitioning allows the execution of multiple CUDA applications (*kernels*) on one GPU. Each *kernel* on the host device has access to a *grid* of thread *blocks*. A block consists of a batch of threads that can be synchronized and is organized by one-, two- or three-dimensional IDs. This allows to uniquely identify each thread and assign tasks to each thread. All threads inside one block have access to a fast shared memory space to exchange data.

Another feature of the CUDA architecture is the interoperability with graphic APIs (OpenGL and Direct3D) which allows to use, for example, rendered images as input to CUDA kernels. Since this data already resides on the graphics device it only needs to be copied on the device to be processed by CUDA. This offers great possibilities for e.g. online image compression which is one topic of this paper (see section 4.3).

3.2 Render Buffer Objects

Another important technique in this field is the introduction of Frame Buffer and Render Buffer Objects in the OpenGL 2.0 specification [1]. They replace the slow pbuffer mechanisms for off-screen rendering. This technique offers, among others, a fast way to render to specified memory regions other than the framebuffer. This can be used e.g. to implement rendering servers for a remote rendering framework as proposed in this paper.

4 Invire - A Concept for an Interactive Remote Visualization System

In this section we introduce the concept for a new system called Invire (INteractive REmote VIsualization). It belongs to the third class of remote rendering systems and focuses on high interactivity, maximized performance and best visualization results. We therefore utilized the new advances in graphics technology described in chapter 3 to improve compression and readback performance. The goal was to achieve adequate frame rates (above 20 FPS) and unrestricted interactivity for the remote visualization of high-resolution (1000x1000 pixels and above) 3D applications. We choose to use lossless compression techniques first to maximize graphics quality and because of their high potential for parallelization. We also work on integrating lossy compression methods which allow to guarantee a fixed bandwidth utilization. The main limitation, however, still is the network connection between server and client. Uncompressed image data is to large to be transferred over current internet or even LAN connections. An example shows that to achieve 25 FPS for a remote visualization with a resolution of 1000x1000 pixels we would need a network connection that has a bandwidth of 100 MB/s: $bandwidth = size\,of\,one\,frame * frames\,per\,second = (1000*1000*4\,\frac{byte}{frame})* 25\frac{frame}{second} = 100\frac{MByte}{second}$.Even Gigabit Ethernet is limited to about 80 MB/s in practical usage. That leads to the conclusion that a compression rate of more than 0.1 is needed to realize our goal on a system with a Fast Ethernet connection (100 MBit/s).

4.1 The Architecture

The overall architecture of Invire (as shown in Fig. 1) is kept simple to maximize the pure remote visualization performance. The server side offers an *Invire Plugin* which can easily be integrated in existing OpenGL applications. This allows the passing of parameters from the host application to Invire. We also work on a transparent integration possibly on the basis of VirtualGL. The *Invire Plugin* reads the current OpenGL context into a Renderbuffer Object (see section 3) which is passed to the *Compression* facility. This part of the software is implemented as modular collection of compression algorithms that can be exchanged arbitrarily. This allows us to compare the different methods and eventually combine them to achieve higher compression rates. After the compression of a rendered frame, it is passed to the *TCPServer* where a header is generated. The header contains information about the image size, compression. Afterwards header and compressed data is send to the *Invire Client*. It receives the compressed frame and passes it to the *Decompression* facility together with the information from the header. After decompression the frame is displayed by the client.

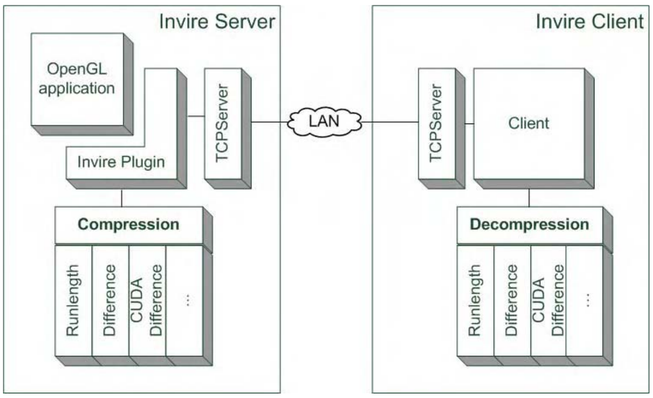


Fig. 1. Cuda memory and thread concept

4.2 Image Transfer

To transfer the images a TCP socket connection is established between server and client. This socket remains active until either client or server cancels the transfer. The advantage of TCP sockets is that the correct order and the integrity of the image data is guaranteed. After a socket connection is established the protocol overhead is minimal and allows a good utilization of the available bandwidth.

4.3 Image Readback and Compression

The most important parts of Invire are the readback and compression methods. We implemented a simple run-length (RLE) and difference encoding using standard programming techniques to have a basis for comparison (for more information on data and image compression see [8]. We shortly present the two basic compression methods and afterwards describe our CUDA-based difference compression algorithm in detail. The use of relatively simple and lossless compression methods has several reasons. First of all, they are very fast since both RLE and difference compression only need to go over the input once. Additionally the difference compression has a very good potential for parallelization as shown in the following. However, the main problem of lossless compression is that the achievable compression rate strongly depends on the input frame. If it is highly arbitrary in terms of consecutive colors or frame-to-frame difference then compression ratios can even be above 1.0. Nevertheless, typical visualization data (CAD, scientific visualization, VR) is often very regular in one or both terms. Therefore one can expect good compression rates for this application. There are also more sophisticated compression methods (e.g. statistical such as Huffman Coding or dictionary based such as LZ) but they cannot meet the requirements of interactive applications because of their comparably long run times or high memory consumption.

Run-Length Encoding. After reading back the image data of the current context to a given location in (host) memory, the run-length algorithm goes over the array, counts consecutive pixels with same color values and writes the sum followed by the actual color information into a new array. The decompression can be done by writing the amount of pixels with the same color in a new array consecutively. Afterwards it can be displayed by the OpenGL function `glDrawPixels()`. The RLE algorithm can encode and decode n pixels in $O(n)$ time.

Difference Compression with Index. Another basic technique for lossless image compression is the difference compression. The current and the last frames are compared pixel wise and only the pixels that are different are saved. Additionally the position of the pixels that changed is needed to decompress the current frame. This can be most efficiently done by an index which maps one bit to every single pixel of a frame. If the bit is 1 the pixel has been changed and the saved pixel value at the position *#of preceding 1s in index* is needed to update the pixel of the last frame. This requires O(n) time to compress and decompress n pixels.

Difference Compression with Index Using CUDA. The difference compression with index method described before is very suitable for parallel execution (on k threads with $n > k$ pixels). Especially creating the index, as well as the copying of the pixel data can be done in $O(n/k)$ time. Therefore we decided to implement this method in parallel and choose to use the CUDA architecture. It is able to process the data directly on the graphics hardware and allows highly parallel execution. Since CUDA offers several SIMD multi processors we decided to split the frames into $m = \frac{n}{k}$ blocks which can be processed independently. This helps to optimize the workload on the graphics hardware. Each block consists of k threads which are working in parallel on one multiprocessor and which have access to a fast, shared memory. Each thread processes one pixel which is assigned to it by its thread and block index (thid and bid). Fig. 2 shows the three main steps of the algorithm. In the first step the index is generated by simply comparing corresponding pixels of the last and the current frame. When all threads have finished, a parallel compaction method (the second step) based on the stream compaction algorithm by [9] is invoked on the shared memory. This algorithm requires $O(\log k)$ time in parallel to compute the amount of empty spaces to its left for every item of the s_result array. It needs to run for all m blocks. With this information the pixels can now be stored in an array without empty spaces which is copied to a position in the global result array which is determined by the block index (bid). Additionally the number of stored pixels is written to a global array (g_changedPixels). After all blocks finished the second step, the g_changedPixels array is used to compute the absolute position in memory for each blocks partial result (step 3). This is also done in parallel based on the scan algorithm presented by [10] which sums up all items to the left of the current value in $O(\frac{m}{k} * \log \frac{m}{k})$ time in parallel. Finally the partial results of the blocks are copied to the calculated memory locations and then the final compressed frame and the index are readback from the graphics hardware to be send to the client. The decompression at the client side is done sequentially as described above or can be parallelized equal to the compression using CUDA if available. Finally the whole algorithm can be run in

$$O(2 * (m) + m * \log k + \frac{m}{k} * \log \frac{m}{k})$$

time in parallel with $n =$ number of pixels, $k =$ number of threads, $m =$ number of blocks and $n = m * k$.

5 Benchmarks and Results

To proof the theoretical concept and to compare the described compression algorithms we prototypically implemented the Invire system and tested it in a sandbox environment. The server part runs on a computer equipped with a CUDA ready Geforce 8800 GTS graphics card by NVIDIA. It has 12 multiprocessors, a wrap size[2] of 32. That leads to $k = 12 * 32 = 384$ threads that can

[2] Number of threads that are executed in parallel on one multiprocessor.

```
divide frame in blocks of 256 pixels and
copy them to shared memory (s_current);
for all blocks do parallel
    for all threads do parallel
        if(compare(s_current[thid],s_last[thid]))
            s_index[thid]=0;
        else
            s_index[thid]=1;
            s_result=s_current[thid];
    g_index[bid*bw+thid]=s_index[thid];
    switch s_current and s_last;
    syncthreads;

    for all threads do parallel
        compaction(s_result);
    for all threads do parallel
        g_result[bid*bw + thid] = s_result[thid];
        g_changedPixels[bid] = #changed pixels;
syncblocks;

    for all threads do parallel
        computeAbsoluteMemoryPosition(g_changedPixels);
        result: g_AbsoluteMemoryPosition[];
syncblocks;

    for all threads do parallel
    g_compResult[g_AbsoluteMemoryPosition[bid]+thid] = g_result[thid];
syncblocks;

send(g_index,g_compResult);
```

Fig. 2. Pseudocode of the parallel "difference with index" compression. Abbreviations: bid = Block index, bw = Block width, thid = Thread Index, s_ = shared, g_ = global.

run concurrently. As testing scenario we chose two OpenGL based applications (see Fig. 3) . The first is a simple rotating teapot on black ground and the second is a driving simulator called Virtual Night Drive (VND) [11]. The VND is specialized on simulating automotive headlights at night and uses the shaders of the graphics card to calculate the luminance intensity pixel wise.The first teapot application is highly regular and has a uniform black background, whereas the VND is relatively arbitrary through its textured and lighted scene and its unpredictable movement. The charts in Fig. 4 show some interesting results. For both test application we measured the average frame rate for various resolutions and compression methods. Fig. 4A) shows the teapot application with resolutions from 16*16 pixels up to 1024*1024 pixels. For resolutions below 256*256 the overhead for compressing and decompressing the frames is obviously to big so that no compression method performs better than just sending uncompressed images. But from 256*256 on the size of the transferred images is the limiting

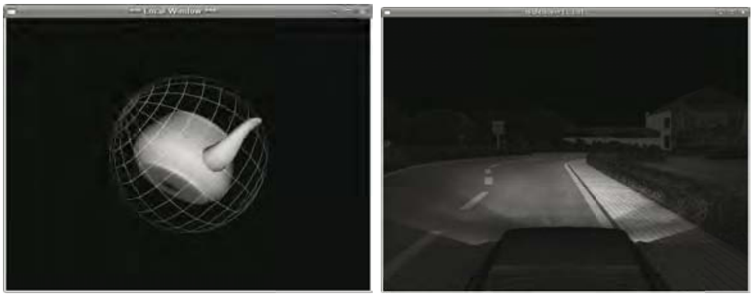


Fig. 3. Test cases: simple teapot, Virtual Night Drive with headlight simulation

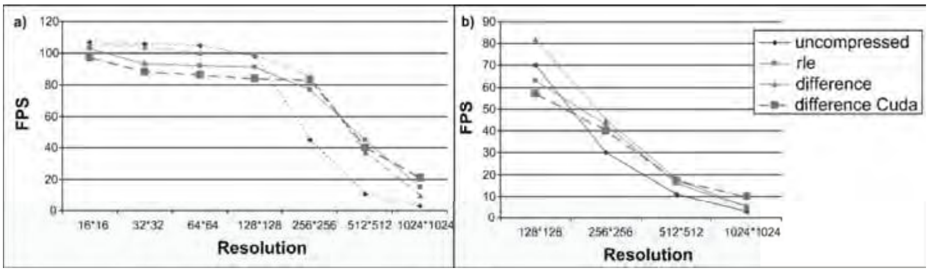


Fig. 4. Benchmarking results with A) Teapot and B) Virtual Night Drive

factor. The RLE and the difference without CUDA method perform nearly similar and are a bit faster than the version with CUDA up to the resolution of 512*512. There the difference coding with CUDA takes the lead and outperforms the other compression methods by a factor of about two (RLE 14 FPS, difference 10 FPS and difference with CUDA 21 FPS). This can be explained by the computational overhead the CUDA-based algorithm introduces (mainly the compaction methods). But for high resolutions this overhead amortizes and we can nearly double the frame rate. The VND application shows similar results. The interesting thing here is, that all algorithms produce slower frame rates because of the poor compressibility of the input data. But still the CUDA-based approach provides the best results for the highest resolution. In comparison with another remote graphics system of the same class -VirtualGL- Invire performed slightly worse (about 20% fewer FPS in the VND application) but that can be explained of the use of lossy compression versus lossless compression techniques.

6 Conclusion and Outlook

In this paper we presented a remote rendering system that takes advantage of recent advances in computer graphics hard and software. We could show that the speedup of a parallel compression method surpasses the resulting computational overhead for high resolutions (1024*1024 pixels and above). However, the implementation that was used for the performance benchmarks still is in a prototypical stage. There are some programming optimizations to make and especially the integration of high quality lossy compression techniques promises good results.

References

1. Segal, M., Akeley, K.: The OpenGL Graphics System: A Specification Version 2.0. Silicon Graphics (2004)
2. NVIDIA: NVIDIA CUDA - Compute Unified Device Architecture. Website (2007), `http://developer.nvidia.com/object/cuda.html`
3. Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P., Klosowski, J.: Chromium: A stream-processing framework for interactive rendering on clusters. ACM Transactions on Graphics 21, 693–702 (2002)
4. Richardson, T., Stafford-Fraser, Q., Wood, K., Hopper, A.: Virtual network computing. IEEE Internet Computing 2(1), 33–38 (1998)
5. Microsoft: Microsoft Windows Server 2003 Terminal Services. Website: (2007), `http://www.microsoft.com/windowsserver2003/technologies/terminalservices/default.mspx`
6. SGI: SGI OpenGL Vizserver - Visual Area Networking. Website (2007), `http://www.sgi.com/products/software/vizserver/`
7. VirtualGL: The VirtualGL Project. Website (2007), `http://www.virtualgl.org/`
8. Salomon, D.: Data Compression: The Complete Reference, 3rd edn. Springer, Heidelberg (2004)
9. Horn, D.: Stream Reduction Operations for GPGPU Applications. In: Pharr, M., Fernando, R. (eds.) GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation, pp. 573–583. Addison-Wesley, Reading (2005)
10. Harris, M.: Parallel Prefix Sum (Scan) with CUDA. Website (2007), `http://developer.download.nvidia.com/compute/cuda/sdk/websiteprojects/scan/doc/scan.pdf`
11. Lietsch, S., Berssenbruegge, J., Zabel, H., Wittenberg, V., Eikermann, M.: Light simulation in a distributed driving simulator. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Nefian, A., Meenakshisundaram, G., Pascucci, V., Zara, J., Molineros, J., Theisel, H., Malzbender, T. (eds.) ISVC 2006. LNCS, vol. 4291, pp. 343–353. Springer, Heidelberg (2006)

Dynamic Balance Control Following Disturbance of Virtual Humans

Cyrille Collette[1], Alain Micaelli[1], Pierre Lemerle[2], and Claude Andriot[1]

[1] CEA/LIST, 18 route du Panorama, BP6, FONTENAY AUX ROSES, F-92265 France
{cyrille.collette,alain.micaelli,claude.andriot}@cea.fr
[2] INRS/MSMP, Avenue de Bourgogne, BP27, VANDOEUVRE, F-54501 France
pierre.lemerle@inrs.fr

Abstract. Subject to disturbance, a human can carry out many balance strategies, changing its posture. Virtual human animation is a challenging problem. In the present paper, we introduce a new dynamic balance control of virtual humans with multiple non coplanar frictional contacts. We formulate a constrained optimization problem (Quadratic Programming). Our virtual human can autonomously manage its balance without following imposed trajectories, in all kind of environments (floor, stairway) while being disturbed by external forces. In contrast to classical methods based on ZMP, it can use its hands to keep its balance by pressing an inclined wall. In every case, it fits its posture to ensure the best balance.

1 Introduction

1.1 Problem Statement

Subject to disturbance, a human can carry out many balance strategies, changing its posture. Since the kinematic structure of a Virtual Human (VH) is similar a human one, a VH is expected to have the same behavior in a virtual environment. The computer graphics industry knows how to produce outstanding animation involving motion capture or graphic animators. Existing control schemes of VH are very limited. Generating consistent motions and behaviors which approximately look credible is a challenge.

1.2 Related Work

V.B. Zortan *et al.* [1] introduced a new technique for incorporating unexpected impacts into a motion capture-driven animation system. But motion capture data are pre-required. P. Faloustos *et al.* [2] proposed composable controllers for physics-based character animation. But each controller is specific to a basic action. The authors do not deal with a unified formulation of physical balance based on stability criteria. Balance with complex and unpredictable contact configuration is not taken into account.

Two usual stability criteria for a VH standing on a horizontal ground are the Center of Mass (CoM) and Zero Momentum Point (ZMP) criteria. In quasi-static situations, a VH will remain static if its CoM projects vertically inside the convex hull of the contact points. In the dynamic case, the ZMP takes into account inertial and Coriolis wrenches.

A VH is able to realize a specified movement if the ZMP projects vertically inside the convex hull of the contact points. However, in case of irregular ground, CoM and ZMP are not adapted.

Some authors work on more global balance criteria in complex environments. Recently, H. Hirukawa *et al.* [3] proposed a universal stability criterion of two-legged robots having multiple non-coplanar contacts. However, it does not deal with hand contacts, disturbance and VH redundancy.

T. Bretl *et al.* [4] [5] presented a general framework for planning the quasi-static motion of a three-limbed climbing robot in vertical natural cliff. To prevent the robot from falling as it moves a limb to reach a new hold, the algorithm exploits friction at supporting holds and adjusts the robot's internal Degrees of Freedom (DoF). Y. Or and E. Rimon [6] characterize robust balance in a 3D gravity environment with multiple non-coplanar contacts. The evolution area of the CoM is a convex vertical prism. It is a global geometrical approach in the static case and can be applied to VH balance. But it is difficult to make use of this analytic approach. However, both approaches are only static; dynamic simulation and VH posture are not taken into account.

Since the 80's, L. Sentis *et al.* [7] used projection methods which have been fully tried and tested. However, contact considered in control law have not a physical meaning. At last, complex balance are not considered. This method presents passivity issues [8].

In terms of control computation, optimization technics have been studied. P.B. Wieber [9] proposes an interesting optimization formulation for walking robot problems. Contact forces are taken into account in the control law. However, his formulation is only applied to walking stability and do not deal with complex balance. Recently, Y. Abe *et al.* [10] proposed an interactive multi-objective control with frictional contacts. However, complex stable balance strategies are not entirely treated.

1.3 Contribution

In this paper, we propose an original framework for motion control of a VH with multiple non coplanar frictional contacts. The following can be considered as key features of this control architecture:

– **Physically meaningful assumptions:** We define physically meaningful assumptions on the human motion which are the core of the control law since they are used as constraints for the minimization problem.
– **Dynamic QP:** We take into account all forces effecting the VH. The solution of the quadratic minimization problem (QP: Quadratic Programming) induces for sure a realizable set of desired forces acting on the VH. However, in the control law only, computed motors jointed torques will be used.
  In physical simulation, integrating the resulting body accelerations yields the VH posture; no position is clamped a priori.
– **Static QP:** A second QP (static QP) is used beforehand the dynamic QP in order to compute jointly an admissible set of contact friction forces and a robust CoM. The robust CoM is used into the dynamic QP.

- **All constraints are considered simultaneously:** Our control scheme does not require priorities of individual control balance and tasks. It avoids tedious parameter adjustment which usually is highly task dependent. Our control schemes turns out to perform well for very different VH disturbances and tasks, yielding a general human-like motion behavior without requiring intermediate postures or even joint trajectories.
- **Robust balance control:** We use a direct and constructive method which emphasizes the computation of instantaneous reference control values in order to maintain a given posture. Due to a heavy disturbance, VH may be led to an unstable configuration. In this case the constraints are still respected but the solution of the minimization problem, ie the computed CoM, does not correspond to a stable posture. Without any further precaution, the VH would fall down.
  This case can be easily detected and VH may change the balance strategy, for instance doing a step forward. This requires the virtual environment to be known by the VH.

In following section, we present the physical model used for the simulation engine. In section 3, we describe our new dynamic balance control of VH in frictional non coplanar multiple contacts virtual environment. Section 4 presents the first results and advanced balance control. Finally, section 5 summarizes the presented control architecture and indicates some possible future research directions.

2 Modeling

2.1 Dynamic Simulation

Our VH is a set of articulated branches of rigid bodies, organized into a highly redundant arborescence. Our VH consists of 32 joints. The skeleton is modeled as a multibody system. The root body of the VH tree is the thorax. This root has 6_{root} DoF and is not controlled. We decided to use human data for modeling our VH [11][12][13]. Our dynamic model elaboration comes from J. Park [14]. The dynamics of the robot is described in terms of its joint coordinates, q.

$$\tilde{M} \cdot (\dot{T} - G) + N \cdot T = L \cdot \tau + C^T \cdot W \tag{1}$$

For this system of $n_{\text{dof}} + 6_{\text{root}}$ equations with n_{dof} the number of degrees of freedom, τ is the set of generalized joint torques, $\tilde{M}$ is the inertia matrix, $N \cdot T$ represents the inertial and Coriolis forces and G represents the gravity. Multiply by the matrix L, τ is expressed in generalized coordinates. The vectors of external forces W is the sum of contact forces and other external disturbance forces: $W = W_{\text{contact}} + W_{\text{disturbance}}$.

$$\tau = \begin{bmatrix} \tau_1 \ \ldots \ \tau_{n_{\text{dof}}} \end{bmatrix}^T \qquad G = \begin{bmatrix} 0 \ 0 \ -g \ 0 \ \ldots \ 0_{3+n_{\text{dof}}} \end{bmatrix}^T \qquad L = \begin{bmatrix} 0_{(6,n_{\text{dof}})} \ I_{n_{\text{dof}}} \end{bmatrix}^T$$

$\dot{T}$, T and X are respectively tree acceleration, velocity and position in generalized coordinates. C is the transformation matrix between T and velocity of all the bodies ($V_{\text{body}}^{R_{\text{body}}}$: twist of body expressed in its own frame R_{body}) [14].

$$\dot{T} = \begin{bmatrix} \dot{V}_{\text{root}} \\ \ddot{q}_1 \\ \vdots \\ \ddot{q}_{n_{\text{dof}}} \end{bmatrix} \qquad T = \begin{bmatrix} V_{\text{root}} \\ \dot{q}_1 \\ \vdots \\ \dot{q}_{n_{\text{dof}}} \end{bmatrix} \qquad X = \begin{bmatrix} X_{\text{root}} \\ q_1 \\ \vdots \\ q_{n_{\text{dof}}} \end{bmatrix} \qquad \begin{bmatrix} V_{\text{root}} \\ V_1 \\ \vdots \\ V_{n_{\text{dof}}} \end{bmatrix} = C \cdot T$$

2.2 Contact Simulation Modeling

Unilateral contacts seen as Coulomb frictional contacts are ruled by a non linear model *i.e.*: $|f_c^t| < \mu \cdot f_c^n$ with f_c^n, f_c^t respectively the normal and tangent contact forces and μ, the dry-friction factor.

2.3 Contact Control Modeling

As a linear formulation for our optimization problem is needed, we use a linearized Coulomb model. Like J.C. Trinkle *et al.* [15], we linearize contact cones into multifaceted friction cones in order to obtain linear constraints (Fig. 1). The linearized contact force of the k-*th* contact is denoted:

$$f^k_{c/\text{linearized}} = l f_c^k \cdot \xi^k \text{ with } \xi^k = \left[\xi_1^k \ldots \xi_{n_e}^k \right]^T \qquad (2)$$

with n_e, the number of edges, $l f_c^k$, the linearized friction cone and ξ^k the forces to every line direction of the k-*th* linearized friction cone. The contact forces computed by our control law must be inside the friction cone which means: $\forall i \in [1, n_e], \xi_i^k \geq 0$.

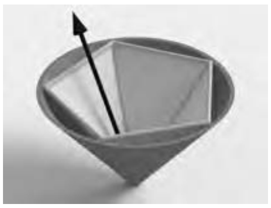


Fig. 1. Linearized friction cone (5 edges)

3 Control

3.1 General Outline

We use a robotic approach and more precisely joint control to handle VH dynamic. As in [9] [10], we formulate a constrained optimization problem (Quadratic Programming: QP). This method deals with a great number of DoF and solves simultaneously all constraint equations. We introduce the following notation, Y: unknown vector, Y^{des}: desired but not necessarily accessible solution, Q: quadratic norm and A, b, C and d matrices and vectors which express linear equality and inequality constraints.

$$\min \tfrac{1}{2} \|Y - Y^{\text{des}}\|_Q^2 \text{ such as } \begin{cases} A \cdot Y + b = 0 \\ C \cdot Y + d \geq 0 \end{cases} \qquad (3)$$

The VH control is based on two successive QP (Fig.2).

Firstly, in the static QP, we compute a consistent goal CoM position, robust with respect to homogenous distribution of desired contact forces, first studied by A. Rennuit [16]. We introduce the following notations: Y_1, Y_1^{des}, Q_1, A_1, b_1, C_1 and d_1. Static QP equations are detailed in section 3.3.

Secondly, in the dynamic QP, we compute the control torques in order to reach the CoM goal position. We introduce the following notations: Y_2, Y_2^{des}, Q_2, A_2, b_2, C_2 and d_2. Dynamic QP equations are detailed in section 3.4.

In the next subsections, we make a behavior assessment and then try to express them as a QP.

3.2 Behavior Assumptions

By its own joint activation, VH gets around and interacts with the environment. The joint control torques are saturated (Eq.11) so that our VH cannot apply unrealistic forces. Working hypotheses of VH are:

– *Known parameters:* State of its contacts and friction (μ) when it interacts. Position of its CoM and its goal CoM, the latest being stable in respect to its contacts layout (Subsection 3.3). Spatial location of environment (ex: ground, wall). Its dynamic model and gravity (Eq.7).
– *Unknown parameters:* Disturbance forces.
– *Behavioral model:* To keep stable contacts, contact acceleration must be null and contact forces must be inside the friction cones. If the contact is broken, the VH try to use the nearest environment to keep its balance. Moreover, it keeps its balance with the acceleration of gravity, tries to reach the goal CoM and adapts its posture.

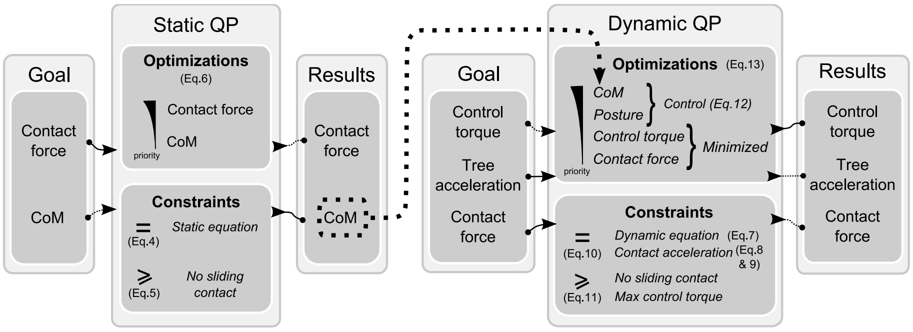


Fig. 2. Static and dynamic QP. Thereafter, only control torque result is useful in the physical simulation.

3.3 Static QP Formulation

In this section, we detail the static QP formulation of Fig. 2. Our VH is reduced to its CoM, subject to the acceleration of gravity and contact forces. The VH posture is not considered.

The unknowns are the CoM position x_G and the linearized contact forces ξ. They are expressed into a vector $Y_1 = \left[x_G \; \xi \right]^T$ of dimension $3 + n_c \cdot n_e$ with n_c, the number of contacts and n_e, the number of edges. From a contact forces distribution (generally, homogeneous for a standing posture) and a CoM position of our choice, we compute a solution which respect several constraints: the static balance equation and non-sliding contacts. With the gravity W_G^{scene} and contact W_C^{scene} applied to VH and expressed in the scene coordinates system, the static balance equation is $W_G^{\text{scene}} + W_C^{\text{scene}} = 0$. We can easily obtain:

$$W_G^{\text{scene}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -m \cdot g & 0 \\ m \cdot g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot x_G + \begin{bmatrix} 0 \\ 0 \\ -m \cdot g \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \Rightarrow \quad W_G^{\text{scene}} = E_1 \cdot x_G + E_2$$

We can compute a matrix E_3 such as $W_C^{\text{scene}} = E_3 \cdot \xi$. So:

$$A_1 \cdot Y_1 + b_1 = 0 \qquad \text{with} \qquad A_1 = \begin{bmatrix} E_1 & E_3 \end{bmatrix} \qquad \text{and} \qquad b_1 = E_2 \tag{4}$$

Contacts must not be sliding (Subsection 2.3) so $\xi \geq 0$. (diag: block diagonal matrix)

$$C_1 \cdot Y_1 + d_1 \geq 0 \quad \text{with} \quad C_1 = \text{diag}\begin{pmatrix} 0_{33} & I_{n_c \cdot n_e} \end{pmatrix} \quad \text{and} \quad d_1 = \begin{bmatrix} 0_{(3+n_c \cdot n_e, 1)} \end{bmatrix} \tag{5}$$

We establish different priorities between the optimized criteria by weighing Q_1.

$$\min \tfrac{1}{2} \|Y_1 - Y_1^{\text{des}}\|_{Q_1}^2 \quad \text{with} \quad Q_1 = \text{diag}\begin{pmatrix} Q_{x_G} & Q_\xi \end{pmatrix} \quad \text{and} \quad Y_1^{\text{des}} = \begin{bmatrix} x_G^{\text{des}} & \xi^{\text{des}} \end{bmatrix}^T \tag{6}$$

We group equations (4), (5) and (6) to solve the optimization constraints problem (Eq.3). x_G and ξ are computed but only x_G is used in the dynamic QP renamed x_G^{goal} (Eq. 12, Fig.2). This method allows us to find a static balance CoM location with multiple coplanar or non coplanar contacts. Moreover, by adjusting the weighing, we easily bring the problem to computing static balance CoM position in order to get the desired contact forces distribution.

In the next subsection, we give details of the dynamic QP.

3.4 Dynamic QP Formulation

In this part, we formulate previous behavior assumption (3.2) as a Dynamic QP (Fig.2). The goal is to compute a control torque τ to apply to the VH's joint. Tree acceleration in generalized coordinates $\dot{T}$ and linearized contact forces ξ are also unknown. They are expressed into a vector $Y_2 = \begin{bmatrix} \tau & \dot{T} & \xi \end{bmatrix}^T$ of dimension $2 \cdot n_{\text{dof}} + n_c \cdot n_e + 6$. The dynamic equation is:

$$L \cdot \tau - \tilde{M} \cdot \dot{T} + D_C^T \cdot \xi - N \cdot T + \tilde{M} \cdot G = 0 \tag{7}$$

Physical simulation (Subsection 2.1) computes L, $\tilde{M}$, N and G. D_C^T formulates ξ in generalized coordinates. Control law does not know disturbance forces, so $W_{disturbance} = 0$. Moreover, we impose a goal a_c^{goal} to the n_c-*th* contact acceleration a_c. We can formulate the velocity of the n_c-*th* contacts v_c as a linear function F of T: $v_c = F \cdot T \Rightarrow a_c = F \cdot \dot{T} + \dot{F} \cdot T$. This acceleration gives a command on $\dot{T}$:

$$a_c = a_c^{\text{goal}} \qquad \Rightarrow \qquad F \cdot \dot{T} = a_c^{\text{goal}} - \dot{F} \cdot T \tag{8}$$

For the k-*th* contact (according to subsection 3.2), depending on whether contact is active or not, we use the following contact acceleration commands:

$$\begin{cases} a_{c(k)}^{\text{goal}} = 0_3 \quad \text{(collision)} \\ a_{c(k)}^{\text{goal}} = k_c \cdot (x_{c(k)}^{\text{goal}} - x_{c(k)}) + \mu_c \cdot (v_{c(k)}^{\text{goal}} - v_{c(k)}) \quad \text{(free motion)} \end{cases} \tag{9}$$

with (k_c, μ_c): contact control gain; $(x_{c(k)}, v_{c(k)})$: position and velocity of contact k; $x_{c(k)}^{\text{goal}}$: projection of $x_{c(k)}$ to the nearest VH's environment and $v_{c(k)}^{\text{goal}} = 0_3$.

We group together equations (7) and (9) in the following constraint equalities system:

$$A_2 \cdot Y_2 + b_2 = 0 \quad \text{with} \quad A_2 = \begin{bmatrix} L & -\tilde{M} & D_c^T \\ 0 & F & 0 \end{bmatrix} \quad \text{and} \quad b_2 = \begin{bmatrix} -N \cdot T + \tilde{M} \cdot G \\ \dot{F} \cdot T - a_c^{\text{goal}} \end{bmatrix} \tag{10}$$

Contacts must be non-sliding (Subsection 2.3) so $\xi \geq 0$. Joint torques are saturated because of motor limitations: $|\tau| \leq \tau^{\max}$ so $\tau \leq \tau^{\max}$ or $-\tau \leq \tau^{\max}$.

$$C_2 \cdot Y_2 + d_2 \geq 0 \quad \text{with} \quad C_2 = \begin{bmatrix} -I_{n_{\text{dof}}} & 0 & 0 \\ I_{n_{\text{dof}}} & 0 & 0 \\ 0 & 0 & I_{n_c \cdot n_e} \end{bmatrix} \quad \text{and} \quad d_2 = \begin{bmatrix} \tau^{\max} \\ \tau^{\max} \\ 0 \end{bmatrix} \tag{11}$$

We want to minimize the torque and the linearized contact forces. Moreover, we express x_G^{goal} (computed in subsection 3.3) through the optimization criteria $\dot{T}^{\text{des}}$. We write the CoM acceleration a_G^{des} as a function of $\dot{T}$. As there is infinite VH postures with the same CoM, we add another posture criterion a_P^{des}. Hence, we obtain the following control:

$$\begin{cases} a_G^{\text{des}} = k_G \cdot \left(x_G^{\text{goal}} - x_G\right) + \mu_G \cdot \left(v_G^{\text{des}} - v_G\right) \quad \text{(CoM control)} \\ a_P^{\text{des}} = k_P \cdot \left(X^{\text{des}} - X\right) + \mu_P \cdot \left(T^{\text{des}} - T\right) \quad \text{(Posture control)} \end{cases} \tag{12}$$

with (k_G, μ_G): CoM control gain; (k_P, μ_P): Posture control gain; $v_G^{\text{des}} = 0_3$ and $T^{\text{des}} = 0_{6+n_{\text{dof}}}$.

Many studies [17][18] analyze standing posture and measure orientation and localization of the various body segments with respect to the gravitational vector. Using their results, we decided that VH tries to place its thorax above x_G^{goal} level, vertically oriented. At last, VH is controlled around an initial posture which can evolve during simulation. We formulate all these choices through X^{des}. Finally, $\dot{T}^{\text{des}}$ is a function (denoted f) of a_G^{des} and a_P^{des}. We give different priorities to the optimization criteria thanks to weighing Q_2.

$$\min \tfrac{1}{2} \|Y_2 - Y_2^{\text{des}}\|_{Q_2}^2 \quad \text{with} \quad Q_2 = \begin{bmatrix} Q_\tau & 0 & 0 \\ 0 & Q_{\dot{T}} & 0 \\ 0 & 0 & Q_\xi \end{bmatrix} \quad \text{and} \quad Y_2^{\text{des}} = \begin{bmatrix} \tau^{\text{des}} = 0_{n_{\text{dof}}} \\ \dot{T}^{\text{des}} = f(a_G^{\text{des}}, a_P^{\text{des}}) \\ \xi^{\text{des}} = 0_{n_c \cdot n_e} \end{bmatrix} \tag{13}$$

We group equations (10), (11) and (13) and solve this dynamic QP problem (Eq.3). τ, $\dot{T}$ and ξ are computed but only τ is used in the physical simulation (Fig. 2). In the next section, we present some results of physical simulation illustrating our control law.

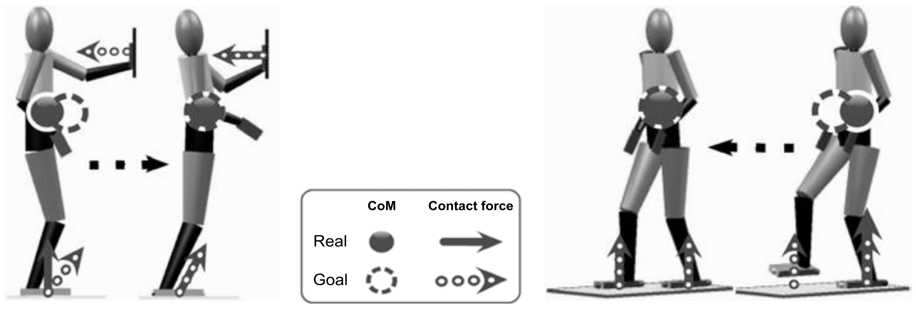
4 Results

“Arboris” developed by CEA/ISIR in Matlab (2006) is used for dynamic simulation environment. Each cone is linearized by 5 edges in order to keep a good compromise between precision and computation velocity in our control law.

4.1 Introduction of a New Interaction

Press on the Wall. The VH has to apply a force on the wall. It moves its CoM and adapts its posture to keep a robust static balance when it applies the desired force on the wall (Fig. 3(a)). Solution of static QP (x_G^{goal}) is explained in subsection 3.3. It is an example of a task in multiple non coplanar contacts.

Put Down the Second Foot. In this example (Fig. 3(c)), the VH initially has its left foot on the ground. In this posture, it is not balanced. Control law goal is to bring its right foot in contact with the nearest environment: the ground. In the same time, it anticipates its next support and puts its CoM forward in order to reach goal CoM position. This goal is computed considering the VH rests on its two feet. The goal CoM position needs the contact points location to compute CoM. This algorithm considers that feet contacts are established. Otherwise, the algorithm computes the contact projections on the environment (horizontal ground). It is important to handle these two aspects (i.e. its right foot and its goal CoM) at the same time in order to have a stable behavior.



(a) Press on the wall (b) Legend (c) Put down the right foot

Fig. 3. Virtual Human accomplishes its goals

4.2 Balance Control Following Disturbances

In the next example (Fig. 4 and 5), disturbance is applied on the ground during a short time (100 ms). The ground has a run of 6 inches and goes to an end stop violently. If it can, the VH balances and reaches a new balance posture. Control law does not explicitly know disturbance, so contacts can be lost or slide a few instants.

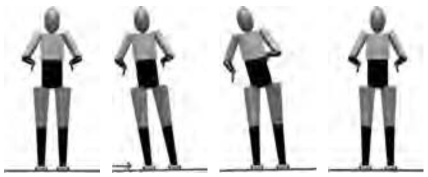


Fig. 4. Perturbation in the sagittal plane

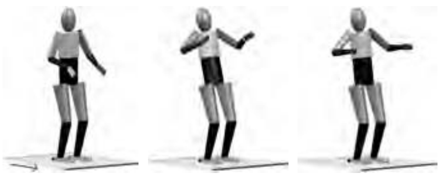


Fig. 5. Perturbation in the frontal plane

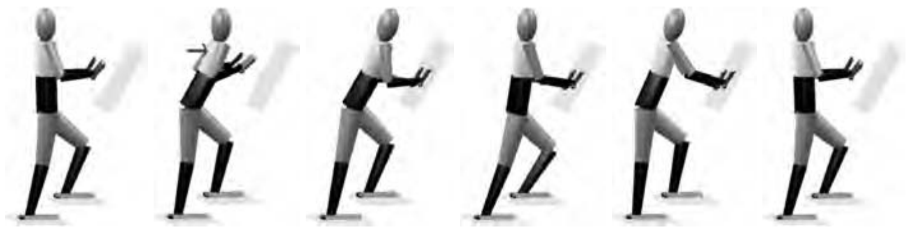


Fig. 6. Using arms to keep balance

4.3 Towards Multi-phase Balance Control

In these examples (Fig. 6 and 7), VH contacts are activated/deactivated as the simulation carries on. It is a more high-level control. We implement a two phase balance control. At the beginning of the scene, the VH is upright and its hand contacts are deactivated. On account of a disturbance force applied on its back, its CoM moves away from its goal position $||x_G^{\text{goal}} - x_G|| >$ dist (dist is an arbitrary distance). In the first case (Fig. 6), hand contacts are activated and try to come in touch with the environment. Thanks to its hands, the VH can maintain its balance. Finally, its hand contacts are deactivated in order to meet up with its starting posture. We note that it is a non-coplanar multi contact problem and the static QP of our algorithm allows us to stabilize the motion. The VH converges to a stable CoM and posture. In the second case (Fig. 7), contact accelerations of the right feet have a new goal so that VH makes one step forward.

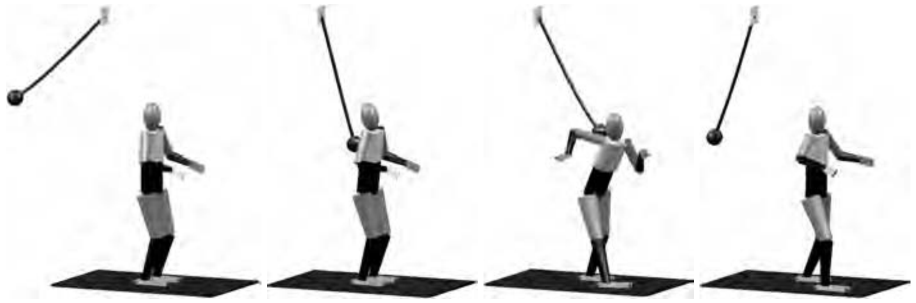


Fig. 7. One step forward

5 Conclusion

We introduce a new dynamic balance control of VH in multiple non coplanar frictional contacts. Thanks to constraints optimization algorithm (QP), the VH can be controlled in a large range of situations and is robust with the same set of parameters (control gain, weighing). With this control law, the VH's behaviors look like human reflex. We notice that this algorithm is adequate for all serial robots, for every contact configuration (not especially end-effector), and for a great number of DoF. Thanks to a higher-level control, a complex balance control can be achieved. Subject to disturbance, our VH can carry out many balance strategies, changing its posture and contacts autonomously.

We are currently planning to use this control architecture in an interactive demo. The basic algorithm is fast enough to be used real-time. Resolution of Dynamic QP is the most costly. Single-threaded C algorithm implement costs 2.6 ms (for an average complexity problem: Fig. 4). We use an Intel® Xeon® CPU 5130 @ 2.00 GHz, 2.00 Go RAM.

Establishment of a simple prehension model will allow us to extend our control law to more complex situations. Behaviors of VH will be validated thanks to bench test on human volunteers. We will study very simple behaviors using a force platform and motion capture. At last, we would like to work on predictive control [19].

Acknowledgments

The authors gratefully acknowledge the ANR and the RNTL PERF-RV2 project. The authors would like to thank Sébastien Barthélemy (ISIR/LRP) for his precious help.

References

1. Zordan, V.B., Majkowska, A., Chiu, B., Fast, M.: Dynamic response for motion capture animation. In: Fiume, E. (ed.) SIGGRAPH 2005. Computer Graphics Proceedings, pp. 697–701. ACM Press, New York (2005)
2. Faloutsos, P., van de Panne, M., Terzopoulos, D.: Composable controllers for physics-based character animation. In: Fiume, E. (ed.) SIGGRAPH 2001. Computer Graphics Proceedings, pp. 251–260. ACM Press, New York (2001)
3. Hirukawa, H., Hattori, S., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., Fujiwara, K., Morisawa, M.: A universal stability criterion of the foot contact of legged robots - adios zmp. In: IEEE International Conference on Robotics and Automation Orlando, Florida, pp. 1976–1983. IEEE Computer Society Press, Los Alamitos (2006)
4. Bretl, T., Latombe, J.C., Rock, S.: Toward autonomous free-climbing robots. Robotics Research , 6–15 (2005)
5. Bretl, T., Rock, S., Latombe, J.C.: Motion planning for a three-limbed climbing robot in vertical natural terrain. In: IEEE Int. Conf. on Robotics and Automation, Taipei, Taiwan, vol. 3, pp. 2946–2953. IEEE Computer Society Press, Los Alamitos (2003)
6. Or, Y., Rimon, E.: Computation and graphic characterization of robust multiple-contact postures in gravitational environments. Department of Mechanical Engineering, Technion, Israel (2004)

7. Sentis, L., Khatib, O.: A whole-body control framework for humanoids operating in human environnements. In: IEEE International Conference on Robotics and Automation Orlando, Florida, pp. 2641–2648. IEEE Computer Society Press, Los Alamitos (2006)
8. Rennuit, A., Micaelli, A., Merlhiot, X., Andriot, C., Guillaume, F., Chevassus, N., Chablat, D., Chedmail, P.: Passive control architecture for virtual humans. In: IEEE/RSJ International Conference on Intelligent Robots and Sytems, Edmonton, Canada, pp. 1432–1437 (2005)
9. Wieber, P.: Modélisation et Commande d'un Robot Marcheur Anthropomorphe. PhD thesis, Ecole des Mines de Paris (2000)
10. Abe, Y., Silva, M.D., Popović, J.: Multiobjective control with frictional contacts. In: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, pp. 249–258. ACM Press, New York (2007)
11. Miller, D., Morrison, W.: Prediction of segmental parameter using the hanavan human body model. Medicine and Science in Sports 7, 207–212 (1975)
12. Hanavan, E.: Mathematical model of the human body. Wright-Patterson Air Force Base, Ohio, AMRL-TR, 64–102 (1964)
13. Dempster, W., Gaughran, G.R.L.: Properties of body segments based on size and weight. American Journal of Anatomy 120, 33–54 (1967)
14. Park, J.: Principle of dynamical balance for multibody systems, multibody system dynamics. 14, 269–299 (2005)
15. Trinkle, J., Tzitzoutis, J., Pang, J.: Dynamic multi-rigid-body systems with concurrent distributed contacts: Theory and examples, philosophical trans. on mathematical, physical, and engineering sciences. Series A 359 (2001) (2575)–2593
16. Rennuit, A.: Contribution au Contrôle des Humains Virtuels Interactifs. PhD thesis, Ecole Centrale de Nantes (2006)
17. Duval-beaupére, Schmidt, G., Cossonp, C.: A barycentrometric study of sagittal shape of spine and pelvis: the conditions required for an economic standing position. Annals of Biomedical Engineering 20, 451–462 (1992)
18. Gangnet, N., Pomero, V., Dumas, R., Skalli, W., Vital, J.-M.: Variability of the spine and pelvis location with respect to the gravity line : a three-dimensional stereoradiographic study using a force platform. Surgical and Radiologic Anatomy 25, 424–433 (2003)
19. Azevedo, C., Poignet, P., Espiau, B.: Artificial locomotion control: from human to robots. Robotics and Autonomous Systems 47, 203–223 (2004)

Haptic Exploration of Mathematical Knots

Hui Zhang, Sidharth Thakur, and Andrew J. Hanson

Computer Science Department, Indiana University
{huizhang,sithakur,hanson}@cs.indiana.edu

Abstract. We present a novel multi-modal haptic interface for sketching and exploring the structure and properties of mathematical knots. Our interface derives from the familiar pencil-and-paper process of drawing 2D knot diagrams to facilitate the creation and exploration of mathematical knots; however, with a touch-based interface, users can also leverage their physical intuition by seeing, touching, and feeling the knots. The pure haptic component provides an intuitive interaction model for exploring knots, focusing on resolving the apparent conflict between the continuous structure of the actual knot and the visual discontinuities at occlusion boundaries. The auditory component adds redundant cues that emphasize the traditional knot crossings, where the haptic proxy crosses a visual disruption in the graphics image. Our paradigm enhances and extends traditional 2D sketching methods by exploiting both touch and sound to assist in building clearer mental models of geometry such as knot structures.

1 Introduction

2D knot diagrams are often used to help explain the concepts in knot theory. They are commonplace in textbooks and provide a form of physical intuition about the abstract principles (see, e.g., [**?**]). Students often draw pencil-and-paper diagrams to represent, study, and visualize 3D knot structures.

Knots are usually drawn on flat 2D media such as a blackboard or a sheet of paper. While such an image depicts the general shape of the curve, all 3D spatial information is implicit, and must be encoded at the crossings (where one strand crosses over or under another section) to resolve possible ambiguities (see Figure 1(a)). A classic "knot crossing-diagram" (see Figure 1(b)) tries to alleviate this problem by cutting away pieces of the curve that lie underneath other pieces in the chosen projection. The drawback of this approach is that one may have to use an eraser while drawing a knot, although experts can directly draw images like Figure 1(b) in a single step. This representation makes the knot's path appear visually discontinuous, while the true structure of the knot is of course continuous (see e.g., Figure 1(c)). Our task in this paper is to show how one can fully exploit computer graphics and computer based haptics to make a smooth transition from the 2D visual representation to an environment rich in 3D information and feedback.

2 Overview

There have been a number of interesting attempts to use interactive computer systems to help describe 3D curves and knots. For example, sketching a knot with Scharein's

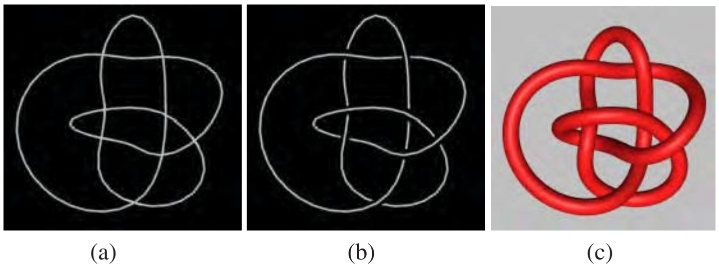


Fig. 1. Common ways of drawing knots. (a) The 2D knot projection has no 3D cues. (b) The 2-1/2 D knot diagram provides sufficient 3D depth information to characterize the 3D geometry. (c) Rendering with light and material adds apparent 3D geometry, depth, and shape to the 2D image.

Knotplot is done in a plane using a mouse-based control system [2], while Cohen et al. create 3D curves by correlating the curve with its sketched shadow to compute the curve's 3D shape [3]. Snibble et al. merge haptic interfaces with the study of geometric characteristics relevant to knots (see, e.g., [4]).

Other representative efforts include a variety of ways to edit and simulate knots and ropes (see, e.g, [5,6,7]). Recent work also suggests how haptic exploration of projected 4D objects can exploit topological continuity by ignoring illusory 3D surface intersections and focusing on the intrinsic 4D geometry [8].

The technique we present in this paper is an extension of the ideas used in [2,3,4,8]. By combining graphics and collision-sensing haptics, we enhance the 2D drawing protocol to successfully leverage 2D pen-and-paper or blackboard skills.

In our approach, users sketch knots in a 2D space, much as we might sketch a 2D knot diagram on a piece of paper, while guided by appropriate constraint forces. Real-time force feedback is used to prevent the haptically sketched curve from passing through curve segments in the 2D drawing that actually collide in the full 3D space; collisions are handled by moving the haptic probe to make over/under-crossing decisions (the "$2\frac{1}{2}$D" method).

Our approach also provides an intuitive touch-based physical navigation of the continuous structure of the actual 3D knot irrespective of the visual discontinuity. In some sense, the virtual reality of the haptic interface surpasses reality, since there are no physical obstructions to get caught on while tracing a shape in virtual space. We automatically override the apparent visual collisions, conflicts in the sketched image of the knot diagram, and keep the haptic probe anchored to the 3D continuity that underlies the whole structure, regardless of whether it is above or below another conflicting part relative to the projection point.

Figure 2 shows the mental model of a user navigating on a Knot 8_{21} diagram. Our paradigm aims at a smooth transition from the 2D visual representation to an environment supporting 3D depth feedback.

Overall features of the interface

- Intelligent guiding forces; past motion is used to haptically hint at a smoothed future direction to avoid getting stuck in sections having high derivatives.

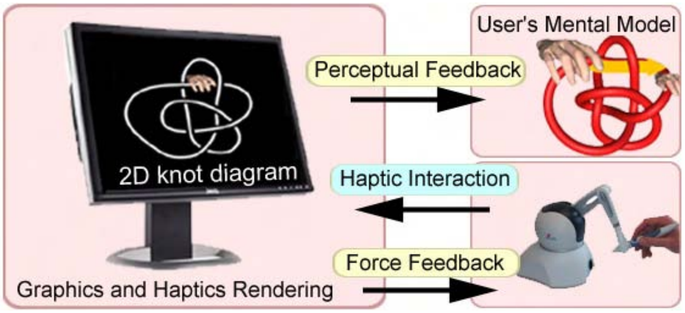


Fig. 2. Screen image, haptic probe, and the user's corresponding mental model during haptic navigation on the 2D representation of a 3D mathematical knot

- Input of over/under crossing interaction using "$2\frac{1}{2}D$" haptic feedback.
- Exploitation of continuous haptic exploration to supplement (discontinuous) visual representation and build more complete mental models.
- Exploit auditory cues to provide image-independent crossing information.

3 Implementation Methods

In this section, we describe the families of methods used to implement the interaction procedures and user interfaces. Our fundamental techniques are based on a wide variety of prior art, including haptic interfaces focusing on virtual realism (see, e.g., [9]), the exploration of unknown objects by robotic fingers (see, e.g., [10,11]), and other variants on haptic exploration techniques [12,13]. Relevant methods of force feedback and user assistance include, e.g., the work of [14,15,16,17,18].

We have found many requirements of our interface task to be unique, and thus we have developed a number of hybrid approaches. For example, when sketching 2D knot diagrams, the haptic probe should be constrained to the 2D projection plane, and yet must still detect and respond to potential collisions in the 3D space. Conversely, when navigating on 2D knot diagrams, *apparent* collisions in the 2D knot diagram that are physically separated in 3D must be ignored.

The basic modeling methods, components, and features characterizing our interface are summarized in the sections below.

3.1 Haptic Knot Creation

Force Modeling for 2D Drawing. Our basic force model simulates a "sticky" stylus in the 2D space using a damped spring configuration model; the probe can move freely in the 2D surface to create new 2D projected images of 3D curves.

The damped spring force model calculates the point $\mathbf{C}'$ as the projection of the haptic device proxy $\mathbf{C}$ on the 2D plane [19]. The difference $\mathbf{N} = \mathbf{C}' - \mathbf{C}$ is used to compute a generalized Hooke's law force

$$\mathbf{F}_m = H|\mathbf{N}|^{1+\beta}\hat{\mathbf{N}} \,. \tag{1}$$

Here H is a constant, and $\beta = 0$ for an ideal (linear) spring. This mechanical restoring force is applied whenever the stylus is displaced from the surface, but we allow force-free motion along the direction tangent to the surface to facilitate exploration of the surface structure. The user must apply substantial effort to overcome this force, so the stylus feels stuck to the surface. The damping force is taken to be

$$\mathbf{F}_d = -K_d \mathbf{V} \, , \tag{2}$$

where $\mathbf{V}$ is the radial velocity used to smooth the force feedback.

Collision Avoidance and Repulsive Forces. When sketching 2D projected images of 3D curves, collision detection is applied to detect whether the proxy apparently collides with other parts of the 2D projected image, and a repulsive force is rendered to prevent the haptic proxy from passing through segments in the projected image that actually collide in the full 3D space; collisions are handled by physically lifting (or pushing) the haptic probe (in the direction perpendicular to the 2D plane) to create over/under-crossings.

Collision handling methods (see, e.g., [20] and [21], to mention only a few) detect a collision between virtual objects when they have just begun to penetrate each other. However, in a haptic interface, the colliding pair positions have physical manifestations, so one cannot simply shift both positions to undo the collision. Therefore we use a dynamic repulsive force to avoid collisions. The force model that we use to physically detect an impending collision and prepare for an over/under-crossing choice is

$$\mathbf{F}_r = -HS^{-1-\beta}\hat{\mathbf{V}} \, . \tag{3}$$

Here S represents the distance between the probe itself and the impending collision with the projected image, and $\mathbf{V}$ is the radial velocity. This force slows down the haptic proxy's velocity as it approaches an existing 2D image segment, thus allowing the system to detect and manage collisions in a physically realistic manner.

When a collision occurs between a piece of an edited object and an existing object in 2D space, users must make explicit over and under choices by lifting or pushing the haptic probe. We thus have a "$2\frac{1}{2}$D" collision avoidance that effectively leverages skills developed from work with pencil and paper, and exploits intuitive force feedback to aid the drawing process. This is closely related to the non-haptic methods for sketching and manipulating 3D curves advocated by Scharein [2] and by Cohen et al. [3].

Smoothing the Sketching Process. The $2\frac{1}{2}$D interface in principle is sufficient to allow the user to sketch a knot diagram on the given 2D surface. However, in practice, this free-hand 2D constrained drawing introduces significant jitter (human and mechanical). To improve on this, we follow Haeberli's *Dynadraw* method [22], connecting a virtual mass to the cursor position via a damped spring. As the user moves the cursor, the literal path is modified to create smooth, calligraphic strokes (see Figure 3(a)(b)). From *Dynasculpt* (Snibbe [4]), a haptic variant of *Dynadraw*, we adopt the method of attaching a virtual mass-spring system to the haptic probe position to smooth the free-hand results.

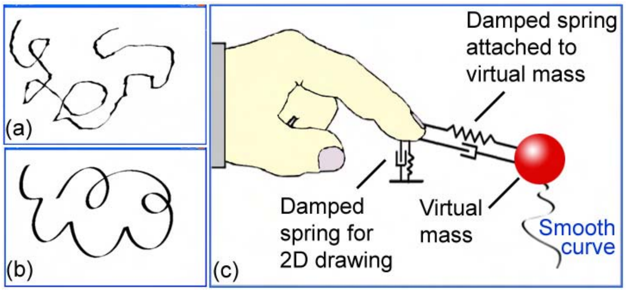

(a)
(b)
(c)
Damped spring attached to virtual mass
Damped spring for 2D drawing
Virtual mass
Smooth curve

Fig. 3. (a) A typical jittery result of free-hand drawing. (b) Smoothed drawing resulting from the method that connects a virtual mass to the cursor position via a damped spring. (c) The force model for 2D sketching adapted from Dynasculpt's dynamic model.

The implementation is calculated using Hooke's law with a damping term, adapting Equations (1) and (2) to give

$$\mathbf{f} = -H(\mathbf{P}_m - \mathbf{P}_f) - K_d\mathbf{V}\ , \tag{4}$$

where H is the spring constant, K_d is the damping constant, $\mathbf{P}_m$ is the position of the virtual mass, and $\mathbf{P}_f$ is the real-world finger-tip position as measured by the haptic system. The position of the virtual mass is updated using Newton's laws, $m{\cdot}\mathbf{a} = \mathbf{f}$, where m is the chosen mass, and we solve the second order differential equation using Euler's method. This force model is illustrated in Figure 3.

Examples. In Figure 4, we illustrate typical steps for the haptic creation of a trefoil knot. Sample distances are interactively adapted, e.g., via bounding sphere checking, to make the final curve segments close to the same size and well-behaved (see [5]). Figure 5 shows examples of more complex results.

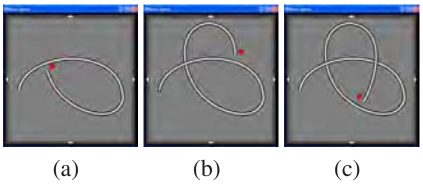


Fig. 4. A sequence of frames showing haptic knot creation via a series of under, over, under... crossings

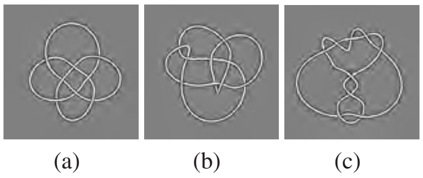


Fig. 5. Examples sketched using the $2\frac{1}{2}$D haptic interface. (a) Knot 8_{18} (b) Knot 9_{33} (c) Knot 9_6.

3.2 Haptic Navigation

The overall experience can be improved by supporting assisted navigation that follows the local continuity of the object being explored. Tracing a real physical knot with one's finger results in collisions of the rope with the fingertip (see, e.g., Figure 6(a)),

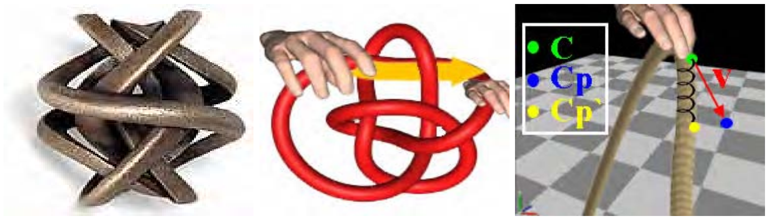


Fig. 6. (a) It is nearly impossible to trace a real knotted rope continuously even when you are holding the physical object in your hands (`www.bathsheba.com/sculpt/clef/`). (b) The free motion of a computer-based haptic probe supports a continuous motion that follows the *local continuity* of the object being explored. (c) The interactive response is improved when damped-spring forces steer the probe in the desired direction.

forbidding smooth navigation; the projected image of a knot can contain massive interruptions of visual continuity as well. The computer-based haptic interface, however, can do something real-life cannot do, which is to support a continuous motion that follows the continuity of the object being explored without encountering physical obstructions; visual collisions are unphysical and have no effect on the haptic motion. The haptic navigation method thus resolves the apparent conflict between the continuous structure of the actual 3D knot and the visual discontinuities at occlusion boundaries in the 2D projection (see Figure 6(b)).

Haptic navigation can be assisted by force suggestions that constrain the allowed motion, while assisting and guiding the user's fingertip (the probe) towards the predicted position. The following steps describe the haptic servo loop model for adding force suggestions (see Figure 6(c)):

```
1. Get current haptic device coordinate $\mathbf{C}$, velocity $\mathbf{V}$, and the instantaneous
   update rate of the device $R$.
2. Compute the predicted haptic device coordinate
   $\mathbf{C_p} = \mathbf{C} + \mathbf{V} \cdot \frac{1}{R}$.   ($\frac{1}{R}$ is the time step.)
3. Compute $\mathbf{C_p}'$ as projection of $\mathbf{C_p}$ on curve image $\mathbf{I}$.
4. Apply a damped spring force between $\mathbf{C}$ and $\mathbf{C_p}'$.
```

Selecting Viewable and Touchable Knot Images. Different choices of projection can result in very different images; 2D knot diagrams are not invariant under changes of projection. Thus, for some tasks, we may wish to vary the chosen projection to optimize the view. One way is to optimize some aspect such as the projection size of the segment currently being touched by the probe; this has the advantage of making both the image size and the haptic-sensitive path correspond to the maximal true local metric curve length around the point being probed [23]. Figure 7 illustrates the perceptual variations possible with a single topological knot, where Figure 7(a) has the fewest interruptions as well as the maximal projected area.

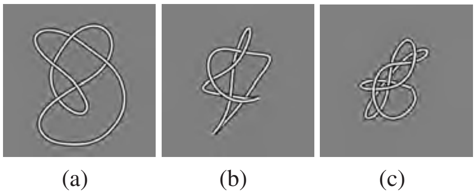


Fig. 7. The rigid 4_1 knot can be represented with different projections from 3D to 2D, some of which are very difficult to understand if too many lines cross in the image

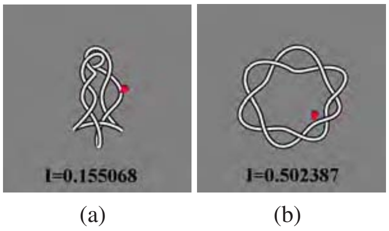


Fig. 8. Computing the knot viewpoint quality measure (**I**) of a 3D knot projected to a 2D plane

We approach the problem by adapting methods used in knot theory, as well as in multidimensional data visualization and viewpoint selection. For example, Kamada and Kawai [24] consider a viewing direction to be good if it minimizes the number of degenerate faces under orthographic projection, Hlavác et al. [25] optimize the exploration of a set of object images, and Vázquez et al. [26,27] use an information theoretic measure for viewpoint entropy. Starting from this background, we synthesize a model that is closely related to the requirements of the haptic exploration task for knot diagrams. Our optimization measure, composed of the projected curve length and segment visibility in the knot images, is given by

$$I(K,c) = \sum_{i=0}^{N_s} \left(\frac{L_i}{L_t} \log \frac{L_i}{L_t} + V(i) \right) . \tag{5}$$

Here L_i represents the projected length of curve segment i and L_t is the total length of the knot curve embedded in 3D; $V(i)$ is the *visibility test function* for curve segment i, where $V(i) = -1$ if the segment is crossed by another segment, and otherwise $V(i) = +1$. Figure 8 shows two typical projected images of the 7_1 knot and their optimization measures. The measure is maximal, hence better, for Figure 8(b).

Auditory cues. In practice, knot structure is encoded by the location and character of the crossings. However, if one explored a knotted curve with a probe constrained

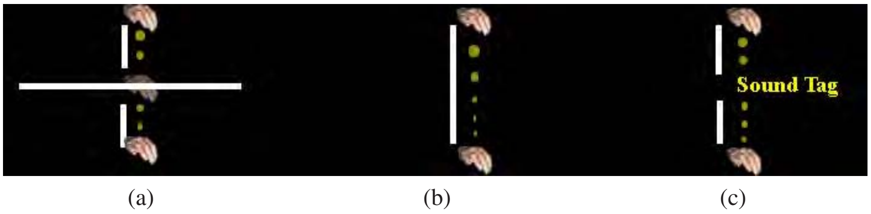


Fig. 9. (a) With visual feedback, the user is acutely aware of sliding through visual interruptions in the 2D knot crossing-diagram. (b) Without using visuals, however, one would be totally unaware of encountering a knot crossing. (c) Sound cues supplement or replace visual cues to assist in building a clear mental model of the mathematical knot.

to the knot, with no visual feedback, all knots would be the similar — just a *smooth* path coming back to itself. We therefore choose to supplement the visual display by adding a sound tag that distinguishes each over and under crossing in the 2D knot crossing-diagram, as illustrated in Figure 9. Sound tags potentially make knot exploration possible without vision.

4 Representation and Display

We next give the details of how exactly knots are rendered and displayed. Our system has two major display modes. Any knot may be displayed in either a "2D knot crossing-diagram" mode or in a "3D smooth tube" mode (see Figure 1(b)(c)). Since the "2D knot crossing-diagram" is common practice in textbooks on knot theory, our interface uses it as the default representation for knots. However, many users also like the "smooth look," with the understanding that the apparently smooth curves and surfaces realistically represent the 3D knot structure. In this section, we describe the methods used to rendering a mathematical knot in both the 2D and 3D representations.

Rendering 2D Knot Crossing-Diagram. The classical means of describing the 3D curve of a knot is to draw a two-dimensional projected curve that is broken each time it is occluded by a piece of the whole curve that is nearer to the 3D projection point. At these occlusion points, the part of the curve nearest the projection point is continuous and the part passing underneath is interrupted for a short interval to each side of the occluding curve at the crossing (see Figure 1(b)). Our method of rendering such a 2D knot diagram is to attach a *thickened* curve segment in background color behind each of the curve segments that are rendered in foreground color, so that a *visual break* is created on each side of an under-crossing. Program 1 describes our 2D knot crossing-diagram rendering method.

Program 1. Procedure for rendering a 2D knot crossing diagram

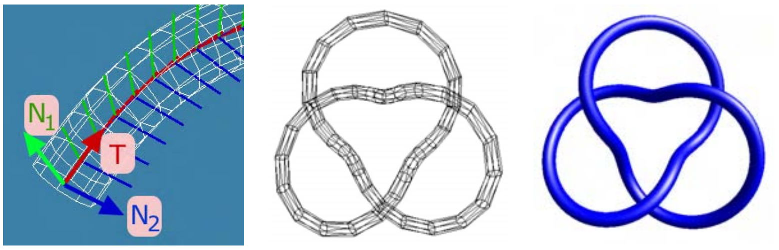
```
glDisable(GL_LIGHTING);
For each curve segment
  glColor3f(BG.r, BG.g, BG.b);
  glLineWidth($\delta + \varepsilon$);
  draw curve segment;
  glColor3f(FG.r, FG.g, FG.b);
  glLineWidth($\delta$);
  glDepthFunc(GL_LEQUAL);
  draw curve segment;
  glDepthFunc(GL_LESS);
glEnable(GL_LIGHTING);
```

Rendering 3D Smooth Knot. A curve $\mathbf{C}(t)$ can be "thickened" by enveloping it in a tube. A polygonal model for tubing can be computed by attaching orientation frames sampled along the curve, and by tracing a 2D profile (such as a circle) in the planes of frames to build the skeleton. An orientation frame is represented in the form of a 3×3 orthonormal rotation matrix $[\mathbf{T} \quad \mathbf{N}_1 \quad \mathbf{N}_2]$.

Here, $\mathbf{T}(t) = \mathbf{C}'(t)/||\mathbf{C}'(t)||$ is the normalized tangent vector determined directly by the curve geometry. $(\mathbf{N}_1(t), \mathbf{N}_2(t))$ are a pair of orthonormal vectors spanning the plane perpendicular to $\hat{\mathbf{T}}$(t) at each point of the curve $\mathbf{C}$(t) (see Figure 10(a)). To generate a tube, we sweep the chosen set of frames through each curve point $\mathbf{C}(t)$ to produce a set of connected points $\mathbf{X}(t)$ on the tube (see Figure 10(b)):

$$\mathbf{x}(t,\theta) = \mathbf{C}(t) + \cos(\theta)\hat{\mathbf{N}}_1(t) + \sin(\theta)\hat{\mathbf{N}}_2(t)$$

The resulting structure is sampled in t over one full 2π period in θ to produce a tessellated tube. The base frames at each point of the curve can be computed by a variety of methods such as the Frenet-Serret [28] or the Bishop (parallel transport) method [29].



(a) (b) (c)

Fig. 10. Tubing model for generating thickened curves. (a) The orientation frames along a curve segment. (b) The polygonal wire frame model. (c) Smooth knot tubing rendered with 3D light and material properties.

5 User Applications and Feedback Results

Our implementations employ a SensAble Technology Omni PHANToM force-feedback haptic device combined with a high-performance graphics card supporting OpenGL. The user interface and graphics rendering are based on OpenGL and the haptics system is based on SensAble's OpenHaptics API. The software runs on a Dell PC desktop with a 3.2GHz Intel Pentium 4 CPU. The haptic frame rate remains above 1000Hz for most of tasks we have encountered (a haptic device requires a refresh rate of about 1000Hz in order to give a kinesthetic sense of stiff contact). The basic technical framework for haptic curve manipulation and understanding described here has been integrated into several distinct user interface environments; we mention two here:

Knot Exploration Interface. A touch-based pedagogical tool was developed to assist students taking a basic undergraduate topology class. The user interface was designed to help students understand the correspondence between the strict 2D approach to representing knots using crossing diagrams and the corresponding structures in 3D space (see Figure 11). A group of eight subjects was instructed in the use of the system and then explored a variety of 2D knot diagrams and their 3D counterparts, both before and after being exposed formally to knot diagrams in the classroom. A set of tasks involving identifying apparently different but topologically identical knots and untangling unknotted curves was presented to the subjects, and these results are reported in more detail elsewhere. The subjects were also asked to give verbal descriptions and evaluations of their experiences, and the general response to the system was that it was of significant value in creating a clear mental model associating the 2D knot diagram with the 3D version of the knot. The following is a summary of the common responses of the participants who used our knot exploration system:

– Features of the application participants liked most: force feedback during navigation on the knot, the ability to rotate and observe knots from different view points, and audio feedback indicating over and under crossings. One participant indicated the usefulness of the cast shadows of the knots in discriminating depth and height in 3D.
– Suggestions and comments: three participants indicated they would like the ability to edit knots in addition to exploration. One participant found it difficult to switch between the mouse and haptic stylus during exploration, while another user found the haptic device tiresome over extended durations.

Motor Assistance. A user interface was developed in which the knot data structures were supplemented by haptically-traceable curves, letters, and numbers that could be chosen easily by, for example, typing in one's name. This system is currently being adapted for extensive use at a laboratory that is devoted to assisting motor-impaired children to develop curve-tracing and letter-tracing skills; initial reports are that this application is quite successful. A snapshot of the application is displayed in Figure 12.

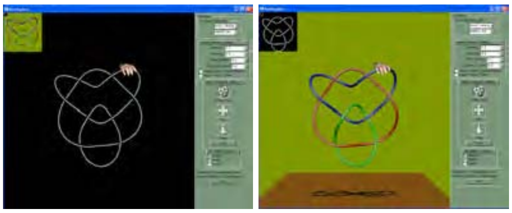


Fig. 11. User interface for Knot exploration: left, the 2D knot diagram mode, and right, 3D knot structure and exploration mode

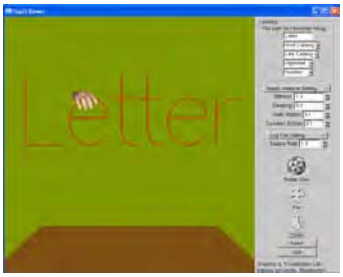


Fig. 12. Touch-based interface to help develop repetitive motion skills such as tracking curves and letters

6 Conclusion and Future Work

We have discussed a family of haptic methods for intuitively exploring mathematical knots. Current computer interfaces can support multi-modal displays that integrate visual, haptic, and auditory feedback and interaction. Exploiting these capabilities permits us to build a particular type of kinesthetic intuition about continuous complex geometric curves and surfaces such as mathematical knots. Starting from this basic framework, we plan to proceed to attack families of significant problems in knot theory such as the interactive manipulation of knotted curves and Reidemeister moves.

Acknowledgments

This research was supported in part by National Science Foundation grants CCR-0204112 and IIS-0430730.

References

1. Livingston, C.: Knot Theory. In: Livingston, C. (ed.) The Carus Mathematical Monographs. Mathematical Association of America, Washington, U.S, vol. 24 (1993)
2. Scharein, R.G.: Interactive Topological Drawing. PhD thesis, Department of Computer Science, The University of British Columbia (1998)
3. Cohen, J., Markosian, L., Zeleznik, R., Hughes, J., Barzel, R.: An interface for sketching 3d curves. In: SI3D 1999. Proceedings of the 1999 symposium on Interactive 3D graphics, pp. 17–21. ACM Press, New York, NY, USA (1999)
4. Snibbe, S., Anderson, S., Verplank, B.: Springs and constraints for 3d drawing. In: Proceedings of the Third Phantom Users Group Workshop, Dedham, MA (1998)
5. Brown, J., Latombe, J.C., Montgomery, K.: Real-time knot-tying simulation. The Visual Computer 20, 165–179 (2004)
6. Wang, F., Burdet, E., Dhanik, A., Poston, T., Teo, C.L.: Dynamic thread for real-time knot-tying. In: WHC 2005. Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 507–508. IEEE Computer Society Press, Washington, DC, USA (2005)
7. Phillips, J., Ladd, A.M., Kavraki, L.E.: Simulated knot tying. In: ICRA, pp. 841–846. IEEE Computer Society Press, Los Alamitos (2002)
8. Hanson, A.J., Zhang, H.: Multimodal exploration of the fourth dimension. In: Proceedings of IEEE Visualization, pp. 263–270. IEEE Computer Society Press, Los Alamitos (2005)
9. Baxter, W.V., Scheib, V., Lin, M.C.: dAb: Interactive haptic painting with 3D virtual brushes. In: Fiume, E. (ed.) SIGGRAPH 2001. Computer Graphics Proceedings, ACM SIGGRAPH, pp. 461–468. ACM Press, New York (2001)
10. Okamura, A.M.: Haptic Exploration of Unknown Objects. PhD thesis, Stanford University, Department of Mechanical Engineering, California, USA (2000)
11. Okamura, A.M., Cutkosky, M.R.: Feature detection for haptic exploration with robotic fingers. The International Journal of Robotics Research 20, 925–938 (2001)
12. Kim, L., Sukhatme, G., Desbrun, M.: A haptic-rendering technique based on hybrid surface representation. IEEE Comput. Graph. Appl. 24, 66–75 (2004)
13. Yu, W., Ramloll, R., Brewster, S.: Haptic graphs for blind computer users. In: Brewster, S., Murray-Smith, R. (eds.) Haptic Human-Computer Interaction. LNCS, vol. 2058, pp. 41–51. Springer, Heidelberg (2001)
14. Zahariev, M.A., MacKenzie, C.L.: Auditory, graphical and haptic contact cues for a reach, grasp, and place task in an augmented environment. In: ICMI 2003. Proc. of 5th Intl. Conf on Multimodal Interfaces, pp. 273–276. ACM Press, New York (2003)
15. Kennedy, J.M.: Optics and haptics: The picture. In: The conference on Multimodality of Human Communication: Theory, Problems and Applications, University of Toronto (2002)

16. Forsyth, B.: Intelligent support of interactive manual control: Design, implementation and evaluation of look-ahead haptic guidance. Master's thesis, The University of British Columbia (2004)
17. Park, J.G., Niemeyer, G.: Haptic rendering with predictive representation of local geometry. In: HAPTICS, pp. 331–338 (2004)
18. Reynolds, C.W.: Steering behaviors for autonomous characters. In: The proceedings of the 1999 Game Developers Conference, pp. 763–782 (1999)
19. SensAble, Inc.: 3D Touch SDK OpenHaptics Toolkit Programmer's Guide (2004)
20. Larsson, T., Akenine-Möller, T.: Collision detection for continuously deforming bodies. In: Eurographics 2001. Short Presentations, Manchester, Eurographics Association 325–333 (2001)
21. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree: A hierarchical structure for rapid interference detection. Computer Graphics 30, 171–180 (1996)
22. Haeberli, P.: Dynadraw. Grafica OBSCURA (1989)
23. Hanson, A.J., Ma, H.: Space walking. In: Proceedings of Visualization 1995, pp. 126–133. IEEE Computer Society Press, Los Alamitos (1995)
24. Kamada, T., Kawai, S.: A simple method for computing general position in displaying three-dimensional objects. Computer Vision, Graphics, and Image Processing 41, 43–56 (1988)
25. Hlavác, V., Leonardis, A., Werner, T.: Automatic selection of reference views for image-based scene representations. In: ECCV (1), pp. 526–535 (1996)
26. Vázquez, P.P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint selection using viewpoint entropy. In: VMV 2001. Proceedings of the Vision Modeling and Visualization Conference 2001, Aka GmbH, pp. 273–280 (2001)
27. Vázquez, P.P., Feixas, M., Sbert, M., Llobet, A.: Viewpoint entropy: a new tool for obtaining good views of molecules. In: VISSYM 2002. Proceedings of the symposium on Data Visualisation 2002, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, pp. 183–188 (2002)
28. Gray, A.: Modern Differential Geometry of Curves and Surfaces. CRC Press, Inc., Boca Raton, FL (1993)
29. Bishop, R.L.: There is more than one way to frame a curve. American Mathematical Monthly 82, 246–251 (1975)

Fitting the World to the Mind: Transforming Images to Mimic Perceptual Adaptation

Michael A. Webster[1], Kyle McDermott[1], and George Bebis[2]

[1] Departments of Psychology
[2] Computer Science and Engineering
University of Nevada, Reno

Abstract. Visual sensitivity is constantly adjusting to the current visual context through processes of adaptation. These adaptive changes strongly affect all perceptual judgments and optimize visual coding for the specific properties of the scenes before us. As a result human observers "see" better when they are first allowed to adapt to a specific context. The basic form of the response changes resulting from adaptation have been studied extensively and many are known in broad outline. Here we consider the advantages of applying these changes to images, to simulate the processes of adaptation that normally occur within the observer. Matching images to the observer may obviate the need for some forms of perceptual learning and offers a number of potential benefits for interacting with visual displays.

1 Introduction

Visual perception is a highly dynamic process, adjusted continuously by mechanisms of adaptation that recalibrate visual coding according to the statistics of the image we are currently viewing [1]. Numerous classic examples illustrate the marked adaptability of perception and how changes in the states of adaptation profoundly alter the way the world looks. For example, after viewing a red square a gray square appears greenish; exposure to a tilted line causes a vertical line to appear tilted in the opposite direction; and after a few moments looking at the downward flow of a waterfall, the static rocks to the side appear to ooze upward. Adaptation aftereffects occur not only for simple stimulus dimensions but also for highly complex and abstract image properties. For example, the perceived configuration of a face can be strongly biased by prior exposure to a distorted face [2] or to faces drawn from different identities [3] or facial categories such as gender, ethnicity, or expression [4]. Such results suggest that the sensitivity changes underlying adaptation occur at all stages of visual coding and may in fact be an intrinsic and fundamental property of neural activity.

Adaptation is thought to confer a number of interrelated functional advantages to the observer. Many of these have been postulated based on information theory, though the extent to which they can be linked to actual improvements in behavioral performance remains to be established. Some potential benefits of adaptation include:

1. Maximizing the limited dynamic range available for visual coding. The range of response levels a neuron can reliably transmit is highly restricted and may be orders of magnitude smaller than the range of stimulus levels that are typically encountered. Thus this range must be centered around the ambient stimulus level much as a camera's settings are adjusted to match the limited operating characteristic of the film or CCD array. The clearest example of the consequent improvements for seeing is in light adaptation, in which visual sensitivity adjusts to the mean light level so that the "exposure" level remains at an appropriate level for perceiving the variations in light around the mean [5]. Similar adjustments may also be important for coding image contrasts. For example, cells in visual cortex code only a limited range of contrasts and must adapt to the ambient contrast level in scenes to avoid response saturation [6].

2. Improving visual discrimination. Adjusting to the mean stimulus level in images allows differences around the mean to be more easily distinguished. Again, we are sensitive to variations in light only around the average light level we are adapted to, and similarly in color vision, discrimination is best for stimuli near the white point and falls progressively for saturated colors [7]. Analogous processes might also underlie high-level perceptual judgments such as the "other race effect" in face perception, in which we can readily discriminate differences between faces within the ethnic group we are exposed to while faces drawn from novel groups appear similar.

3. Improving coding efficiency within mechanisms. To carry the most information a neuron's responses should be matched to the distribution of stimuli so that any given response level occurs with equal probability [8]. This means that responses should change rapidly where stimulus levels are abundant while asymptoting where stimulus levels are rare, similar to histogram equalization. It is unclear to what extent actual processes of adaptation can adjust to the specific distribution of stimulus levels and whether these adjustments differ for different stimulus dimensions. However, in color coding it is clear that adaptation independently alters responses to both the mean and the variance of the stimulus [9].

4. Improving coding efficiency across mechanisms. At least at peripheral stages, information in the visual system is thought to be represented within filters or channels that span different ranges of the stimulus dimension [10]. Thus color is coded by three cone types with different spectral sensitivities while orientation or size are represented by multiple channels each tuned to a limited range of tilts or spatial scales. By allowing each channel to adjust to the mean stimulus level it is exposed to, responses are equated across the population of mechanisms. This maximizes efficiency by allowing each channel to carry the same amount of potential information [11].

5. Removing redundancy. Some forms of adaptation might include mutual inhibition between channels whenever they respond together to the stimulus. Such processes could improve coding efficiency by decorrelating the responses across the set of channels, allowing each to carry independent information about the stimulus [12]. A possible sign of these adjustments is contingent aftereffects, in which observers adapt to the correlations between stimulus dimensions. For example, in the McCollough Effect, viewing red vertical bars alternated with green horizontal bars causes a vertical gray bar to appear greenish while horizontal bars appear reddish. The color aftereffect is thus contingent on the pairing of color with orientation (and also on the pairing of chromaticity and luminance in the image). The negative color aftereffects are consistent with the responses predicted by decorrelating the joint responses to color and luminance and tilt in the stimulus [12], but could also arise from independent sensitivity changes within mechanisms selectively tuned to different combinations of orientation and color.

6. Maintaining perceptual constancy. The preceding examples all share the potential advantage of improving the discriminative capacity of perception by maximizing the information carried by the neural response. A somewhat different goal of perception may be to maximize recognition – to allow a given visual response to faithfully represent a consistent property of the stimulus. The problem of recognition often involves extracting invariant characteristics of the object by discounting extraneous sources of variation in the visual response. The role of adaptation in solving this problem has been most thoroughly investigated in the case of color constancy, in which the goal of the observer is to recognize the same surface under different viewing conditions [13]. This problem is ill-posed because the color signal reaching the observer depends on both the surface reflectance function and the spectrum of the incident illumination. Many different processes contribute to color constancy in human vision, yet adapting to the average color in the scene can often lead to approximate constancy by removing the changes in the average color owing to the illuminant [13]. The same process may also underlie a more important form of constancy by removing variations in the observer. For example, the lens of the human eye includes an inert pigment that selectively absorbs short-wavelength light. The density of the lens pigment increases with age and thus the average light spectrum reaching the receptors is very different in younger and older observers. However, color appearance changes little with age, because adaptation continuously readjusts the visual response to discount the changes in visual sensitivity [14]. Similar adjustments may account for the constant perception of image focus despite the marked differences in visual acuity across the visual field or as spatial sensitivity varies during development and aging [15].

7. Building predictive codes. Finally, by adjusting to the current stimulus adaptation may allow the visual code to take the form of a prediction about the world [16]. The concept of predictive codes is closely related to norm-based codes in visual perception. In these the stimulus is represented by how it deviates from an average or prototype stimulus. Many visual dimensions may be represented relative to norms, including color, shape, and motion, and high-level properties such as facial identity [17]. Adaptation is thought to play a critical role in such codes by establishing and updating the norm based on the average stimulus history. This has the advantage that resources can be devoted to signaling only the errors in the prediction, which are generally the most informative features of the image. In particular, adapting to the average properties of images may increase the perceptual salience of novel image features and thus more readily allow the visual system to detect statistical outliers or "suspicious coincidences" in the environment [12].

As the foregoing list illustrates, most accounts of adaptation assume that the response changes are designed to improve visual performance. That is, an observer can see better if they are appropriately adapted to the image they are currently viewing. Normally this requires changing the sensitivity of the observer in order to match visual coding to the current environment. Thus the processes of adaptation involve "Fitting the mind to the world" [18]. However, image processing provides the potential for increasing visual performance by instead adapting the image to the observer, or "fitting the world to the mind." To the extent that the visual transforms underlying adaptation are known, images might be processed to simulate the changes in perception that would normally occur through adaptation, obviating the need for adjustments in the observer. Here we consider some of the transforms that are likely to occur and their implications for particular visual tasks. Our aim is not to offer a specific transform, but rather to note the potential for developing algorithms that might improve human performance with visual displays.

2 Processes of Adaptation

Here we describe the visual response changes that actually occur during perceptual adaptation. While we remain far from a complete understanding of these changes, certain general principles appear well established and common to many stimulus dimensions. These principles include the following factors:

1. Adjusting to the mean. Many adaptation phenomena involve changes in sensitivity driven by the average properties of the stimulus [1]. For example, light and chromatic adaptation reflect adjustments to the average luminance and chromaticity of the stimulus, while motion adaptation involves a change in the response to the average direction. Changes in sensitivity to the mean stimulus are the most ubiquitous property of visual adaptation and lead to the largest and most compelling visual aftereffects.

2. Normalization vs. repulsion. For many dimensions, adaptation to the mean causes the stimulus to appear more neutral and thus represents a renormalization of the visual code [17]. To draw again from color, viewing a red field causes the field to appear more desaturated and in the limit to appear achromatic. The currently viewed stimulus thus becomes the new norm or neutral point for color coding. In other cases adaptation is found to reduce sensitivity to the adapting stimulus without changing its appearance. A well known example of this is spatial frequency adaptation, in which subjects are adapted to a spatially varying grating of a particular frequency [10]. While the grating itself may appear reduced in contrast, its perceived size remains unaltered. However, the perceived size of gratings higher or lower than the adapting level appear biased or “repulsed” away from the adapting frequency. In general, repulsion tends to occur for stimulus dimensions like size or frequency, which do not have a unique neutral point, while normalization is more likely to occur for stimulus dimensions like color that include a clear norm.

3. Subtractive vs. multiplicative response changes. In some cases adaptation tends to reflect a simple gain change in visual sensitivity, equivalent to rescaling the response by a constant. Such scaling closely describes some of the sensitivity changes in chromatic adaptation. In other cases the adapted response is better described by subtracting a fixed constant from the response. A complete account of light and chromatic adaptation in fact includes both forms of adjustment, with multiplicative scaling of the receptor sensitivities and subtractive adjustments from spatial and temporal filtering [19]. The response changes underlying pattern adaptation may also approximate a subtractive loss in sensitivity [20], but have not been well characterized for many dimensions. The nature of the change has important implications for modeling the perceptual consequences of adaptation. For example, multiplicative changes have large effects at all contrast levels while subtractive changes become negligible at high contrasts where the visual response is already strong.

4. Adjusting to the variance. Adaptation adjusts not only to the average stimulus level but also to the range of variation around the average, or to contrast. For example, after viewing a pattern that alternates between red and green, reds and greens appear less saturated relative to other colors [1]. There have been comparatively few studies of how the visual system adapts to variance in other stimulus dimensions, but such adjustments seem necessary in order to exploit the full dynamic range of the visual mechanisms encoding these dimensions.
5. Selectivity of visual aftereffects. Visual aftereffects are selective – viewing a tilted line reduces sensitivity to lines with similar tilt, but has much less effect on the detectability or appearance of lines at more distant orientations [21]. Conversely, adapting to a particular color may lead to more uniform changes in appearance throughout color space [9]. These differences in part reflect how narrowly or broadly tuned the mechanisms are that encode the stimulus dimension. A large volume of research has focused on characterizing the number and selectivity of visual channels in different visual tasks [10]. However, this remains largely an empirical question so that it is not yet possible to specify a uniform principle that might guide the choice of filters for modeling a particular adaptation effect.

3 Formal Models of Adaptation

A number of computational models have now been developed to predict adaptation effects in the human visual system. These incorporate many of the basic processes outlined above and thus offer the potential to devise image transforms that simulate adaptation. Color coding is again the domain that has seen the most attention and for which current models have come closest to accurate quantitative predictions [19] [22]. As noted above, light and chromatic adaptation involve multiplicative gain changes in the cones followed by subtractive adjustments in post-receptoral processes. Models of these processes can now closely account for visual sensitivity to luminance and color under a wide variety of viewing conditions, and are at a point where they could be readily implemented in visual displays.

Models for adaptation processes at more central sites in the visual system have yet to reach the same degree of rigor, yet have nevertheless succeeded in characterizing the general form of aftereffects observed in pattern-selective adaptation. A number of these studies have combined gain control within the channel responses and decorrelation between the channel responses in order to account for the selective changes in sensitivity that follow adaptation to color contrast or orientation or motion [23-26]. However, essentially the same perceptual changes can also be modeled by instead assuming independent response changes across multiple mechanisms tuned to the stimulus dimension [10]. Thus either class of model could be used to alter the image to simulate pattern adaptation effects.

4 An Illustration of Performance Improvements with Adaptation

As a proof of concept of how pre-adapting images could facilitate visual performance, we consider the simplest case of adapting to a change in the mean color of an image.

Adjustments to the mean color are well predicted by independent multiplicative gain changes in the sensitivity of the three classes of cones, a process known as von Kries adaptation [9]. We examine how these adjustments could affect an observer's ability to detect color targets in the image. In previous studies, we showed that observers are faster at finding an odd color target if they are first adapted to the set of colors defining the background against which the target appears [27]. In the present case, we explore the effects of adapting to backgrounds that differ simply in the average color.

Figure 1 shows the set of colors composing the stimulus in terms of a standard space defined by two chromatic axes that represent relative responses in the long and medium wavelength sensitive cones at constant luminance (L-M) or the responses in the short wavelength sensitive cones at constant luminance (S-(L+M)). These axes thus isolate the component responses of the S cone or the L and M cones for any color and also represent the principal axes along which chromatic information is encoded at early stages of the visual pathway [28]. For our stimulus a background set of colors was displayed on a monitor by showing a dense set of overlapping ellipses with colors drawn from either the S or the L-M axis relative to a mean chromaticity, which in the example shown varied along the +L (reddish mean) or –L (greenish mean) pole of the L-M axis. Target colors were varied to span a range of hues and contrasts varying away from the background axis (unfilled symbols). On a given trial the target was displayed at a random location on the screen, and reaction times were measured for responding whether the target was on the right or left side. The measurements were made after the observer was adapted by viewing random successive samples of the background shown every 250 ms for 3 minutes, and settings were compared after adapting to the actual average color of the background (e.g. +L) or to a background that had a different color than the one on which they had to search for the target (e.g. –L).

Figures 3 plots the reaction times when the observer was searching for a color target that differed from the background, when the background color axis was orthogonal to the axis of the mean color shift. The stimulus colors for this condition are shown in Figure 2. Consider the upper left panel. Here the task was to find target colors that differed from the S axis of the background, and thus to find targets that were defined only by the color difference along the L-M axis. Search times varied from a few seconds for colors close to the background to a few hundred milliseconds for targets that were far removed from the background colors and thus easily "popped out" of the display. These search times are shown by the unfilled symbols for the condition in which the observer was adapted to the average color of the background. The critical comparison is when the observer was instead adapted to the wrong average color. This is shown by the filled triangles and search times are strongly delayed for all targets. The reason for the poorer performance is that by being adapted to the wrong average color the observer's achromatic point was shifted away from the mean color of the background so that during the search all colors appeared reddish and consequently less discriminable. That is, the adaptation was optimized for the wrong point in color space. These performance deficits could have been removed by instead adapting the +L image so that its mean coincided with the observer's current state of adaptation (i.e. transforming the image so that average chromaticity was –L). The performance gain from this adjustment is again indicated by the difference in search times going from the unadapted (filled triangles) to the adapted background (unfilled circles).

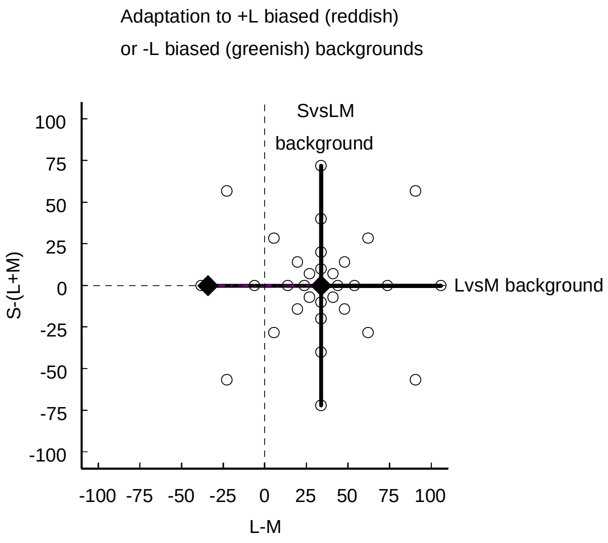


Fig. 1. Set of background (solid line) or target (circles) colors for the visual search task. Observers were adapted to one of two average colors (diamonds), corresponding to the mean of the background or a mean with the opposite color bias.

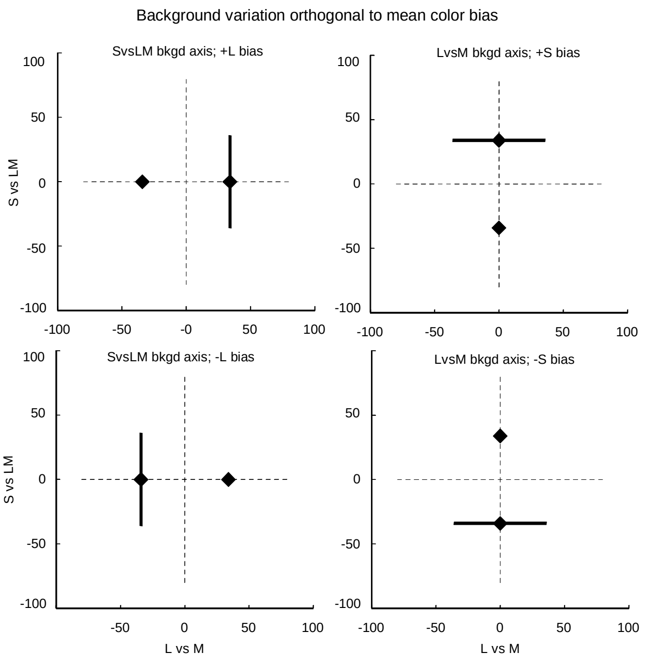


Fig. 2. Mean adapting colors (diamonds) and background colors (lines) when targets had to be detected by the color differences along the axis of the adaptation

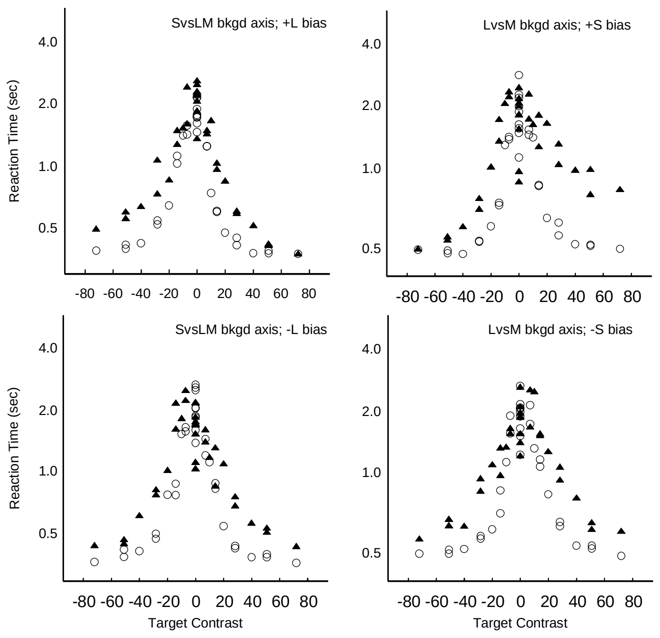


Fig. 3. Search times as a function of the target color contrast relative to the background axis after adapting to the mean background color (circles) or the opposite color bias (triangles)

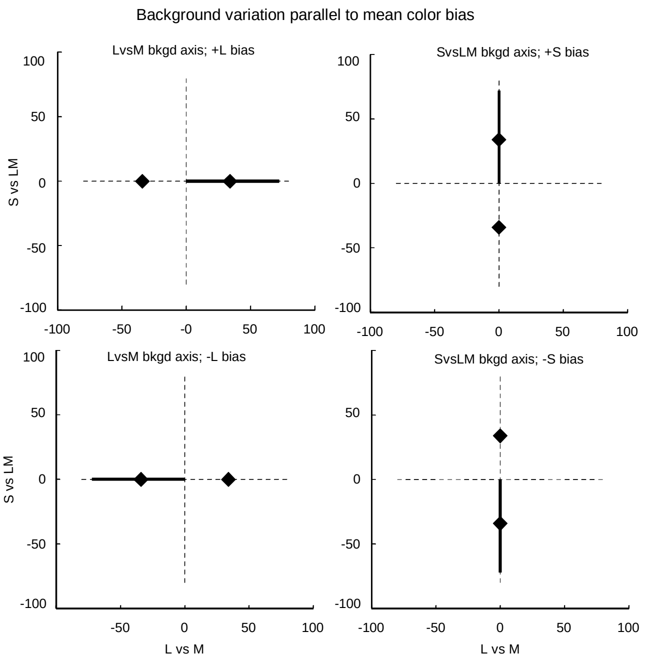


Fig. 4. Mean adapting colors (diamonds) and background colors (lines) when targets had to be detected by the color differences along an axis orthogonal to the axis of adaptation

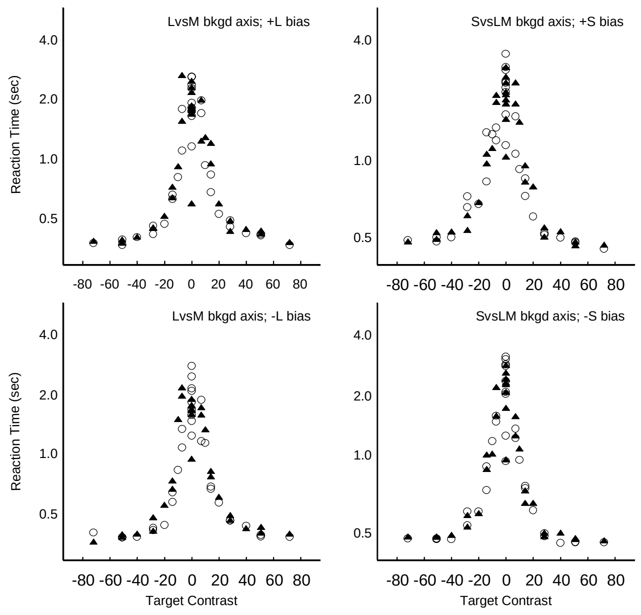


Fig. 5. Search times as a function of the target color contrast relative to the background axis after adapting to the mean background color (circles) or the opposite color bias (triangles)

Figures 4 and 5 show the results for a second set of conditions, in which the adapting axis is now parallel to the axis of the mean color shift. In this case there is no cost or benefit of changing the state of adaptation. Considering again the conditions for the top left panel, the lack of an effect is because the target colors are detected based on the responses along the S-cone axis, yet the adaptation is only changing the sensitivity in the L and M cones. These results illustrate that how adaptively changing images (or observers) will alter performance requires models that are appropriately informed about the actual response changes underlying visual adaptation.

5 Conclusions

Incorporating the actual coding processes of the human visual system into image processing algorithms can significantly improve image quality (e.g. [29] [30]). In this paper we have argued that visual displays and perceptual performance can potentially be enhanced by mimicking the sensitivity changes that occur whenever the visual system is exposed to an image. If the goals of the observer are known and the stimulus cues that support these goals are well defined, then simulating adaptation may confer little advantage, for in that case the appropriate information can readily be extracted, and many approaches already exist for highlighting task-relevant information in displays. Our approach is instead suited to more open-ended contexts, in which the goals of the observer and the potentials in the image are not always clearly defined. For such contexts it seems likely that the best way to help the observer is to facilitate the very processes that the visual system has evolved to optimize perception.

Examples where this approach might prove especially helpful include tasks that require scanning and interpreting highly biased image sets or highly variable sets. In the former case the observer may be confronted with the equivalent of an "other image effect," in which the bias in the images masks the underlying differences between them, while in the latter the state of adaptation may be unable to track changes in the images at each moment. Another example where pre-adapting the image might prove useful is when the observer is looking for anomalies in image features. As we noted above, one of the postulated functions of adaptation is to enhance the salience of novel properties in the environment, and such effects could be especially relevant in interpreting medical images. In fact, fundus photos of the eye are often pre-adapted to remove the average color bias so that color anomalies are more readily visible, and analogous adjustments could potentially be applied to pre-adapt the spatial structure of images, for example in x-ray images. A further potential clinical application is adapting images to adjust for visual losses owing to injury or disease in order to help the observer to adjust to their changed vision. Finally, simulating realistic adaptation transforms could also prove valuable for extending the sensory limits of vision into different spectral or spatial or temporal frequencies, since these transforms could adjust the stimulus ranges in ways that the visual system is designed to encode.

An obvious question is how quickly an observer can themselves adapt to changes in a display, since this might indicate how beneficial a change in the image could be. While some adaptive adjustments occur very rapidly, others have a remarkably long time course [31]. Thus in principle adapting images may be equivalent to weeks or even months of visual training. The types of perceptual training and visual tasks that could be facilitated are limited now primarily by the limits on current models of perceptual adaptation.

Acknowledgments

Supported by a NASA Nevada EPSCOR Collaborative Pilot Grant and by National Eye Institute Grant EY-10834.

References

1. Webster, M.A.: Pattern selective adaptation in color and form perception. In: Chalupa, L.M., Werner, J.S. (eds.) The Visual Neurosciences, vol. 2, pp. 936–947. MIT Press, Cambridge (2003)
2. Webster, M.A., MacLin, O.H.: Figural after-effects in the perception of faces. Psychonomic Bulletin and Review 6, 647–653 (1999)
3. Leopold, D.A., et al.: Prototype-referenced shape encoding revealed by high-level aftereffects. Nature Neuroscience 4, 89–94 (2001)
4. Webster, M.A., et al.: Adaptation to natural facial categories. Nature 428, 558–561 (2004)
5. Barlow, H.B.: Dark and light adaptation: Psychophysics. In: Jameson, D., Hurvich, L.M. (eds.) Handbook of Sensory Physiology, pp. 1–28. Springer, Heidelberg (1972)

6. Ohzawa, I., Sclar, G., Freeman, R.D.: Contrast gain control in the cat visual cortex. Nature 298, 266–268 (1982)
7. MacLeod, D.I.A., von der Twer, T.: The pleistochrome: optimal opponent codes for natural colours. In: Mausfeld, R., Heyer, D. (eds.) Colour Perception: Mind and the Physical World, pp. 155–184. Oxford University Press, Oxford (2003)
8. Laughlin, S.B.: Form and function in retinal processing. Trends in Neuroscience 10, 478–483 (1987)
9. Webster, M.A., Mollon, J.D.: Colour constancy influenced by contrast adaptation. Nature 373, 694–698 (1995)
10. Graham, N.: Visual Pattern Analyzers. Oxford University Press, Oxford (1989)
11. Field, D.J.: Relations between the statistics of natural images and the response properties of cortical cells. Journal of the Optical Society of America A 4, 2379–2394 (1987)
12. Barlow, H.B.: A theory about the functional role and synaptic mechanism of visual aftereffects. In: Blakemore, C. (ed.) Vision: coding and efficiency, pp. 363–375. Cambridge University Press, Cambridge (1990)
13. Pokorny, J., Shevell, S., Smith, V.C.: Colour appearance and colour constancy. In: Gouras, P. (ed.) Vision and Visual Dysfunction 6: The Perception of Colour, Macmilan, London (1991)
14. Werner, J.S., Schefrin, B.E.: Loci of achromatic points throughout the life span. Journal of the Optical Society of America A 10, 1509–1516 (1993)
15. Webster, M.A., Georgeson, M.A., Webster, S.M.: Neural adjustments to image blur. Nature Neuroscience 5, 839–840 (2002)
16. Srinivasan, M.V., Laughlin, S.B., Dubs, A.: Predictive coding: A fresh view of inhibition in the retina. Proceedings of the Royal Society of London B 216, 427–459 (1982)
17. Leopold, D.A., Bondar, I.: Adaptation to complex visual patterns in humans and monkeys. In: Clifford, C.W.G., Rhodes, G. (eds.) Fitting the Mind to the World: Adaptation and Aftereffects in High Level Vision, pp. 189–211. Oxford University Press, Oxford (2005)
18. Rhodes, G., et al.: Fitting the mind to the world: Face adaptation and Attractiveness Aftereffects. Psychological Science 14, 558–566 (2003)
19. Hood, D.C., Finkelstein, M.A.: Sensitivity to light. In: Boff, K.R., Kaufman, L., Thomas, J.P. (eds.) Handbook of Perception and Human Performance, Sensory Processes and Perception, vol. 1, pp. 5-1– 5-66. Wiley, New York (1986)
20. Georgeson, M.A.: The effect of spatial adaptation on perceived contrast. Spatial Vision 1, 103–112 (1985)
21. Clifford, C.W.G.: Perceptual adaptation: motion parallels orientation. Trends in Cognitive Sciences 6, 136–143 (2002)
22. Walraven, J., et al.: The control of visual sensitivity: Receptoral and postreceptoral processes. In: Spillmann, L., Werner, J.S. (eds.) Visual Perception The Neurophysiological Foundations, pp. 53–101. Academic, San Diego (1990)
23. Atick, J.J., Li, Z., Redlich, A.N.: What does post-adaptation color appearance reveal about cortical color representation? Vision Research 33, 123–129 (1993)
24. Barlow, H.B., Földiák, P.: Adaptation and decorrelation in the cortex. In: Durbin, R., Miall, C., Mitchison, G.J. (eds.) The Computing Neuron, pp. 54–72. Addison-Wesley, Wokingham (1989)
25. Clifford, C.W.G., Wenderoth, P., Spehar, B.A: functional angle on adaptation in cortical vision. Proceedings of the Royal Society of London B 267, 1705–1710 (2000)
26. Zaidi, Q., Shapiro, A.G.: Adaptive orthogonalization of opponent-color signals. Biological Cybernetics 69, 415–428 (1993)

27. McDermott., K., et al.: Visual Search and eye movements in novel and familiar contexts. In: Rogowitz, B.E., Pappas, T.N., Daly, S.J. (eds.) SPIE 6057: Human Vision and Electronic Imaging XI, p. 60570C-1-12 (2006)
28. MacLeod, D.I.A., Boynton, R.M.: Chromaticity diagram showing cone excitation by stimuli of equal luminance. Journal of the Optical Society America 69, 1183–1186 (1979)
29. Van Hateren, J.H.: Encoding of high dynamic range video with a model of human cones. ACM Trans. Graphics 25, 1380–1399 (2006)
30. Meylan, L., Alleyson, D., Susstrunk, S.: Model of retinal local adaptation for the tone mapping of color filter array images. Journal of the Optical Society of America A 24, 2807–2816 (2007)
31. Delahunt, P.B., et al.: Long-term renormalization of chromatic mechanisms following cataract surgery. Visual Neuroscience 21, 301–307 (2004)

Wavelet-Based Stratified Irradiance Caching for Efficient Indirect Illumination

Matt Berger[1] and Lijun Yin[2]

[1] Air Force Research Laboratory
[2] SUNY Binghamton

Abstract. This paper presents a new method for efficient computation of indirect lighting in local lighting environments. We establish a separation between the BRDF and the irradiance for each vertex, such that during runtime we are able to quickly reconstruct per vertex irradiance. In reconstructing irradiance, we establish an important relationship between three components: stratified irradiance shared across vertices, the fast wavelet transform, and a wavelet-based nonlinearly approximated inner product. By nonlinearly approximating the BRDF for each vertex, we demonstrate how stratified irradiance has spatial independence in the 2D Haar wavelet domain, in turn allowing for large extents of irradiance samples contributing to many vertices. By expressing irradiance in terms of shared scaling coefficients, we introduce an efficient algorithm for evaluating the inner product between the irradiance and the BRDF. Our system is tailored towards the interactive rendering of static but geometrically complex models which exhibit complex reflectance materials, capable of interactive lighting and interactive view under frame rates of 2-6 fps, ran entirely on a single CPU.

1 Introduction

Achieving interactive global illumination in local lighting environments still remains a difficult task. In these types of environments, interactivity refers to the interactive modification of the lighting, viewpoint, geometry, and surface reflectance. Existing approaches suffer a variety of trade-offs in achieving interactivity, ranging from: gross approximations in simulating global illumination [1][2], incurring heavy precomputation cost [3], or limiting interactivity in order to resolve these types of drawbacks [4][5].

The focus of our work is in achieving a middle ground between these trade-offs, such that we strive to find a good balance between interactivity, reasonable precomputation, and accurately modeling light transport. We have thus developed a system which supports interactive lighting and interactive view for moderately complex geometry and BRDFs, while physically modeling diffuse interreflections, under modest precomputation time.

In achieving interactive global illumination, the main contribution of this paper is in compactly representing and efficiently reconstructing local irradiance. In computing radiance resulting from indirect illumination, we separate the irradiance from the BRDF for every vertex in the scene. By wavelet projecting the

BRDF and applying nonlinear approximation, we establish an important relationship between three components: shared stratified irradiance, the fast wavelet transform, and the nonlinear wavelet-based inner product. We illustrate how the 2D Haar wavelet-based nonlinear approximation of the BRDF can be used as a means of exploiting irradiance similarity amongst vertices. By utilizing these properties, we show how per-vertex irradiance may be efficiently reconstructed at runtime, in a form that is amenable to performing a very fast inner product with the BRDF. Our system is particularly effective in rendering moderately complex geometric models, of up to 80,000 triangles, composed of glossy BRDFs.

The paper is organized as follows. Section 2 covers all previous work related to our approach. Section 3 introduces a new method of efficiently computing the nonlinear wavelet-based inner product from the fast wavelet transform. In Section 4 we establish the relationship between compact support in wavelets and irradiance similarity. Section 5 details our system implementation, while Section 6 shows our results and analysis. Section 7 discusses a number of interesting directions from which our approach may take, and Section 8 concludes the paper.

2 Related Work

A significant amount of work has been devoted to achieving interactive global illumination in recent years. Here is a brief review of the existing approaches which are most relevant to our work.

2.1 PRT w/ Distant Lighting

Precomputed radiance transfer (PRT) originated in the context of distant lighting [6][7][8][9], where lighting is only an angular function. The basic idea behind PRT is to establish a separation between quantities that are static and dynamic. The static quantities are precomputed and stored away, such that during runtime, the dynamic quantities may be quickly computed and combined with the static quantities to produce radiance. In storing the raw static illumination data, it is typically the case that memory usage will be much too large, and the computation time involved in evaluating the radiance will be excessive. Illumination quantities are hence projected to an orthonormal basis in order to obtain high data compression, as well as effectively utilize properties of the orthonormal basis such that the integral of the product of multiple functions is still reasonably efficient to compute.

2.2 PRT w/ Local Lighting

The extension of PRT into local lighting environments is nontrivial. It becomes very difficult to obtain an effective separability of illumination quantities when lighting is a function of both position and direction. The work of [3] demonstrates a very coarse separability by precomputing full global illumination solutions at every vertex of the scene, in all directions, over the entire space of where point lights may be positioned. This is followed by several stages of aggressive data compression. The approach produces very impressive results, in allowing for interactive lighting and view under low-frequency glossy BRDFs, but only at the expense of very costly precomputation.

More recent approaches have targeted the disadvantage of large precomputation time, but only at the expense of limiting the interactivity. [10] focuses on precomputing light paths, and porting their algorithm onto the GPU for real time results. However, their approach only supports diffuse BRDFs and very simple geometry. [5] develop a 4D radiance transport operator, where transport is expressed in both the angular and spatial domain. They demonstrate very efficient precomputation, and allow for glossy BRDFs, but only support geometry consisting of large, flat quads, as they must define basis functions on the geometry itself.

Our work is most similar to the work on direct-to-indirect transfer [4]. The basic idea is to precompute the indirect transfer of a set of samples (gather samples), and how they contribute to a set of samples to be shaded. Our approach also reconstructs indirect illumination from a set of direct illumination samples, but instead of representing the incident indirect illumination transfer in a global context as in [4], we represent it locally per-vertex. A local representation of the BRDF allows for modification of view, whereas a global representation fixes the view.

3 Nonlinear Wavelet-Based Inner Product

This section goes over the mathematical framework of our approach.

Let us start from the rendering equation [11]:

$$L_r(\mathbf{x},\Theta) = \int_{\Omega} L_I(\mathbf{x},\omega) f_r(\mathbf{x},\Theta,\omega) V(\mathbf{x},\omega)(n(\mathbf{x})\cdot\omega)d\omega \tag{1}$$

Here, $\mathbf{x}$ is the point we are interested in computing radiance L_r for in direction Θ, resulting from indirect illumination. We are integrating over the differential solid angle $d\omega$, over the hemisphere oriented at normal $n(\mathbf{x})$. L_I is incident radiance at $\mathbf{x}$ in direction ω, f_r is the BRDF defined in the local frame at $n(\mathbf{x})$, V is the visibility function, and $(n(\mathbf{x})\cdot\omega)$ is the cosine term.

In computing radiance stemming from indirect illumination, assume that we are only interested in single-bounce diffuse interreflections, and the scene is in a closed environment, which reduces $V(\mathbf{x},\omega)$ to a constant of 1. Furthermore, assume that local lighting is defined by a point light source. We may express radiance computation as:

$$L_r(\mathbf{x},\Theta) = \int_{\Omega} L_D(\mathbf{x},\omega) f_r(\mathbf{x},\Theta,\omega)(n(\mathbf{x})\cdot\omega)d\omega \tag{2}$$

Here L_D is the incoming radiance function, which represents direct illumination from a point visible at $\mathbf{x}$ from direction $-\omega$. L_D may be defined as:

$$L_D(\mathbf{x},\omega) = L_P(\tilde{\mathbf{x}},\alpha)p(\tilde{\mathbf{x}})(n(\tilde{\mathbf{x}})\cdot\alpha) \tag{3}$$

L_P is the direct illumination of the point light source, $\tilde{\mathbf{x}}$ is the point visible at $\mathbf{x}$ from direction $-\omega$, α is the direction from which the point light emits radiance at $\tilde{\mathbf{x}}$, and p is the diffuse reflectance at $\tilde{\mathbf{x}}$.

In allowing for interactive light movement, we may separate the cosine-weighted BRDF from the irradiance altogether, giving us:

$$\hat{f}_r(\mathbf{x}, \Theta, \omega) = f_r(\mathbf{x}, \Theta, \omega)(n(\mathbf{x}) \cdot \omega) \tag{4}$$

$$L_r(\mathbf{x}, \Theta) = \int_{\Omega} L_D(\mathbf{x}, \omega)\hat{f}_r(\mathbf{x}, \Theta, \omega) d\omega \tag{5}$$

$\hat{f}_r$ is a static quantity and may be precomputed ahead of time, but the incident radiance function L_D will be changing, and will need to be reconstructed. Hence there are two main components to our approach: fast reconstruction of L_D, and fast evaluation of the inner product of L_D and $\hat{f}_r$. The remainder of this section introduces a new method for fast computation of the inner product between these two functions, while the following section goes over an efficient means of reconstructing L_D for each vertex.

3.1 2D Haar Wavelet Basis

Working from the above representation in terms of raw data will result in very demanding memory requirements and expensive computational costs. It is preferable to express the data in terms of an orthonormal basis for data compression and efficient radiance computation. We opt to work in wavelet space, as much of the previous work in PRT [8][4] has shown that wavelets are particularly effective in representing illumination quantities.

We utilize the 2D Haar wavelet basis in representing the illumination quantities. The 2D Haar basis is defined by a mother scaling function $\phi(x, y)$ and three mother wavelet functions: the horizontal wavelet $\psi^H(x, y)$, vertical wavelet $\psi^V(x, y)$, and diagonal wavelet $\psi^D(x, y)$. Refer to Figure 1 for a visual illustration of all four functions.

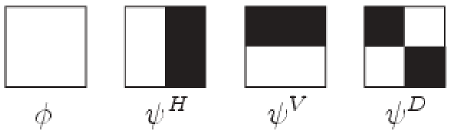


Fig. 1. 2D Haar mother scaling function and wavelet functions

In constructing the basis, we opt to use nonstandard decomposition of the mother scaling and wavelet functions. This forms a complete quad tree of functions, where the root of the tree is the mother scaling function, and the remaining nodes in the tree represent the dilated and translated wavelet functions. Both $\hat{f}_r$ and L_D may now be expressed in 2D Haar wavelets by parameterizing the hemispherical domain to an axis-aligned cube map, and imposing the 2D Haar basis on each cube map face as in [8].

3.2 Clustered Inner Product

Suppose that $\hat{f}_r$ has been projected to the 2D Haar wavelet basis, but L_D has not - it is dynamic. During precomputation we may nonlinearly approximate $\hat{f}_r$, by retaining a small number of the wavelet coefficients resulting from the wavelet projection. As demonstrated in [9], a wavelet approximation in $\hat{f}_r$ is particularly sparse, requiring only $0.1\% - 1\%$ of the wavelet coefficients to obtain 99% accuracy. A result of the high sparsity in the BRDF approximation is the necessity to only compute the irradiance wavelet coefficients which correspond to $\hat{f}_r$. Regardless of the complexity of the irradiance, the details will be limited according to the approximation of $\hat{f}_r$.

In the fast computation of the required wavelet coefficients, we define the *scaling coefficient quad tree* (SCQT) for L_D, which is a quad tree consisting of scaling coefficients for all dilations and translations. If we make the assumption that this can be quickly constructed per vertex (to be described in the next section), then we may compute wavelet coefficients *on-demand*, according to $\hat{f}_r$. We may then utilize the tree-like structure of the basis in fast computation of the wavelet coefficients.

While this method is rather efficient, it is still a good $3x$ slower than performing a standard inner product between two vectors. This is due to $\hat{f}_r$ leading the iteration in computing the wavelet coefficients for L_D. If we instead reverse the problem, and have L_D lead the iteration, then we obtain a much more efficient inner product solution which does not require the computation of L_D's wavelet coefficients.

Consider nonlinearly approximating $\hat{f}_r$ on a per wavelet node basis. That is, the nonlinear approximation is made at the discretization of the wavelet node, instead of the wavelet coefficient, where a wavelet node is defined by its translation and dilation in the wavelet decomposition. All three wavelets (horizontal, vertical, and diagonal) must be retained for each node.

From this representation, at each node we may store partial sums of the wavelet coefficients, instead of the coefficients themselves. Assuming $c^W(\hat{f}_r)$ where $W \in \{H, V, D\}$ indicates a particular wavelet coefficient for the BRDF, we obtain:

$$\begin{aligned} c_{ll} &= c^H(\hat{f}_r) + c^V(\hat{f}_r) + c^D(\hat{f}_r) \\ c_{lr} &= c^V(\hat{f}_r) - c^H(\hat{f}_r) - c^D(\hat{f}_r) \\ c_{ul} &= c^H(\hat{f}_r) - c^V(\hat{f}_r) - c^D(\hat{f}_r) \\ c_{ur} &= c^D(\hat{f}_r) - c^H(\hat{f}_r) - c^V(\hat{f}_r) \end{aligned} \tag{6}$$

The contribution of the node to the inner product is reduced to:

$$R(d, i, j) = \phi_{ll} \cdot c_{ll} + \phi_{lr} \cdot c_{lr} + \phi_{ul} \cdot c_{ul} + \phi_{ur} \cdot c_{ur} \tag{7}$$

$\phi_{ll}, \phi_{lr}, \phi_{ul}, \phi_{ur}$ represent L_D's scaling coefficients at dilation $d+1$ that fall into the support of the wavelet node at dilation d, translation (i, j).

The complexity of this method is quite comparable to the complexity of the standard inner product. The only disadvantage is a coarser approximation of $\hat{f}_r$, in that we must approximate at the discretization of wavelet nodes.

4 Wavelet-Based Stratified Irradiance

Either of the inner product methods will only be effective if we are able to quickly reconstruct the irradiance for every vertex. This section establishes an important relationship between the compact support of wavelets and irradiance similarity exhibited in most scenes.

4.1 Spatially Independent Stratified Irradiance

Most global illumination algorithms utilize irradiance similarity in some form. In most scenes, irradiance will vary rather slowly across surfaces, hence much computational efficiency will be gained if this property is exploited. Exploiting this in terms of shared point lights, as in the various virtual point light algorithms [12][1], does not fully exploit the irradiance similarity that most scenes exhibit. Representing incident radiance in a spatially contiguous formulation allows for more efficient and accurate methods of indirect illumination computation. As demonstrated in [13], irradiance may be implicitly clustered for efficient rendering, although their approach is far from interactive.

If we instead *explicitly* cluster irradiance according to the integration domain of every vertex in the scene, we may obtain a very efficient means of computing irradiance at runtime. We may group incident radiance samples together which contribute to the spatially contiguous subdomains of a subset of vertices of a scene. We term this *shared stratified irradiance*, in that irradiance computation for a single vertex is broken up into disjoint subdomains, where each subdomain of that vertex contributes to subdomains of other vertices.

Assuming the aforementioned cube map representation for irradiance, where the 2D Haar wavelet basis is imposed on each cube map face, we may be able to effectively represent shared stratified irradiance. We define a *Haar face* as being a set of radiance samples which completely cover the support of a particular basis function. The integration of a Haar face gives us the irradiance for that subdomain.

Stratified irradiance is formed by deriving Haar faces for which vertices share. However, note that the irradiance samples that reside in a Haar face need not contain the same ordering if the corresponding BRDF approximation for each vertex does not require that Haar face. Refer to Figure 2 for an example. We term the samples that compose a Haar face under this situation as being *spatially independent*. In this formulation, we obtain particularly high shared irradiance, as a large group of samples may contribute to a large number of vertices. The next section goes over a method of deriving Haar faces for shared stratified irradiance.

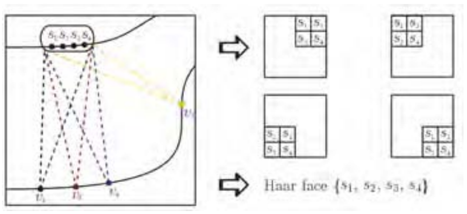


Fig. 2. Shared Stratified Irradiance. For each Haar face, the sample configuration need not be preserved if the corresponding BRDF for each vertex doesn't require it.

4.2 Stratified Irradiance Caching

We have developed a conservative greedy approach to deriving Haar faces. We splat incident radiance samples that compose a Haar face for a particular vertex. We then iterate over all vertices to derive other Haar faces from which these samples may form.

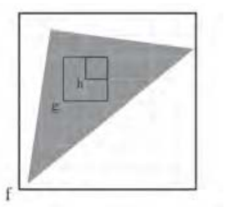


Fig. 3. Demonstrates how the projected surface area formed from one Haar face may be used in culling other Haar faces from processing

In deriving other Haar faces from a set of samples, visibility tests must be performed to verify that those samples indeed map into a Haar face. This is a very expensive operation however, so we use an efficient method of culling vertices, based on comparing the projected surface areas formed from Haar faces that map into the radiance samples.

We note that if the projected surface areas of two Haar faces from two separate vertices significantly differ, it is unlikely (and undesirable) for them to share stratified irradiance. Refer to Figure 3 for an example. In the figure, a triangle spans Haar faces f, g, and h, where each Haar face belongs to a unique vertex. Since the projected surface areas of g and h are relatively similar, ignoring visibility, the vertices are likely to share the same samples. However, since the projected surface areas of f and h differ greatly, it is very unlikely for them to share the same samples. Only if the samples are embedded entirely in h, would they share samples.

Hence we cull those vertices which do not meet this criteria, and then perform visibility tests on the rest to verify the derivation. This method of selection is similar to how distance metrics are applied in choosing virtual point light samples for [1][2] (although visibility is neglected in their algorithms). Using the projected surface area, however, provides for a better measure of irradiance similarity than just distance, as the orientation of the radiance samples also has an effect on similarity.

Projected surface areas are also used to effectively select Haar faces to derive vertices from. Note that certain samples which compose a Haar face may contribute to very little vertices; corners of a scene are prime examples. In order to provide a sufficient measure of projected surface areas, we blanket the scene with a set of samples, and bucket projected surface area intervals for each sample. Each bucket value represents a range of projected surface area values, and the number of vertices whose Haar faces contain projected surface areas in that range. The set of samples are stored in a kd-tree for efficient lookup; as Haar faces are generated, the kd-tree is queried for the vertex frequencies, and if the value is lower than a certain threshold, the Haar face is not considered for processing. Nearby geometry is thus naturally excluded from processing using this method of selection.

The last portion of the algorithm is in efficiently and effectively iterating over Haar faces, and knowing when to terminate the iteration. We note that iterating over every Haar face of every vertex is prohibitively expensive; it is desirable to quickly converge to a solution by wisely choosing Haar faces to process. Hence we cluster vertices according to irradiance similarity, using the algorithm of [14]. A hierarchical k-means algorithm is used to form clusters, where a cluster of vertices is split if: the bounding sphere formed from the geometry exceeds a threshold, or a sample for this cluster lies within the cluster. We then iterate over these clusters, instead of vertices, for generating Haar faces, only selecting a small number of vertices from each cluster.

5 System Implementation

This section covers the implementation details of our system, split between a precomputation component and a runtime component.

5.1 Precomputation

As part of precomputation we first setup the cosine-weighted BRDF $\hat{f}_r$ for each vertex. We parameterize the BRDF to each cube map face, scale for the solid angle, and perform a fast wavelet transform for each face. We then gather all wavelet coefficients and nonlinearly approximate the BRDF, depending on the type of inner product method used. We sort the wavelet nodes according to the summation of the area-weighted magnitude of each horizontal, vertical, and diagonal wavelet coefficient.

We then derive the shared stratified irradiance, as described in 4.2. Our current implementation uses a $(6 \times 32 \times 32)$ cube map for parameterizing irradiance, such that the total number of samples used to approximate the integration is bound at 3072. We perform the algorithm on (2×2) Haar faces first, as clustering will be most effective when considering smaller subsections of stratified irradiance. From the (2×2) Haar faces, we perform bottom-up clustering to derive (4×4) Haar faces. We then store away the irradiance samples, (2×2) Haar faces, and (4×4) Haar faces, while the vertices store references to the Haar faces. This sets up the local irradiance functions for every vertex, where the irradiance is composed of shared scaling coefficients.

Recall from Section 4.1 that scaling coefficients may be cached away without regard to their spatial configuration, so long as they are not directly used in computing the inner product. For diffuse BRDFs and low-glossy BRDFs, the wavelets retained are typically at rather low frequencies, hence the property of spatial independence may be fully utilized in these cases. For BRDFs that contain particularly sharp specular lobes, the spatial property of cached Haar faces will likely be needed; hence, we store the Haar faces themselves within the BRDF of each vertex. Unfortunately, this fixes the view for this range of BRDFs. We may separate the BRDF from each vertex altogether, as in [9], however we would lose the spatial information necessary to evaluate the approximation.

5.2 Runtime

At runtime we first compute direct illumination on the incident radiance samples. Currently our system supports omnidirectional point lights and spotlights. This is accomplished through a shadow mapping implementation on the GPU. The shadow mapping algorithm produces aliasing artifacts as shown in our results. Shadow mapping extensions, such as perspective shadow maps [15], may be used in resolving the aliasing artifacts.

Once the direct illumination has been computed, we may quickly reconstruct the irradiance for every vertex. The global pool of scaling coefficients derived from the precomputation are first computed. At this point, the computation rests on each vertex traversing its scaling coefficient quad tree from the shared scaling coefficients, in order to fully reconstruct the irradiance. For each vertex, the quad tree of scaling coefficients may be handed to the nonlinearly approximated BRDF for fast evaluation of the inner product.

Our system also handles multi-bounce diffuse interreflections, albeit at the cost of interactivity. We assign for each radiance sample the vertex closest to it (the nearest neighbor vertex), by the smallest euclidean distance. For each light bounce iteration we perform the above algorithm in computing the indirect illumination for each vertex. Before the pass of each iteration, for each sample we accumulate its nearest neighbor vertex's indirect illumination to its own value. A more accurate accumulation of indirect illumination may be achieved by storing the k-nearest vertices for each sample, and performing a density estimation on the vertex indirect illumination.

6 Results

This section presents our results and analysis for the various scenes rendered with our algorithm. Computing the direct illumination via shadow mapping is the only aspect of our algorithm ran on the GPU; everything else is run on a single 3.2 GHz CPU with 2GB of RAM, implemented entirely in Java. We opt to use single-bounce indirect illumination for all scenes; multi-bounce indirect illumination contributes very little to the overall illumination, and is significantly more expensive to compute.

Figure 4 shows 4 scenes rendered with our algorithm, and table 1 displays the statistics for the 4 scenes. In all scenes we retain 10 wavelet nodes for each BRDF. As the statistics show, the SCQT computation is the bottleneck at runtime. In further analyzing the SCQT computation, we find that the time spent on computing the global pool of scaling coefficients comprises a small portion of the runtime; most time is spent on traversing the per-vertex quad trees. We find that having all vertices contain references to a global set of scaling coefficients will result in poor cache coherency when individually traversing each quad tree. Future work entails more effective memory organization of the shared scaling coefficients, as well as different means of constructing the individual SCQTs.

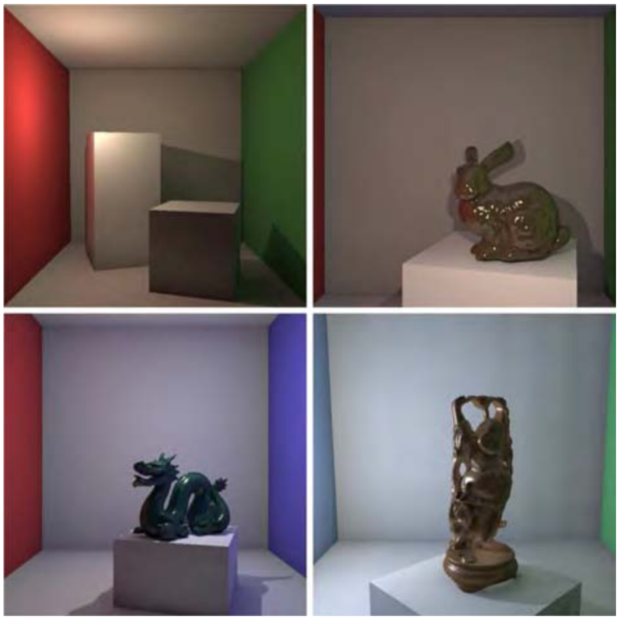


Fig. 4. Sample scenes rendered with our system

7 Limitations and Future Work

Our system is particularly effective in the rendering of complex models (the Stanford models, for instance), but is ineffective for scenes containing high occlusion, as the irradiance similarity will be poor. However, the algorithm of [5] is orthogonal to that of ours, in that it is effective for scenes of high occlusion, such as

Table 1. Statistics for the four scenes. SCQT refers to the amount of time taken for computing the scaling coefficients each frame, while IP refers to the amount of time involved in computing the inner product per frame.

Scene	Tris	Precomp	Memory	Runtime	SCQT	IP
Box	10K	1.6 hours	20 MB	5.3 fps	180 ms	9 ms
Bunny	80K	3.8 hours	160 MB	2 fps	430 ms	60 ms
Dragon	56K	2.6 hours	110 MB	2.6 fps	330 ms	50 ms
Buddha	75K	3.3 hours	145 MB	1.7 fps	510 ms	70 ms

architectural environments, but is unable to handle complex models. We would like to combine both approaches in order to develop a system capable of rendering a wide variety of scenes, while still maintaining reasonable precomputation times.

As shown in our results, the SCQT computation is by and large the bottleneck of our framework. Utilizing temporal coherence in the lighting would greatly speed up this portion of the algorithm, at very little accuracy degradation. We may apply the recent work of [16] in selectively updating scaling coefficients for each vertex. We may reuse coarser scaling coefficients as lighting is moved, while focusing on finer scaling coefficients in representing the sharper details.

Since the current framework caches irradiance, it is restricted to diffuse interreflections; however, it would not take much to extend the work of [17] in allowing for glossy interreflections. In applying a separable BRDF approximation to the interreflections, we may separate the view-dependent and light-dependent terms from the BRDF such that the view-dependent terms are setup during precomputation, while the light-dependent terms are computed at runtime.

8 Conclusion

We have presented a new technique for efficient computation of indirect illumination. We have established an important relationship between the irradiance similarity exhibited in most scenes, and the fast wavelet transform, allowing for efficient reconstruction of per-vertex irradiance. Furthermore, we showed how the irradiance representation may be utilized in efficiently computing the inner product between the irradiance and the BRDF. Our system is tailored towards interactive lighting under global illumination in scenes composed of complex, glossy models. We believe that efficiently and compactly representing local irradiance has a wide variety of interesting research directions to be taken.

References

1. Dachsbacher, C., Stamminger, M.: Reflective shadow maps. In: Proceedings of the 2005 symposium on Interactive 3D graphics and games, pp. 203–231 (2005)
2. Dachsbacher, C., Stamminger, M.: Splatting indirect illumination. In: Proceedings of the 2006 symposium on Interactive 3D graphics and games, pp. 93–100 (2006)

3. Kristensen, A., Akenine-Möller, T., Jensen, H.: Precomputed local radiance transfer for real-time lighting design. In: Proceedings of ACM SIGGRAPH 2005, vol. 24, pp. 1208–1215 (2005)
4. Hašan, M., Pellacini, F., Bala, K.: Direct-to-indirect transfer for cinematic relighting. ACM Transactions on Graphics (TOG) 25, 1089–1097 (2006)
5. Kontkanen, J., Turquin, E., Holzschuch, N., Sillion, F.: Wavelet Radiance Transport for Interactive Indirect Lighting. Rendering Techniques 2006 (Eurographics Symposium on Rendering) (2006)
6. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 527–536 (2002)
7. Sloan, P., Hall, J., Hart, J., Snyder, J.: Clustered principal components for precomputed radiance transfer. ACM Transactions on Graphics 22, 382–391 (2003)
8. Ng, R., Ramamoorthi, R., Hanrahan, P.: All-frequency shadows using non-linear wavelet lighting approximation. ACM Transactions on Graphics (TOG) 22, 376–381 (2003)
9. Ng, R., Ramamoorthi, R., Hanrahan, P.: Triple product wavelet integrals for all-frequency relighting. ACM Transactions on Graphics (TOG) 23, 477–487 (2004)
10. Szécsi, L., Szirmay-Kalos, L., Sbert, M.: Light animation with precomputed light paths on the GPU. In: Proceedings of the 2006 conference on Graphics interface, pp. 187–194 (2006)
11. Kajiya, J.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 143–150 (1986)
12. Keller, A.: Instant radiosity. Fachbereich Informatik, Univ. (1997)
13. Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., Greenberg, D.: Lightcuts: a scalable approach to illumination. In: International Conference on Computer Graphics and Interactive Techniques, pp. 1098–1107 (2005)
14. Arikan, O., Forsyth, D., O'Brien, J.: Fast and detailed approximate global illumination by irradiance decomposition. Proceedings of ACM SIGGRAPH 2005 24, 1108–1114 (2005)
15. Stamminger, M., Drettakis, G.: Perspective shadow maps. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 557–562 (2002)
16. Overbeck, R., Ben-Artzi, A., Ramamoorthi, R., Grinspun, E.: Exploiting Temporal Coherence for Iincremental All Frequency Relighting. In: Eurographics Symposium on Rendering (2006)
17. Wang, R., Tran, J., Luebke, D.: All-frequency relighting of glossy objects. ACM Transactions on Graphics (TOG) 25, 293–318 (2006)

Efficient and Realistic Cumulus Cloud Simulation Based on Similarity Approach

Be Wang[1], Jingliang Peng[2], Youngmin Kwak[1], and C.-C. Jay Kuo[1]

[1] Department of Electrical Engineering
University of Southern California, Los Angeles, CA, 90089-2564
{beiwang,youngmik}@usc.edu, cckuo@sipi.usc.edu
[2] Department of Computer Science
Sun Yat-sen University, Guangzhou 510275, P.R. China
jingliap@gmail.com

Abstract. Simulation of 3D clouds is an important component in realistic modeling of outdoor scenes. In this paper, we propose a novel physically driven cumulus cloud simulation algorithm based on the similarity approach. By using the similarity approach, the overall cloud characteristics is captured with a set of constant parameters, which in turn enables decoupling of the 3D cloud simulation into the 1D vertical and the 2D horizontal simulations. As a result, the proposed cloud simulation algorithm greatly facilitates computing efficiency, general shape control and wind effect simulation, while yielding realistic visual quality.

Keywords: similarity approach, decoupling, cloud simulation.

1 Introduction

Clouds are pictures in the sky. They look so fantastic and are so close to our daily life that realistic cloud simulation and rendering become inseparable parts in many 3D applications, such as high-quality 3D film production and realistic 3D flight simulation. At the same time, infinite variations in cloud shape and cloud appearance make cloud simulation and rendering a big challenge, in terms of both computation and reality. This has motivated intensive research in the field of 3D cloud simulation and rendering. The major focus of this work is on efficient and realistic cloud simulation. Particularly, we simulate a special type of clouds, the cumulus clouds, as most previous works do.

Among previous cloud simulation algorithms, some are based on physics rules and require high computation, the others reduce the computation with procedural approaches but yield less flexibility in dynamics. The cloud simulation algorithm proposed in this work simulates clouds based on physics rules, which enables flexible control of general cloud shape; meanwhile, it facilitates efficient computation, especially with current parallel computing and GPU technologies. To be more specific, the most distinguished features of the proposed cloud simulation algorithm include:

– **Efficient computation.** Based on the similarity approach, the 3D cloud simulation is decoupled into 1D vertical and 2D horizontal simulations, which facilitates efficient computation in the following aspects:
  - Reduced cost in each simulation step. After the simulation decoupling, the vertical simulation, as described later in this paper, costs negligible computing power. In other words, the computation along one dimension is almost removed from the 3D cloud simulation.
  - Suitable for parallel computing. After the simulation decoupling, the 2D simulations can be potentially performed in parallel.
  - Potential for GPU acceleration. The 2D simulations can be potentially accelerated by efficient 2D fluid solvers on GPUs.

– **Flexible general shape control.** This is enabled by the fact that a set of constant parameters used with the similarity approach directly correspond to the shape characteristics of a cloud.

2 Related Works

Research on cloud simulation and rendering started as early as more than two decades ago. Previous methods can be classified into two main categories: visually driven and physically driven method.

Visually driven methods do not simulate clouds following physics process. Instead, they model clouds based on the amorphous shape property. Such methods include procedural modeling [1,2], fractals modeling [3,4,5], qualitative simulations [6] and texture sprite modeling [7]. The above mentioned methods are computationally inexpensive and easy to implement. In these methods, loud shapes can be controlled by various parameters. However, the adjustment of parameters corresponding to cloud shape is often conducted by trial and error, which has limited extension.

On the contrary, physically driven methods simulate clouds based on the physics theory of fluid dynamics. Most of these methods involve intensive computation due to the solving of partial differential equations (PDEs), in particular, the Navier-Stoke's equation. After the milestone work of Stam's stable fluid solver [8], computation in Navier-Stoke's equation solving was greatly reduced by the simplified fluid dynamics. However, his method focuses on the motion of smoke, the important physics process of cloud simulation such as phase transition is not taken into account. Kajiya *et al.* [9] first incorporated the phase transition factor into their cloud simulation scheme. However, their method is computationally expensive. Dobashi [10] developed an efficient method for cloud simulation using cellular automaton. Yet, it is hard to simulate complicated physics process using boolean operations [10]. Miyazaki[11] then used Coupled Map Lattice(CML), an extension of cellular automaton, to simulate various types of clouds. Harris also took the phase transition into account when using Stam's stable fluid solver for fluid dynamics [12]. In his work, he introduced an important concept of potential temperature in cloud simulation and achieved computation speed-up by implementation on hardware. In both [11] and [12], cloud simulation depends on the initial water vapor condition and a set of fluid parameters such as viscosity; no method is provided to control the general shape of cloud based on the convection property of the atmosphere.

In order to address the shortcomings of the previous cloud simulation methods, we propose in this work a physically driven scheme to simulate 3D clouds, especially 3D cumulus clouds. It simulates the physics process including the phase transition process, leads to efficient computation and enables flexible general shape control.

3 Overview of Our Method

According to their appearance, clouds can be classified into three main categories: cumulus, cirrus and stratus. Different types of clouds are dominated by distinct physics processes. Thus, in an alternative classification, we categorize clouds into convection dominant and non-convection dominant clouds. Cumulus clouds bear the extreme property of the convection process while cirrus and stratus clouds are dominated by other physics processes. In this work, we first derive a simplification of the convection process, then focus on the cumulus cloud simulation.

To simplify the convection process, the basic modeling unit of turbulent plume (or plume for short) is adopted, and a cumulus cloud is generated from several plumes. This is motivated by the statement, "Cumulus clouds develop by a series of rising plumes or turrets"[13]. To be specific, plume is *a buoyant jet in which the buoyancy is supplied steadily from a point source and the buoyant region is continuous*[13]. For a plume, we take the assumption that it undergoes in a stable stratified fluid, *i.e.* ,**the radial profiles of velocity and buoyancy are geometrically similar at all heights within the plume**[13]. Furthermore, the atmospheric science [14] justifies that acceleration contributions from different factors in dynamics equation have different orders of magnitude in vertical and horizontal directions. Based on the assumption and justification above, it is desirable to decouple the regular 3D space into 1D vertical and 2D horizontal subspaces and treat them separately. Through similarity analysis, we are able to describe the characteristic of simple fluids with a set of constant parameters, with which to control the cloud shape. The 3D simulation decoupling in our work provides advantages in not only computational efficiency, but also flexible general shape control, as detailed later in this work.

In our decoupling cloud simulation, the user first specifies general cloud shape for simulation by setting constant parameters that describe the atmospheric properties such as buoyancy frequency, heat flux, entrainment inflow coefficient, and implementation parameters such as number of horizonal lattices, number of vertical layers and boundary conditions. Then, we treat the horizontal and the vertical direction separately. In the horizontal direction, Stam's stable fluid solver [8] is implemented to simulate whirl features and details; in the vertical direction, the plume property is derived from the constant parameters. Note that, based on the assumption in similarity at all heights, key-frame horizontal layers can be selectively picked up at different heights, and 2D simulations are performed only on those key-frame layers. Thereafter, intermediate cloud layers are interpolated between key-frame horizontal layers.

During the simulation process, water droplet density values are stored in 3D grid based volume, which can be viewed as a 3D texture so that the 3D texture rendering techniques can be directly applied. In addition, approximate 3D illumination can be efficiently done by casting shadows inside a volume. In this work, the method of self-shadowing as proposed in [15] is adopted.

4 Simplification of Dynamics

Most previous physically driven methods simulate fluid dynamics through solving 3D PDEs, which is usually time-consuming. Here, through the application of similarity theory, we strive to capture the major characteristics of cumulus clouds with several constant parameters, which in turn leads to simplified dynamics in the cloud simulation. Based on the simplified dynamics, we are then able to decouple the regular 3D space into 2D horizontal and 1D vertical. Note that we mainly focus on the simulation of cumulus clouds in fair weather, based on which the assumption of dynamics simplification is made in this section.

4.1 Similarity Solution of Turbulent Plume

The atmosphere is viewed as simple fluid in previous physically driven cloud simulators. 3D governing equations are solved in the form of PDEs. For turbulent plume, the governing equations regarding momentum, heat and mass are [13]:

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho_0}\nabla p + \mathbf{B}\hat{k} + \nu\nabla^2\mathbf{u}; \qquad \frac{d\mathbf{B}}{dt} = \kappa\nabla^2\mathbf{B}; \qquad \nabla\cdot\mathbf{u} = 0 \tag{1}$$

Here, ρ_0, $\mathbf{u}$, p, $\mathbf{B}$, ν and κ are the density, the velocity vector, the pressure, the kinetic viscosity, the buoyancy and the heat diffusion coefficient, respectively.

For simple fluid flows and under certain assumptions, major fluid characteristics can be described by a set of constant parameters without rigorously solving the governing equations. For this purpose, the similarity theory and dimensional analysis is applied in this work. The theory has been successfully applied to the investigation of simple convective plumes and thermals [13,16,17]. It is worthwhile to point out that our similarity solution of plume is based on that in [13] as described in this subsection.

Given a steady fluid, simple dependence of dependent variables on independent variables and some boundary conditions, we can find a *similarity solution* consistent with the governing equations, through the utilization of *dimensional analysis*. Knowing that, we are able to obtain the explicit relationships between the major fluid characteristics and a set of constant parameters. In order to get the similarity solution for plumes in a stable stratified fluid, the following assumptions are made, which are true for typical cumulus clouds in fair weather: 1) The flow is steady. 2) The respective radial profiles of mean vertical velocity and mean buoyancy are similar at all heights. 3) The mean turbulent radial velocity is proportional to the mean vertical velocity at any height. 4)The flow is Boussinesq.

Based on the above assumptions, by integrating the governing equations over the horizontal plane, we get a new set of equations that are consistent with the governing equations. The particular form of radial dependence only affects the numerical coefficients in the result relations. Two typical radial profiles are the 'top-hat' and the 'Gaussian' profiles. In the following derivation, we adopt the particular form of 'top-hat'. In addition, at any height, we denote $u = -\alpha w$ where u, α and w are the mean turbulent radial velocity, a constant proportional to the fractional entrainment of mass, and the mean vertical velocity, respectively. Finally, the new set of equations regarding momentum, heat and mass continuity for turbulent plumes, are:

$$\frac{d}{dz}(\pi R^2 w^2) = \pi R^2 \mathbf{B}; \quad \frac{d}{dz}(\pi R^2 w \mathbf{B}) = -\pi R^2 w N^2; \quad \frac{d}{dz}(\pi R^2 w) = 2\pi\alpha R w \quad (2)$$

In equation 2, a new parameter N is introduced, which is the buoyancy frequency and is a measurement of stratification of atmosphere.

The new set of Equations 2 can be numerically solved to get the dimensionless solution of horizontal extent R, vertical velocity w and Buoyancy B, as explicit analytical functions dependent on the height z, which is plotted in Fig. 1. From Fig. 1, we see that the vertical velocity (w) vanishes at a dimensionless height (z) of 2.8 while the buoyancy (B) first vanishes when $z = 2.1$. In the plume, after the individual particles of fluid overshoot the zero buoyancy level, they decelerate to the zero velocity while at the same time spreading away from the central axis. The spreading bulk forms between the levels of zero buoyancy and zero vertical velocity. The resulting solution leads to a successful representation of the convection process that closely matches the lab experiment data[16,17].

The dimensionless solution for Equations 2 depicts the characteristics of a turbulent plume, *i.e.*, how R, w and B vary with respect to the height z. A turbulent plume has two development stages. In the first one, a buoyancy jet is released from the point source; In the second, the bulk spreads away from the central axis of the plume while the buoyancy jet is still continuously supplied. In

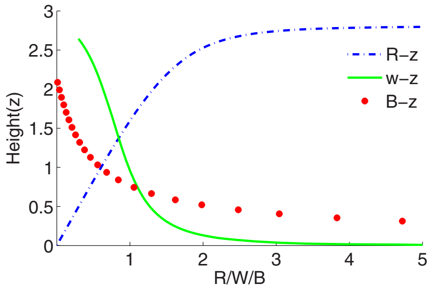


Fig. 1. The height dependence of the dimensionless forms of the horizontal extent R, vertical velocity w and buoyancy B for a turbulent plume

the second stage, water vapor transits to water droplets and a cloud forms. Thus, in our simulation, the focus is on the second stage of the plume's development. Two important variables depict the property of a plume at the second stage. One is the distance z^* between zero buoyancy level and zero vertical velocity level, which confines the plumes' vertical extent thus the cumulus' vertical extent. The other is the radius R^* of a plume when the buoyancy jet first reaches the zero buoyancy level, which in turn determines the approximate base size of a cloud.

In getting the dimensionless solution, the parameter dependence of the equations is simplified by nondimensionalizing the dependent and independent variables. Then, by reversing the process of dimensional analysis, we get the parameter-dependent variables, z^* and R^* dependent on three independent parameters N, F_0 and α as

$$\begin{aligned} z^* &= 2^{-7/8}\pi^{-1/4}\alpha^{-1/2}F_0^{1/4}N^{-3/4}z \\ R^* &= 2^{-1/8}\pi^{-1/4}\alpha^{1/2}F_0^{1/4}N^{-3/4}R \end{aligned} \tag{3}$$

where N, F_0 and α are the buoyancy frequency, the heat flux and a constant which is proportional to the fractional entrainment of mass, respectively; z and R are the dimensionless height and radius obtained from Equations 2; z^* and R^* are the height at the zero vertical velocity level and the radius at the zero buoyancy level corresponding to the specific values of N, F_0 and α.

4.2 Plateau-Like Radial Kernel

In Subsection 4.1, the 'top-hat' radial profile is taken to derive the similarity solution and utilized to describe the major plume characteristics. In order to achieve flexibility in general cloud shape control, it is desirable to model different modes in the variation of R with respect to z within the vertical range of a plume. For this purpose, we adopt a plateau-like radial kernel to approximate all profiles ranging between 'top-hat' and 'Gaussian'. The plateau-like radial kernel, as used in [18] for point-blending, is governed by the following equation:

$$\varphi(r) = e^{\frac{-a\cdot\left(\frac{r}{b}\right)^n}{1-\left(\frac{r}{b}\right)^n}}, \quad \text{for } 0 \le r \le b \tag{4}$$

In Equation 4, b is the radius at the level where buoyancy begins to vanish, a is the width of the kernel, and n controls the slope of drop-off. As shown in Fig. 2, a larger a value makes a narrower kernel while a larger n value generates a more 'top-hat'-like kernel.

Now that the variation of R with respect to the height z is well modeled with the plateau-like radial kernel, we are able to separately simulate the 2D horizontal dynamics at each height, using the Stam's stable fluid solver [8]. In order to run Stam's stable fluid solver, we need to define conditions for water phase transition and governing forces in horizontal dynamics, which are described in Subsection 4.3 and 4.4, respectively.

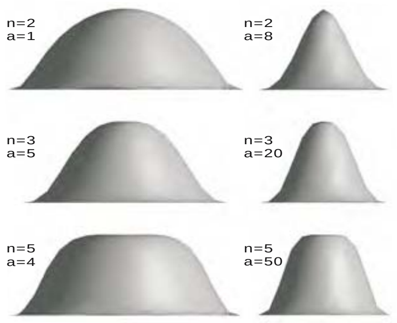


Fig. 2. The plateau-like kernel for water vapor distribution

4.3 Water Vapor Condensation and Related Atmospheric Concepts

Water vapor accumulates continuously with the buoyancy jet, when it reaches a certain amount so called *saturation point*, water phase changes and condensation happens. This saturation point is related to the environmental pressure and temperature which change linearly with height.

The maximum water vapor density [12] is determined by equation 5:

$$w_{max}(T,p) = \frac{380.2}{p} exp\left(\frac{17.67T}{T+243.5}\right) \tag{5}$$

where T is the temperature and p is the pressure. As the density of water vapor exceeds the maximum amount, water vapor transits into water droplets.

4.4 Governing Force in Dynamics

Virtual Temperature and Virtual Temperature Force. Virtual temperature is defined as the temperature that dry air would have if its pressure and density were equal to those of a given moist air[19]. It is approximately calculated by $T_v = T(1 + 0.6q_v)$, where q_v is the density of water vapor. The primary convection process is driven by the buoyancy force from the ground upward along the vertical direction. In the decoupling model, we no longer use the buoyancy force as a direct driving force, since its effect has already been modeled in the plume model and the similarity solution. Along the horizontal dimension, the difference in water vapor density results in the difference in virtual temperature, which in turn drives the movement of water vapor particles. In order to incorporate this effect, we add a new force term along the direction with the biggest local gradient; *i.e.*

$$F_{vt} = \tau(T_v - T_{vi}), \tag{6}$$

where T_v is the virtual temperature of the center voxel in a grid, T_{vi} is the virtual temperature of its neighbor that has the minimum virtual temperature, and τ is a constant scale factor.

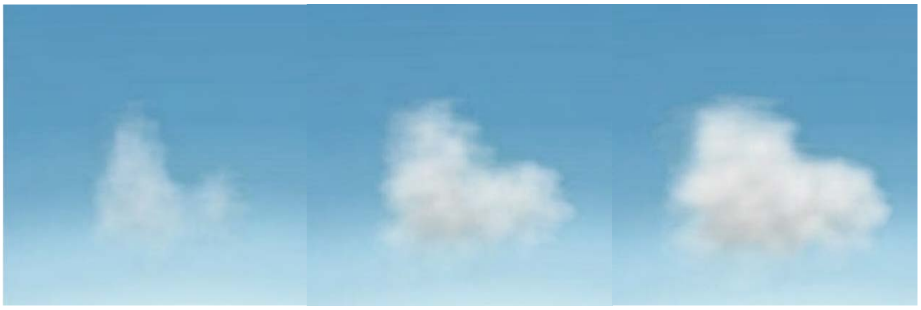


Fig. 3. Cloud formation with a grid of size $64 \times 64 \times 10$

Another external force we apply on the horizontal direction is the vorticity confinement. Fluids such as smoke typically contain rotational flows at a variety of scales. As explained by Fedkiw *et al.* [20], numerical dissipation in simulation on a coarse grid damps out these interesting whirling features. Vorticity confinement is used to restore these motions.

5 Decoupling Simulation

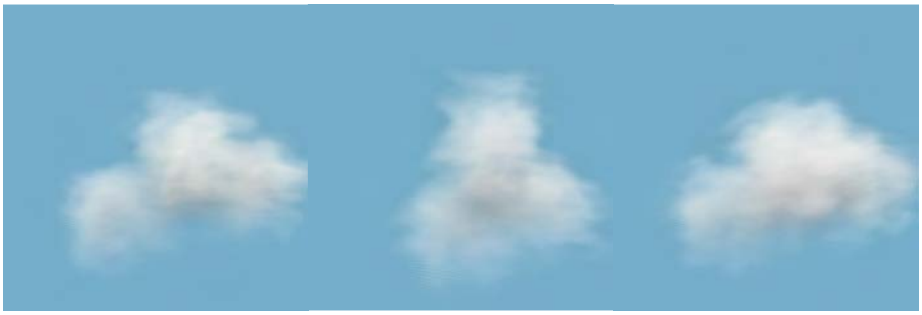
Following the brief description in section 3, here is the decoupling simulation in detail. The cumulus cloud is often formed by more than one plumes, thus we develop several turbulent plumes together. Due to the assumption of similarity in radial profile at all heights, we select several key frames for the horizontal development. Over the horizontal domain, the 2D space is divided regularly into uniform girds, and we simulate the horizontal dynamics by solving the 2D Navier-Stoke's equation driven by the virtual temperature difference. Here the virtual temperature difference results from the variation in the density of water vapor. The vertical dimension characteristics is explicitly captured by the constant parameters in the similarity solution, thus no PDE solving is needed in the 1D vertical dimension.

In initialization, the user is free to choose the values of a set of fluid parameters. This parameter set includes the atmosphere's stratification N, the heat flux F_0 and the mass entrainment constant α. For a bigger value of F_0, the plume has more momentum and is therefore potential to reach a higher elevation leading to a cumulus cloud with a larger vertical extent. For a larger value of α, more entrainment mass is involved and there is more mixing with the environmental atmosphere, which leads to a lower likelihood for the plume to reach a high vertical extent. The parameter N is a measurement of atmosphere stratification. If N is larger, the potential temperature difference is bigger for the same height step and it involves more heat exchange, more mixing and thus a lower vertical extent. The other two user-definable parameters in the initialization stage are the kernel width a and the slope drop-off n. When n is bigger (smaller), the plume kernel is closer to the 'top-hat' ('Gaussian') profile. If the kernel width a is larger, there are less intersection areas between the kernels of adjacent plumes. Through this initialization, the user can control the general shape of a cloud with great flexibility.

– **Boundary Condition.** Decoupling simulation has no vertical boundary limits since plume has vertical extents. In the horizontal direction, we let the velocity vanish at the boundary.
– **Interpolation Between Key Cloud Layers.** Under the assumption of geometrical similarity at all heights, we do not have to perform the horizontal computation at each height. Instead, we choose several cloud layers along the vertical axis, which we call 'key cloud layers', for the horizontal simulations. The density values of the remaining cloud layers are obtained through interpolation with noise perturbation.

6 Experimental Results

The process of a cumulus cloud's formation is illustrated in Fig. 3 where a cumulus cloud gradually develops from that in Fig. 3(a) to that in Fig. 3(c). For this example, a grid size of 64×64 and 10 key cloud layers are used for the simulation of horizontal dynamics. Due to the computation reduction in the vertical dimension and the reduced number of cloud layers for horizontal simulation, efficient computing has been achieved while good visual quality is still observed as shown in Fig. 3∼5. The example in Fig. 3(c) is generated on an Intel Pentium 4 3.0GHz machine. According to our experiments, for a grid of size $128 \times 128 \times 32$, we need an average of 3.01 seconds for each iteration in the simulation. For a grid of size $64 \times 64 \times 64$, our cloud simulator costs 1.13 seconds for each iteration in the simulation.



(a)F={2.3, 5.5, 2.5}; N=0.01, α=0.3; n=2,a=1

(b)F={1.6, 9.8, 2.0}; N=0.01, α=0.3; n=2,a=1

(c)F={2.3, 5.5, 2.5}; N=0.01, α=0.3; n=5,a=50

Fig. 4. Examples of cumulus cloud with different parameter sets

To illustrate the effect of constant parameters on the cloud shape, we show in Fig. 4 three cumulus clouds synthesized with different sets of parameters. In Fig. 4, each cumulus cloud is composed of three plumes. For the example in Fig. 4(a), the values of heat flux for the three plumes are 2.3, 5.5 and 2.5, respectively; the buoyancy frequency is 0.01; the entrainment coefficient is 0.3; the width and the slope drop-off of the plateau kernel is 1 and 2, respectively. For the example in Fig. 4(b), we keep the plateau kernel parameters the same as those in Fig. 4(a), and only change the values of heat flux for the three plumes to 1.6, 9.8 and 2.0. Compared with Fig. 4(a), a bigger vertical extent of cloud is observed in Fig. 4(b) due to the bigger heat flux value. In Fig. 4(c), all the atmospheric parameters F, N , α are kept the same as those in Fig. 4(a), while the width and the slope drop-off of the plateau kernel are 50 and 5, respectively. We observe that the cloud in Fig. 4(c) is more fluffy than that in Fig. 4(a) since the width of the plateau kernel is closer to 'top-hat' for the cloud in Fig. 4(c). It is demonstrated in Fig. 4 that, with the proposed cloud simulator, the general shape of a cumulus cloud can be flexibly controlled through the adjustment of constant parameters.

In Fig. 5(a), a sky scene with multiple clouds is presented. In this figure, the clouds are generated using four different sets of constant parameter values. In Fig. 5(b), the same scene under a different illumination is rendered. The visual results as shown in Fig. 5 demonstrate the potential of our algorithm for realistic cloud modeling and illumination. The cloud images in Fig. 5 are rendered with Behrens's method [15]. The rendering time is less than five seconds for each scene.

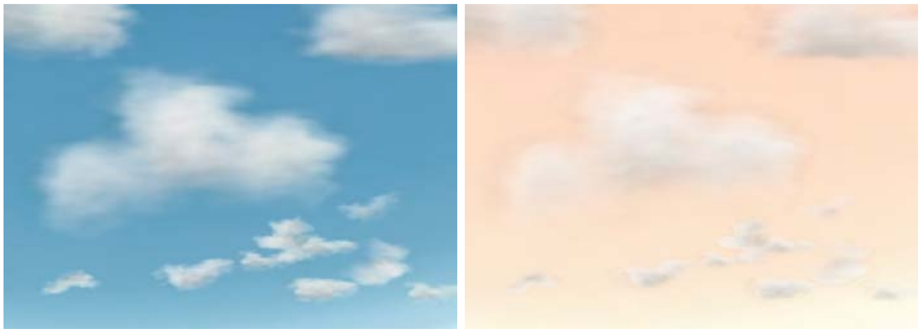

(a) (b)

Fig. 5. Examples of multiple cumulus cloud scene

7 Conclusion and Future Work

In this paper, a new cumulus cloud simulation method based on the similarity approach of turbulent plume is proposed. With the similarity approach, we use a set of constant parameters to describe the general characteristics of the cumulus clouds, and decouple the 3D simulation into the 2D horizontal and the 1D vertical simulation. This new algorithm reduces the computational cost of the 1D vertical simulation to almost zero. In addition, it provides flexible general shape control. In the future, we plan to work on flexible wind effect simulation, apply parallel computing to the 2D simulations at different heights, and utilize the fast 2D fluid solver on GPUs to further reduce the computational cost.

References

1. Ebert, D.S.: Volumetric modeling with implicit functions: A cloud is born. In: SIGGRAPH 1997 (1997)
2. Schopik, J., Simons, J., Ebert, D.S., Hansen, C.: A Real-Time cloud modeling, rendering, and animations system. In: EUROGRAPHICS (2003)
3. Voss, R.: Rourier synthesis of gaussian fractals: 1/fnoises, landscapes, and flakes. In: Proc. SIGGRAPH 1983, vol. 10 (1983)
4. Gardner, G.Y.: Visual simulation of clouds. In: SIGGRAPH (1985)
5. Nishita, T., Dbashi, Y., Nakamae, E.: Display of clous taking into account multiple anisotropic scattering and sky light. In: ACM SIGGRAPH, ACM Press, New York (1996)
6. Neyret, F.: Qualitative simulation of convective cloud formation and evolution. In: Eurographics Workshop on Animation and Simulation, pp. 113–124 (1997)
7. Wang, N.: Realistic and fast cloud rendering. In: ACM SIGGRAPH, ACM Press, New York (2003)
8. Stam, J.: Stable fluids. In: ACM SIGGRAPH, Aug 1999, ACM Press, New York (1999)
9. Kajiya, J.T., Herzen, B.P.V.: Ray tracing volume densities. In: SIGGRAPH 1984, pp. 165–174 (1984)
10. Dbashi, Y., Kaneda, K., Okita, T., Nishita, T.: A simple, efficient method for realistic animation of clouds. In: ACM SIGGRAPH, ACM Press, New York (2000)
11. Miyazaki, R., Yoshida, S., Dobashi, Y., Nishita, T.: A method for modeling clouds based on atmospheric fluid dynamics. In: IEEE Computer Graphics & Applications, IEEE Computer Society Press, Los Alamitos (2001)
12. Harris, M.J., Baxter III, W.V., Scheuermann, T., Lastra, A.: Simulation of cloud dynamics on graphics hardware. In: EUROGRAPHICS (2003)
13. Emanuel, K.A.: Atmospheric Convection, 1st edn. Oxford University Press, Oxford (1994)
14. Jacobson, M.Z.: Fundamentals of Atmospheric Modeling. Cambridge University Press, Cambridge (1998)
15. Behrens, U., Ratering, R.: Adding shadows to a texture-based volume renderer. In: IEEE Symposium on Volume Visualization, pp. 39–46. IEEE Computer Society Press, Los Alamitos (1998)
16. Morton, B.: Buoyant plumes in a moist atmosphere. J. Fluid Mech. 2 (1957)
17. Turner, J.S.: The starting plume in neutral surroundings. J. Fluid Mech. 13 (1962)
18. Pajarola, R., Sainz, M., Gudiotti, P.: Object-space blending and splatting of points. Technical report, UCI-ICS Technical Report (2003)
19. Andrews, D.G.: An Introduction to Atomospheric Physics, 1st edn. Cambridge University, Cambridge (2000)
20. Fedkiw, R., Stam, J., Jensen, H.W.: Visual simulation of smoke. In: ACM SIGGRAPH, pp. 15–22. ACM Press, New York (2001)

Six Degrees of Freedom Incremental Occlusion Horizon Culling Method for Urban Environments

Gurkan Koldas[1], Veysi Isler[1], and Rynson W.H. Lau[2]

[1] Department of Computer Engineering
Middle East Technical University
Turkey
[2] Department of Computer Science
University of Durham
United Kingdom

Abstract. Early culling of the invisible geometric primitives in a complex scene is valuable for efficiency in the conventional rendering pipeline. This may reduce the number of geometric primitives that will be processed in the rest of the pipeline. In this paper, we propose a conservative six degrees of freedom (DoF) incremental occlusion culling method called Delta-Horizon (ΔH). ΔH method is based on constructing an occlusion horizon (OH), which is a set of connected lines passing just above all visible primitives, for culling the invisible primitives beyond. Utilizing the coherence of occluders enables the incremental update of OH in consecutive frames. Although ΔH method may work in image space, we utilize polar coordinates and build OH in object space. This not only facilitates a quick update of OH, but also overcomes the drawbacks of previous OH methods such as viewing in six DoF, occlusion culling within up-to-360-degree viewing frustums in one pass and camera zooming without additional cost.

1 Introduction

Real-time visibility is an important issue in virtual environment applications. Visibility studies focus on culling the invisible primitives to reduce the rendering cost. The basic visibility culling method discards the primitives outside the current view frustum. However, it does not purge obscured primitives inside the view frustum.

The aim of traditional visibility culling methods is to obtain exact visible set. However, this is too expensive and infeasible for real-time applications. Hence, conservative visibility methods, which overestimate the visible set of primitives, are proposed to compromise between culling accuracy and culling speed [1,2]. This overestimated visible set of primitives is referred to as *Potential Visible Set (PVS).* Apart from culling accuracy, there have been visibility studies on the location of the viewer (point-based / region-based / cell-based), solution space (2D / 2½D / 3D, discrete / continuous, image precision / object precision) and occluder fusion support [1,3 ,4].

The motivation of this study is to develop a walkthrough application in complex urban environment where buildings are the main source of occlusion [5]. The viewer only sees a small part of the city because close buildings usually occlude the ones behind. The methods which are developed for 3D visibility can also be used for urban walkthroughs. However, the computation of full 3D occlusion includes a significant overhead since buildings are connected to the ground. 2½D calculations should be enough for urban environments.

In this paper, we propose *Delta-Horizon (ΔH)* method for real-time visualization of complex urban environments. It is based on OH method proposed by Downs et al. [6]. The proposed ΔH method, as in [6], computes the occlusion in 2½D, works on visibility based occluder selection, and supports both conservative visibility and occluder fusion. Besides, it builds OH using polar coordinates in object space which give rise to several advantages. Firstly, ΔH method allows flexible six DoF camera movements. Downs' method has only four DoF. Secondly, it accommodates a 360-degree-wide field of view in one pass. This is desired for applications that need wide angle visibility such as radar simulations and panoramic viewing [7]. Thirdly, PVS computed in object space may be utilized without recomputation when the camera zooming is required. Finally, a significant improvement is achieved in updating OH incrementally.

The rest of the paper is organized as follows. Section 2 reviews related work on visibility methods based on occlusion culling. Section 3 initially explains the details of the former OH method [6] and then presents the proposed ΔH method in detail. Section 4 discusses the performance gain in the empirical results. Finally, Section 5 presents a conclusion and possible future work.

2 Related Work

Comprehensive surveys on visibility culling may be found in [1,14]. However, the proposed ΔH method is an eye-point based method which is computed from a point for each frame. The point-based visibility culling studies which can be roughly classified as geometric, image-based and hardware-based are summarized as follows.

2.1 Geometric Point Visibility

Geometric point visibility methods resolve the relations of primitives in object space to compute occlusion.

The method proposed by Coorg and Teller [8,9] makes use of a set of large convex occluders to compute the occlusion in the scene. The method compares two objects, and takes one of them as an occluder and the other as the occludee. The endpoint connecting lines of occluder and occludee are defined as supporting and separating lines which partition the space into regions. In 3D, vertex-edge pairs generate supporting and separating planes instead of lines.

Hudson et al. [10] proposed a method where a set of occluders is chosen dynamically. It computes occluders for each cell in preprocess. When the viewpoint is inside the cell, it computes shadow frustums of the selected occluders to cull the bounding boxes of objects' hierarchy [1].

Bittner et al. [11] improve the Hudson et al.'s method. They combine the shadow frustums of the occluders into an occlusion tree as in [12]. The comparison is reduced to a tree with an O(logN) depth, while taking occluder fusion into consideration [1].

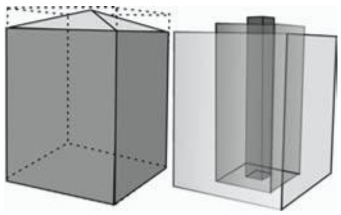


Fig. 1. Left: Building with bounding box. Right: CVPs of a building.

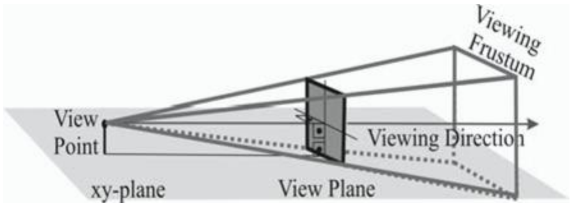


Fig. 2. Viewing direction is parallel to the xy- plane (i.e., ground plane)

2.2 Image-Based Point Visibility

Image-based methods execute the culling operation on the discrete representation of the image. They fill the image during the rendering of the scene. The subsequent objects are culled away quickly by the already filled parts of the image [1,13].

Greene proposes Hierarchical Z-Buffer (HZB) method, which is based on an octree hierarchy in object space and a z-buffer in image-precision [14]. The z-buffer is defined as a multi-level buffer where the finest layer is the contents of the z-buffer. Each element in all layers holds the furthest z-value. Objects have been tested from coarser to more accurate level.

The Hierarchical Occlusion Map (HOM) method proposed by Zhang [15] is similar to the HZB. HOM keeps opacity information with the z-values of the occluders. The method tests objects for overlap with occluded regions and then compares their depth values. It supports approximate visibility culling. Hence, visible objects through a few pixels can be culled using a user-specified opacity threshold.

Wonka et al. [5,16] propose a conservative image-based occlusion culling method for urban environments. They use two auxiliary data structures in a regular 2D grid: the scene grid and occluder grid. In each frame, the method selects a set of occluders from the precomputed occluder grid and renders the occluder shadows to an auxiliary buffer called the cull map. Each pixel in the cull map corresponds to a cell of the scene grid (object space). Visibility of a cell is calculated according to the corresponding pixel value in the cull map. The method supports occluder fusion. The computation time depends on the numbers of pixels and occluders determined in preprocessing. The overhead incurred by copying the frame buffer and visibility traversal increases proportional to the size of the cull map.

Unlike the above methods, OH methods in [1,17] work in object space and use image plane to find the PVS. We discuss these methods in the following section.

2.3 Hardware-Based Point Visibility

Some of the methods use Graphical Processing Unit (GPU) to estimate the occlusion in image space. The drawbacks are sending all the primitives to GPU and reading data from the frame buffer. The latter process is very slow. Hardware vendors have started adapting occlusion culling features into their designs. A feedback loop or flag added to the hardware reports any change in the z-buffer when a fragment passes the depth test [1,18 ,19]. When computing the PVS, the depth test can be optimized to check the bounding boxes or an enclosing k-dop [20], which closely encloses the objects. Hardware vendors also employ occlusion culling methods [21,22] and provide occlusion culling support for common graphics APIs [13,23,24].

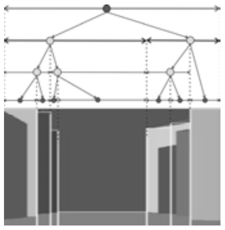


Fig. 3. Binary tree representation of OH

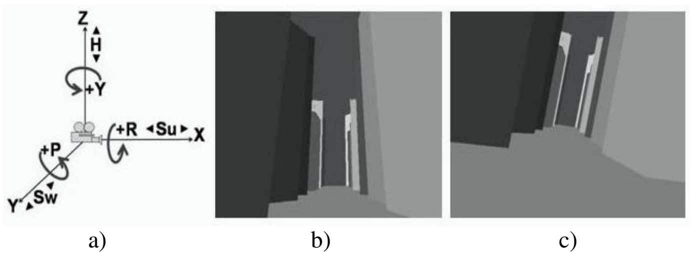


Fig. 4. a) Six DoF: Surge (Su), sway (Sw), heave (H), roll (R), pitch (P) and yaw (Y). b-c) If camera pitches (P) or rolls (R), OH in the view plane can not be built conservatively in former methods [6,17]

3 Delta Horizon (ΔH) Method

After discussing the former OH methods in [6,17], we explain the proposed method in detail.

3.1 Details of Former OH Methods

Downs et al. use the height fields to determine occlusion in 2½D for urban environments [6]. Each building is modeled with a bounding box and a set of Convex Vertical Prisms (CVPs) as seen in Fig. 1. A bounding box is used for outer hull and CVPs are applied for inner hull to occupy the inside of the building. Additionally, the scene is organized in a 2D quad tree scene hierarchy on the xy-plane. Each quad tree cell keeps the maximum height of any object within the cell or any of its descendants.

A sweeping plane builds OH in a front-to-back traversal. Cells are tested against OH during the sweep. If the entire cell lies below the horizon, it is culled away. If it is not, buildings in the cell are tested. If the building is visible at its minimum distance to view point, it is added to PVS and its CVPs are applied to update OH at its maximum distance. This process ends after all the cells in view frustum are tested.

OH is a series of 2D lines in the view plane which is perpendicular to xy-plane as seen in Fig. 2. It is a conservative mask of the space occluded by all buildings encountered in a front-to-back traversal. OH is approximated as a piecewise constant function and represented in a binary tree as seen in Fig. 3. Each node in the tree has an x range and a y-value for the mask height. The minimum and maximum mask heights of its descendants are stored at every internal node. Testing of a cell or a building is terminated without recursing to the leaves of the tree if its mask is completely below the minimum value or above the maximum value of a portion of the OH.

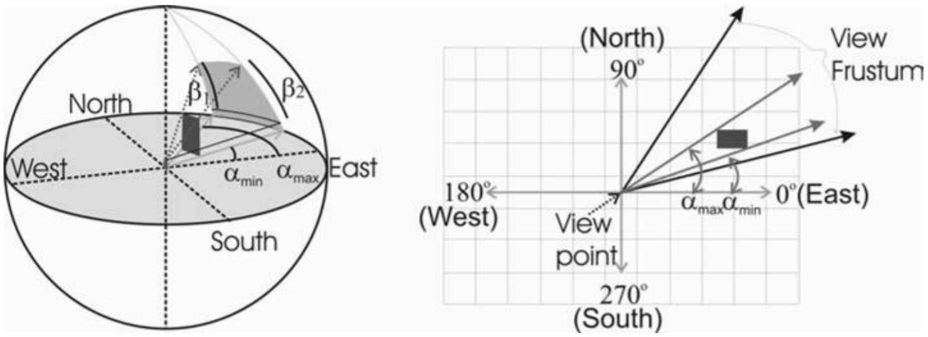


Fig. 5. a) Occluder shadow in polar coordinates. Horizontal angles (α_{min}, α_{max}) bound a CVP in both sides and measured from the positive x-axis (East). Vertical angle (β-values) bounds the height of a CVP and measured from the viewing direction parallel to the ground plane. b) Top view: OH is built only for the horizontal angle values where view frustum occupies.

Lloyd and Egbert [17] adapted OH method to hierarchical terrains. This method also uses a quad tree structure and individual line segments to approximate the edges of the leaf cells instead of CVP to compute OH as in [16].

Overhanging structures and bridges do not contribute to occlusion in the scene, but they will be correctly culled or accepted potentially visible. The drawbacks of these OH methods are summarized as follows:

1) OH can not be built in the view plane when the camera pitches or rolls as seen in Fig. 4b and 4c. Thus, these methods have only four DoF.
2) These methods have to build OH and compute PVS for each frame from scratch when the viewer moves.

3.2 Proposed Object Space OH Method

The motivation of implementing OH in object space is to overcome the drawback of four DoF. This drawback prevents the implementation of OH method in walkthrough application even if it is easy to implement. The main difficulty is how to represent OH in object space. OH is built as a series of 3D lines in polar coordinates. Like the former methods, it is the accumulation of the shadow frustums of CVPs. The basic difference is the computation of shadow frustum. We summarize how polar coordinates are utilized in the computation of shadow frustums as follows:

1) The view point is located at the center of polar coordinate system.
2) In polar coordinates, positive x-axis points to east, positive y-axis points to north, and positive z-axis points up, as seen in Fig. 5a.
3) The shadow frustum of a CVP is computed using its visible cross-section and extends to infinity as seen in Fig. 5a. Shadow frustum is estimated by two horizontal angles and one vertical angle with respect to view point.
4) Horizontal angles (α_{min}, α_{max}) bound a CVP in both sides. They are measured from east and range between 0 and 360 degrees. To find α_{min} and α_{max} of a CVP, we project CVP to the xy-plane orthogonally as seen in Fig. 5b. If an occluder shadow

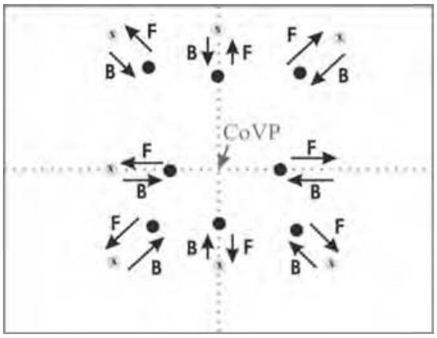


Fig. 6. Coherence of vertex projection in consecutive frame

Table 1. Impacts of occluders

User Action	x: part of Current OH	Consecutive OH ... in Vertical Axis	Consecutive OH ... in Horizontal Axis
Forward Movement	$\exists$ x above CoVP	moves up	moves away from CoVP
	$\exists$ x below CoVP	moves down	moves away from CoVP
Backward Movement	$\exists$ x above CoVP	moves down	moves to CoVP
	$\exists$ x below CoVP	moves up	moves to CoVP
Lateral Movement	$\forall$x	do not move	move to opposite site
Change in Viewing Direction	$\forall$x	do not shift	shift to opposite site

spans zero horizontal degree (i.e., extends from 350 degree to 20 degree in polar coordinates), we split it into two parts and test both parts individually.

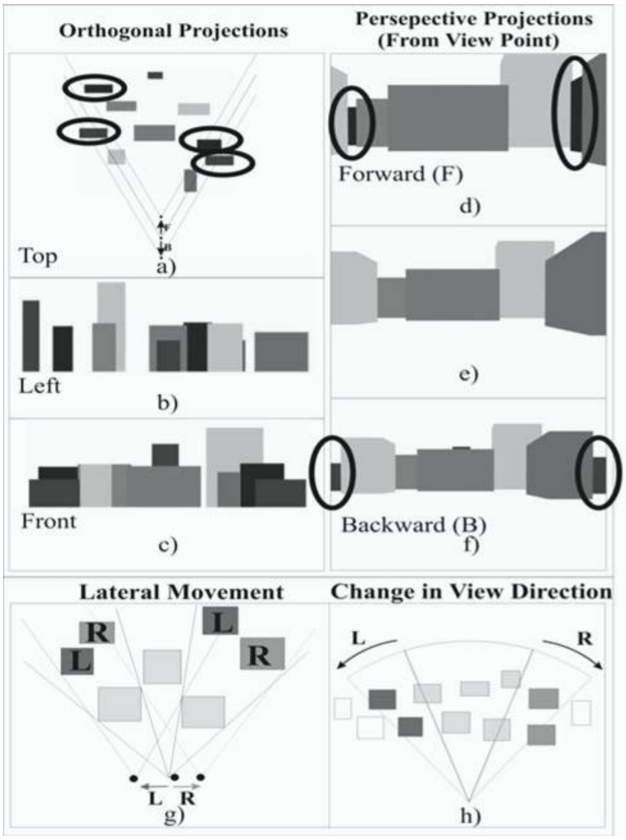
5) Vertical angles (β-values) bound the height of a CVP. They are measured from the xy-plane and range between −90 and +90 degrees. The vertices at α_{min} and α_{max} bound the occluder shadow frustum on both sides. The smallest vertical angle of these two vertices is chosen as β_{min} for conservative visibility.

The contributions of the proposed object space method are summarized as follows:

1) It supports six DoF. It does not fail when the camera pitches or rolls since OH is built before viewing transformation.
2) Polar coordinates enable building OH up-to-360-degrees in one pass.
3) Computing PVS before viewing transformation enables zooming without re-computation.

3.3 Incremental OH Method: ΔH Method

Former methods test all the cells in view frustum and builds OH by the cells in front of OH. However, the invisible cells beyond OH are tested. In the proposed ΔH method, we aim to reduce the cost of testing invisible cells beyond OH. Its motivation is derived from the fact that OH does not change significantly in a consecutive frame during a walkthrough. The proposed ΔH method is implemented in both image and object spaces. From here, we initially explain the coherence of occluders and describe the proposed method in image space.



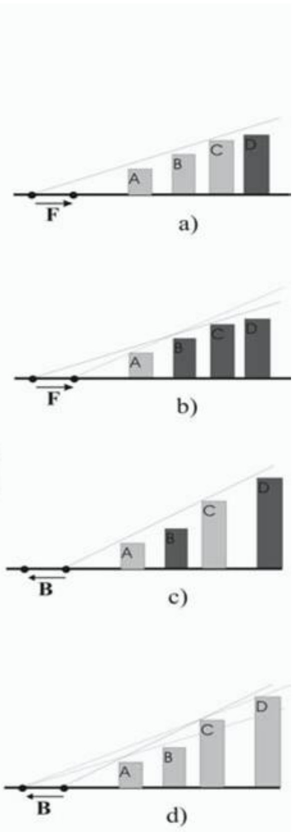


Fig. 7. Coherence of occluder. Orthogonal projections from a) top, b) left and c) front. d-f) Perspective projections from view point. d) In forward (F) movement. e) Before movement. f) In backward (B) movement. g) In lateral movements. h) In view ing direction changes.

Fig. 8. a) Before forward (F) movement: A, B, C are visible and C builds OH. D is occluded. b) After forward (F) movement: A is visible and builds OH. B and C become occluded. c) Before backward movement: A and C are visible and C builds OH. B and D are occluded. d) After backward movement: A, B, C and D are visible and D builds OH.

In image space method, viewing direction is parallel to xy-plane. The primitives in viewing direction project to the Center of View Plane (CoVP) as seen in Fig. 6. Fig. 6 shows the coherence of vertex projections in the view plane when the viewer moves forward (F) and backward (B). It is obvious that the consecutive projections move away from CoVP in forward movement and move to CoVP in backward movement.

Fig. 7 summarizes the coherence of occluders that are taller than the view point's height. They project upper half of the view plane. Visible Buildings in forward and backward movements are marked as seen in Fig 7d and Fig 7f with respect to Fig 7e. Fig 7g shows the visible buildings (according to left (L) and right (R) movement) in lateral movements. If the viewing direction only changes, the buildings entered the view frustum are dark in Fig 7h.

We consider how occluders in current OH impact the consecutive OH. The impact of an occluder varies according to its distance in object space and projection position in the view plane. Table 1 summarizes the impact of occluders for the user actions below:

Forward movement: The parts of current OH above CoVP move up and its parts below CoVP move down in vertical axis. Simultaneously, they move away from CoVP in horizontal axis as seen in Fig. 7d. Buildings in front of OH may form OH in consecutive frame regardless of being visible or not, as seen in Fig 8a and Fig 8b.

Backward movement: The parts of current OH above CoVP move down and its parts below CoVP move up in vertical axis. Simultaneously, they move to CoVP in horizontal axis as seen in Fig 7f. An occluded building in front of current OH may be visible as seen in Fig 8c and Fig 8d. Occluded buildings behind OH may form OH in consecutive frame as seen in Fig. 7f.

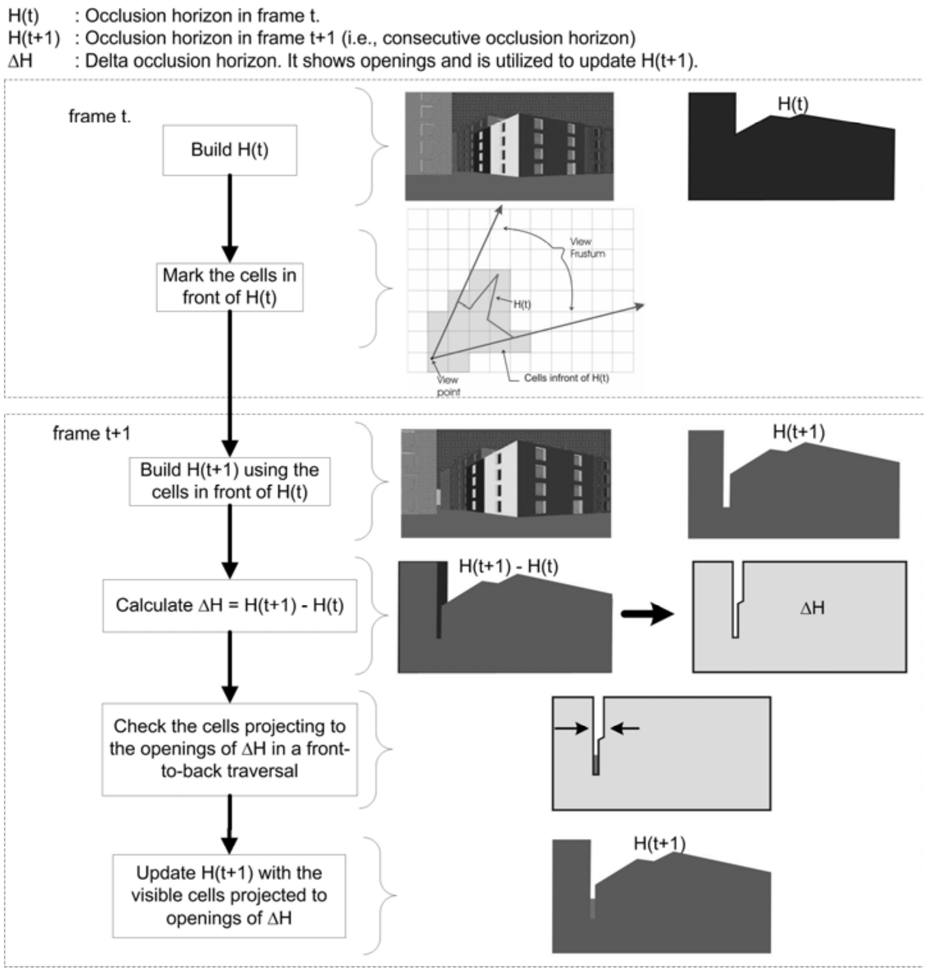

H(t) : Occlusion horizon in frame t.
H(t+1) : Occlusion horizon in frame t+1 (i.e., consecutive occlusion horizon)
ΔH : Delta occlusion horizon. It shows openings and is utilized to update H(t+1).
frame t.
Build H(t)
H(t)
Mark the cells in front of H(t)
View Frustum
H(t)
View point
Cells infront of H(t)
frame t+1
Build H(t+1) using the cells in front of H(t)
H(t+1)
Calculate ΔH = H(t+1) - H(t)
H(t+1) - H(t)
ΔH
Check the cells projecting to the openings of ΔH in a front-to-back traversal
Update H(t+1) with the visible cells projected to openings of ΔH
H(t+1)

Fig. 9. Steps of proposed ΔH method

Table 2. Performance comparisons

	OH(IS)	OH(OS)	ΔH
Frame Time (msec.)	243.9	113	27.25
Culling Time (msec.)	227.2	96.9	10,55

a)

	OH(OS)/OH(IS)	ΔH/OH(IS)	ΔH/OH(OS)
Frame Time	2.16	8.95	4.15
Culling Time	2.34	21.54	9.18

b)

Lateral movement: All parts of current OH move to opposite side with respect to the movement. The more an occluder is away from the view point in object space, the less its consecutive projection moves in the view plane. Buildings in front of current OH may form OH in consecutive frame.

Changing only viewing direction: If viewing direction changes, new primitives enter the view frustum as seen in Fig.7h and OH shifts to opposite side in consecutive frame.

We summarize the abbreviations and the steps of ΔH method depicted in Fig. 9 as follows:

H*(t)*	**:**	OH in frame *t*.
H*(t+1)*	**:**	OH in frame *t+1* (i.e., consecutive OH)
ΔH	**:**	Delta OH. It shows openings and is utilized to update **H*(t+1)***.

1) Mark the cells in front of **H*(t)*** at frame *t*.

2) Build **H(*t+1*)** by the cells in front of **H*(t)*** from close to far. If an occluded primitive in frame *t* is visible in frame *t+1*, it should be added to the PVS.

3) Calculate Δ***H*** = **H*(t+1)* – H*(t)***. ΔH has some parts risen and some parts lowered (i.e., openings). Subtraction details are as follows: There is no need to check the cells behind the parts of risen ΔH because they have already occluded in frame *t*. Thus the values of these regions in ΔH are extended to maximum values of horizon. Lowered parts of horizon in consecutive frame denote the openings to be checked in object space. We only copy the values of **H*(t+1) to* ΔH** for these regions.

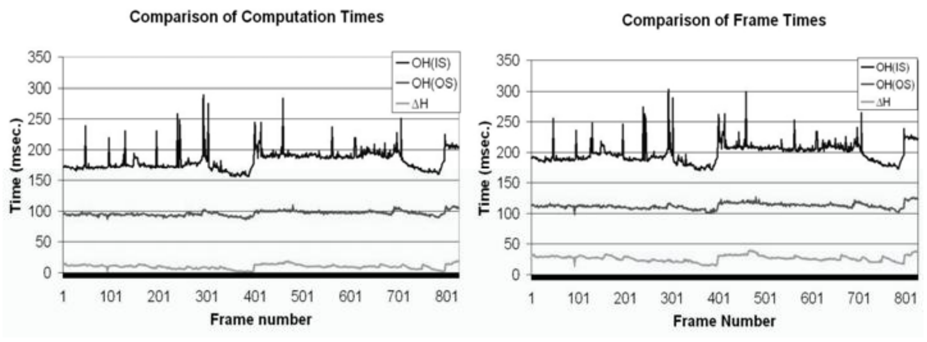
4) Check the cells projecting to the openings of **ΔH** in a front-to-back traversal.

5) If the viewer only changes the viewing direction, the cells entered the view frustum should be tested as seen in Fig. 7h.

After testing all the cells in the openings of **ΔH, H*(t+1)*** is updated incrementally**.** We utilize link list and binary tree data structures together in the implementation. Binary tree data structure enables search the tree in (log n) time. Linked list data structure enables to compute ΔH easily. Thus, we significantly decrease the cost of OH method by reducing the number of cells tested behind OH.

4 Results and Discussions

As a testbed, we modeled an urban environment consisting of 160,000 buildings, each of which has 46 triangles. The testbed has 7.36M triangles totally and is organized into 2D quad tree on the xy-plane. The buildings are scattered into the quad tree at the lowest level cell. In the experiments, OH is computed only for view frustum as seen in Fig. 5b.



a) b)

Fig. 10. Statistics gathered in the testbed a) Comparison of frame times. b) Comparison of OH computation times.

We perform our tests on PC Pentium IV 3 GHz. with an nVidia GeForce FX5700LE-256 MB graphics card. We implement the methods in MS Visual C++ 6.0 with OpenGL API. The statistics are gathered on a predetermined walkthrough. In Fig. 10, we compare the timings of the following culling methods: OH culling method in image space (*OH(IS)*), proposed OH method in object space (*OH(OS)*) and proposed ΔH method (Δ*H*) in object space. Fig. 10a summarizes the frame times which are the total of culling time (i.e., time of building OH and computing PVS) and rendering time (i.e., time of sending PVS to GPU and rendering). We only summarize the culling times to compare the cost of methods as seen in Fig. 10b.

Table 2 compares the averages of times gathered in Fig.10 and summarizes the speeds up of the proposed methods. The results show that ΔH achieves 8.95 times speedup in frame time and 21.54 times speedup in culling time over OH(IS). Besides, OH(OS) achieves 2.16 times speedup in frame time and 2.34 times speedup in culling time over OH(IS). The latter speed ups are caused by the difference between estimating the projected coordinates in image plane and angle values in polar coordinates. We utilized gluProject() function to estimate the window coordinates in OH(IS). If there is a cheaper way to estimate the window coordinates in OH(IS), these speedups may reduce. Therefore, we need to compare the times of OH(OS) and ΔH to see the performance gain in ΔH method. It is important to note that ΔH achieves 4.15 times speedup in frame time and 9.18 times speedup in culling time over OH(OS). As a result, ΔH method reduces the costs of frame and culling times significantly.

The performance gain of the proposed ΔH method is caused by reducing the number of testing cells behind OH. In addition, the number of nodes in ΔH tree *(after step 3 in Section 3.3.)* is reduced to 20% at the average. Thus it speeds up testing primitives and updating OH.

There is a trivial constrain for the proposed ΔH method. If user moves more than the width of a building, we can not update the occlusion horizon with a building which has smaller width than the distance moved. This extreme case is depicted in Fig. 11. The building A is visible in the consecutive frame. Unfortunately, ΔH method misses it because the horizon is risen as seen in Fig. 11. This extreme case can be solved by three different approaches: 1: Limiting the movement of a user with respect to the smallest width of the buildings in the environment 2: Updating OH with the buildings which have wider width than the distance of movement. 3: Switching the incremental method to non-incremental method for an instance.

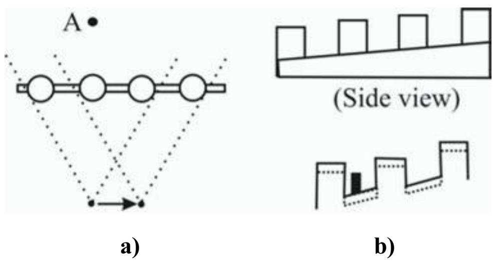


Fig. 11. a) Top view. Building A becomes unoccluded in lateral motion. b) Side view: Occlusion horizon rises and no cells beyond it are checked, missing the green building.

5 Conclusion and Future Work

In this paper, we propose six DoF incremental occlusion horizon culling method (ΔH method) based on Ref. [6]. The proposed ΔH method, like former method, computes the occlusion in 2½D, works on visibility based occluder selection, and supports both conservative visibility and occluder fusion. Besides, it builds OH using polar coordinates in object space which give rise to several advantages. Firstly, the proposed method allows six DoF. Former method fails in viewing cases such as pitching and rolling of the camera. Secondly, the use of polar coordinates makes it easy to perform culling within the view frustum up-to-360-degree in one pass. This is desired for applications that need wide angle visibility such as radar simulations and panoramic viewing. Thirdly, computing OH and PVS in object space enables zooming without extra cost of recomputation. Finally, a significant improvement is achieved in updating OH incrementally.

As a future work, we initially plan to extend it to region-based method. Afterwards, we plan to develop an on-demand loading conservative visibility method for distributed simulation systems.

References

1. Cohen-Or, D., Chrysanthou, Y., Silva, C., Durand, F.: A Survey of Visibility for Walkthrough Applications. IEEE Trans. on Visualization and Computer Graphics 9(3), 412–431 (2003)
2. Airey, J.: Increasing Update Rates in the Building Walkthrough System with Automatic Modelspace Subdivision and Potentially Visible Set Calculations. PhD thesis, University of North Carolina, Chapel Hill (1991)
3. Bittner, J.: Hierarchical Techniques for Visibility Computations. PhD Dissertation, Czech Technical University, Prague, Dec (2002)

4. Bittner, J., Wonka, P.: Visibility in Computer Graphics. Journal of Environment and Planning B: Planning and Design 30(5), 725–729 (2003)
5. Wonka, P.: Occlusion Culling for Real-time Rendering of Urban Environments. PhD Thesis, Technische Universität Wien (2001)
6. Downs, L., Moeller, T., Sequin, C.: Occlusion Horizons for Driving Through Urban Scenery. In: Proc. ACM Symp. on Interactive 3D Graphics, pp. 121–124. ACM Press, New York (2001)
7. Koldas, G., Cevikbas, S.B., Isler, V.: Echo Generation for Mobile Radar Simulation. In: Proc. 2nd IEEE Int'l. Conf. on Tech. for Homeland Security and Safety, pp. 79–87. IEEE Computer Society Press, Los Alamitos (2006)
8. Coorg, S., Teller, S.: Temporally Coherent Conservative Visibility. In: Proc. ACM Symp. Computational Geometry, pp. 78–87. ACM Press, New York (1996)
9. Coorg, S., Teller, S.: Real-time Occlusion Culling for Models with Large Occluders. In: Proc. ACM Symp. on Interactive 3D Graphics, pp. 83–90. ACM Press, New York (1997)
10. Hudson, T., Manocha, D., Cohen, J., Lin, M., Hoff, K., Zhang, H.: Accelerated Occlusion Culling Using Shadow Frusta. In: Proc. ACM Symp. Computational Geometry, pp. 1–10. ACM Press, New York (1997)
11. Bittner, J., Havran, V., Slavik, P.: Hierarchical Visibility Culling with Occlusion Trees. In: Proc. Computer Graphics International 1998, pp. 207–219 (1998)
12. Chin, N., Feiner, S.: Near Realtime Shadow Generation Using BSP Trees. ACM Computer Graphics 23(3), 99–106 (1989)
13. Gummerus, S.: Conservative from Point Visibility. Master's Thesis, University of Tampere (2003)
14. Greene, N., Kass, M., Miller, G.: Hierarchical Z-buffer Visibility. In: Proc. ACM SIGGRAPH, pp. 231–240. ACM Press, New York (1993)
15. Zhang, H.: Effective Occlusion Culling for the Interactive Display of Arbitrary Models. Ph.D. Thesis, University of North Carolina, Chapel Hill (July 1998)
16. Wonka, P., Schmalstieg, D.: Occluder Shadow for Fast Walkthroughs of Urban Environments. Computer Graphics Forum 18(3), 51–60 (1999)
17. Lloyd, B., Egbert, P.: Horizon Occlusion Culling for Real-time Rendering of Hierarchical Terrains. In: Proc. Conf. on Visualization, pp. 403–410 (2002)
18. Scott, N., Olsen, D., Gannet, E.: An Overview of the Visualize Fx Graphics Accelerator Hardware. The HewlettPackard Journal, 28–34 (May 1998)
19. Severson, K.: Visualize Workstation Graphics for Windows NT. HP Product Literature (1999)
20. Klosowski, J., Held, M., Mitchell, J., Sowizral, H., Zikan, K.: Efficient Collision Detection Using Bounding Volume Hierarchies of Kdops. IEEE Trans. on Visualization and Computer Graphics 4(1), 21–36 (1998)
21. Morein, S.: ATI Radeon HyperZ Technology. In: Proc. SIGGRAPH/Eurographics Graphics Hardware Workshop, Hot3D Session (2000)
22. Nvidia Lightspeed Memory Architecture II. Technical brief, NVIDIA (January 2002)
23. DirectX 9 SDK. Microsoft Corporation (2003), http://msdn.microsoft.com/directx
24. Segal, M., Akaley, K.: The OpenGL Graphics System. A specification (Version 2.0), Silicon Graphics Inc., (October 2004)

A Framework for Exploring High-Dimensional Geometry

Sidharth Thakur and Andrew J. Hanson

Computer Science Department, Indiana University Bloomington
{sithakur,hanson}@cs.indiana.edu

Abstract. To extract useful information from high-dimensional geometric or structural data, we must find low-dimensional projections that are informative and interesting to look at. The conventional, manual-interaction methods used for this purpose are ineffective when the dimensionality of the data is high, or when the geometric models are complex. Standard methods for determining useful low-dimensional views are either limited to discrete data, or to geometric information embedded in at most three dimensions. Since geometric data embedded in dimensions above three have distinct characteristics and visualization requirements, finding directly applicable techniques is a challenge. We present a comprehensive framework for exploring high-dimensional geometric data motivated by projection pursuit techniques. Our approach augments manual exploration by generating sets of salient views that optimize a customizable family of geometry-sensitive measures. These views serve to reveal novel facets of complex manifolds and equations.

1 Introduction

Humans are adept at recognizing structural relations and patterns among data points, and the shape and geometric features of objects embedded in up to three spatial dimensions. Visualization methods attempt to exploit this perceptual capability to facilitate understanding and communicating complex information. Some common examples are visualizations of vector and scalar fields, state space diagrams of dynamical systems, high-dimensional geometry, and information visualizations that map multi-variate information onto geometric structures such as height maps, graphs of networks, and geometric primitives.

However, when the dimensionality of the information space increases beyond three, it is difficult to meaningfully represent all the relevant attributes of data such as structural patterns or geometric relations. The problems in visualizing and interpreting high-dimensional information arise because the nature of high dimensions is unintuitive and reducing the dimensions of the data for inspection in the familiar dimensions two and three inevitably results in loss of information. This problem is particularly relevant to geometric information because the distortions of structural relations, such as those due to occlusions of important features, or apparent but false intersections of lines and surfaces in the projection, can make it difficult to recover the properties of the models being inspected.

Facilitating the ability to identify key aspects of high-dimensional geometric data[1] is indispensable in many applications and domains of science. For example, while exploring projective and Riemannian geometry, an important consideration is the ability to recognize the global and local object features of complex geometric objects. This is crucial, for example, to understand salient topological properties. Other applications that involve user-intensive activities, such as goal-directed exploration, require tailored approaches for navigating and exploring the associated high-dimensional information spaces.

Although the visualization interfaces for multi-variate data and geometric information in two and three dimensions have matured considerably, high-dimensional geometry-specific issues and requirements have received only limited attention. As a result, the tools for exploring high-dimensional geometry still employ either two-variable combination displays or manual selection methods, which can be tedious and ineffective for exploring complex objects in very high dimensions. In this work, we attack the problem of finding efficient and perceptually-salient goal-directed methods suitable for exploring high-dimensional geometric models; we are motivated by the tremendous potential payoff implicit in exploiting high-performance graphics to make the world of high-dimensional geometry accessible to experts and novices alike.

2 Background

The exploration and recovery of structure in high dimensions has been pursued actively to investigate multi-variate data, multi-dimensional functions and high-dimensional geometry. Investigations in all these domains are concerned with the preservation and discovery of structural relations in the dimensionally reduced representations.

Multi-variate data analysis is usually concerned with the exploration of the most basic data primitive, the point, which represents a vector of information variables. Popular methods used for exposing clusters and groupings of the data in lower dimensions are PCA [1], Euclidean distance-preserving methods like MDS [2], methods based on random projections [3, 4], and tour-based methods that employ a series of space-filling projections [5, 6, 7]. A particularly important and useful approach is called *projection pursuit* [8, 9], which involves the computation of the information content in some low-dimensional projection using a numerical measure (or *projection index*).

The domain of multi-dimensional functions focuses on visualizing higher-order primitives, such as manifolds and hypersurfaces occurring in the visualization of physical simulations, mathematical constructions, and complex dynamical systems [10, 11, 12, 13, 14, 15].

High-dimensional geometry, on the other hand, is concerned with the exploration of mathematical models embedded in high spatial dimensions. The fascination with geometry in high dimensions can be traced back to Abbott's classic treatment in *Flatland* [16]; however, modern efforts to graphically represent high-dimensional models are usually attributed to Noll's proposed methods for hyper-objects [17], and Banchoff's

[1] Terminology: *geometric data* and *geometric models*, are used interchangeably to refer to a geometric object; *geometric information* refers to the higher level construct that includes the geometric data and the embedding space.

visualizations of classical two-manifolds embedded in four dimensions [18, 19]. Since then, extensive research has focused on developing tools and interfaces for exploring geometry and structure in three [20, 21], four [22, 23, 24, 25, 26] and higher-dimensional spaces [27]. Some important contributions of these investigations are 4D lighting models [28,29,30], interfaces for manipulating high-dimensional objects [31,32], and the recent physically-based and multi-modal explorations of the fourth dimension [33,34,35].

Our contribution to visualizing high-dimensional geometry diverges from previous work in that we combine major concepts and methods of projection search and exploration with perception-motivated user interaction tools. The foundation of our approach is the fact that geometric data *and their projections* have inherent structural relations that can be exploited to discover views that are useful both for research-motivated exploratory analyses and for pedagogical, explanatory demonstrations.

3 Framework for Exploring Geometry

Our framework for exploring geometry in high dimensions is based on a variant of projection pursuit that we will refer to as *geometry projection pursuit* (GPP). The goal of GPP is to find projections of geometric models that reveal their important characteristic features[2]. Our framework for GPP includes two important aspects:

- *Optimal view finding.* We are concerned with finding useful low-dimensional views of a given geometric model by optimizing appropriate geometric projection indices. A *combination* of one or more indices can be used based on the goals of the exploration task or the nature of the geometric information. Section 4 discusses optimal view finding and index construction in detail.
- *Visualizing the search space.* Our framework includes the exploitation of displays that might be called "meta-projections" of the measures plotted against the projection variables themselves. This rich visualization is composed of the displays of the measure of the projection indices in all the possible sub-space projections. These *meta-landscapes* are employed to represent the richness of various subspaces, and are treated in Section 5.

Figure 1 summarizes the important steps involved in the process. The framework integrates manual exploration in N-dimensions with optimal view selection. Selecting optimal views involves either optimizing a set of projection indices or exploring a visualization of the projections in all relevant subspaces.

3.1 Geometry-Specific Issues Addressed by the Framework

The distinct nature of geometric information compared to other kinds of data in high dimensions dictates specific requirements for visualization and GPP. Our framework addresses two important issues concerning high-dimensional geometry:

[2] Although the bulk of the literature on projection search and dimensionality reduction focuses on finding informative views in two dimensions, we are concerned with finding 'good' views in two, three, or even four dimensions, where the latter can be explored using efficient human-guided interactive techniques in 3D and 4D viewers.

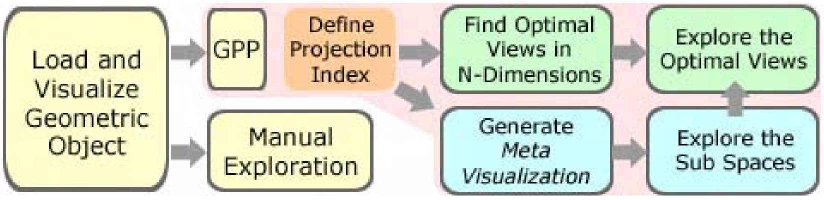

Load and Visualize Geometric Object
GPP
Define Projection Index
Find Optimal Views in N-Dimensions
Explore the Optimal Views
Manual Exploration
Generate *Meta Visualization*
Explore the Sub Spaces

Fig. 1. Framework for visualizing high-dimensional geometry: after loading a geometric model into th visualization interface. The geometry can be explored manually or by index optimization with GPP. GPP assists exploration by automatic generation of optimal views or via *meta visualizations*, which are snapshots of index measures in plotted in parameter subspaces.

– *Nature of Geometric Information.* Unlike point sets of multi-variate data, geometric data are composed of vertices that are associated in some order to form *simplices*, geometric primitives that describe an elementary p-dimensional region using $p+1$ vertices; geometric data have structural information inherent to the constituent data points. Geometric models are typically represented by geometric primitives (simplices) of increasing order of complexity, such as points (vertices), lines (edges), and surfaces (polygons). An important implication of this organization is that the standard approaches to multi-variate data analysis, such as clustering or 'dense' views, are no longer generally applicable due to overlap or occlusion in the projections of the higher order primitives. A typical example is the toroidal 2-manifold embedded in 4D shown in Figure 2: the torus is represented using primitives of increasing complexity but decreasing ambiguity[3].

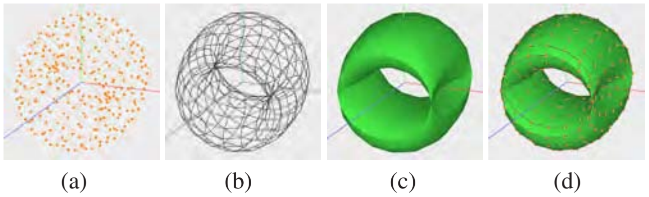


Fig. 2. A torus embedded in 4D, displayed using different orders of simplices as geometric primitives. The vertices of the torus are obtained by the parametric equation: $\mathbf{x} = (\cos(\theta), \sin(\theta), \cos(\phi), \sin(\phi))$, where $\theta, \phi \in [0, 2\pi]$. The different primitives are: (a) points, (b) edges, (c) surfaces, and (d) all together.

– *Specific tasks in GPP:* The design of projection indices used in projection search must be sensitive to some specific requirements in GPP. For example, an important task in GPP is to identify salient topological features of an object. An example of a

[3] The higher order primitives may not always disambiguate the geometric structure; for example, while projecting from from N dimensions to two or three dimensions, false intersections may arise as artifacts of projection. Additional cues like depth-keyed coloring may be used to provide information on the relative depth of the intersecting segments.

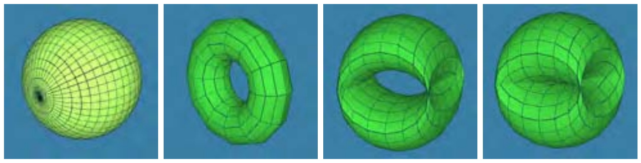


Fig. 3. Holes in geometric objects: (a) A sphere with no holes (genus=0). (b) A 3D-embedded torus with one hole (genus=1). (c) A 4D-embedded torus also with one hole (genus=1). (d) Shows the 4D-embedded torus in a projection for which the hole is no longer visible.

particularly interesting feature is the number of holes in a 2-manifold, which indicates the object's genus. Figure 3 shows a sphere (with zero holes), and orthogonal projections of 3D and 4D embeddings of the torus, each with a single hole. Other relevant features include symmetry, occlusion and branch-cuts.

4 Optimal View Finding in N-Dimensions

In this section we describe the first important component of our framework for GPP: enumerating appropriate information measures and finding one or more low-dimensional views that have the maximum amount of information content.

Traditionally, the problem of defining a "good view" of some 3D object or a complex scene is encountered frequently in object representation and recognition research. Traditional methods employ both image-based and non-image based rendering techniques to determine scene complexity and the overall information in the projected views or silhouettes of objects [36, 37]. Other approaches are derived from object and shape perception literature, which include contrasting models for either 2D or 3D shape encoding (see [37]).

In our scope of work, good or desirable views correspond to the projections (usually in 2D or 3D) that reveal some desired property of the objects embedded in N spatial dimensions. Unlike viewpoint selection in three dimensions, our problem is two-fold, in that not only does a good viewpoint need to be found in N dimensions, but a guideline for suitable projection is required that maximizes an information measure (projection index) in some target projection dimension. In our case the choice of the projection index is usually dictated by the nature of the geometric information or the goal of the exploratory task.

4.1 Geometric Indices

A projection index is a function that computes a numerical value corresponding to the total information content in some projection. Projection pursuit in multi-variate data analysis typically makes use of indices that are based on statistical or entropy-based calculations. Appropriate indices for geometry exploration, however, can (and should) exploit geometry-specific information inherent in the data. Below, we provide a list of representative geometry-based indices, which is by no means exhaustive and can be extended to include other meaningful representations:

Table 1. Examples of projection indices suitable for GPP

Geometric Primitive	Indices
Points = 0-simplices	The most straightforward set of indices are point spread and point-point distances. Methods used for discrete data points are also applicable. Given a set of N-dimensional points, $P = (p_1, p_2, .., p_m)$ we define: $$I_{\text{spread}} = \sum(\lvert p_i \rvert)$$ $$I_{\text{point\_distances}} = \sum_{i \neq j}(\lvert p_i - p_j \rvert)$$
Edges = 1-simplices	Edges provide our first measure that directly represents topological structure of a geometric object. Some suitable measures are: $$I_{\text{edge\_intersection}} = \sum F(e_{ij}, e_{mn})$$ $$I_{\text{edge\_length}} = \sum_{i \neq j} e_{ij}$$ where e_{ij} and e_{mn} are edges between pairs of distinct vertices i, j and m, n; the function $F(e_{ij}, e_{mn})$ determines if there is intersection between the two edges.
Faces = 2-simplices	2-simplices represent a two-dimensional manifold or surface patch in terms of triangles. A simple measure given the surface elements represented by the vertex triplets (i, j, k) is: $$I_{\text{surface\_area}} = \sum \text{Area}_{ijk}$$ Some other suitable measures are those which compute the number of occlusions based on 3D depth buffer hits, or face/face intersections in the projections.
Volume (or Balls) = 3-simplices	A 3-simplex is a three-dimensional volume element. Projection criteria can be selected to minimize/maximize the projected volume, or compute the occlusions based on 3D or 4D depth buffer hits.
Hypervolume = 4-simplices	Interesting views can correspond to projections that maximize the polygonal surface area or volume in the 4D projection, which can be explored in special tools that allow full 4D rotations. This is distinct from maximizing the 2D or 3D projections of these quantities.

Once one or more indices are chosen, we use the procedure for GPP shown in Figure 4. The step involving viewpoint selection in the N-dimensional space can be achieved by different methods. For example, random configurations of the $N \times N$ rotation matrix

Table 1. *(continued)*

Geometric Primitive	Indices
Non-geometric indices	Image-based rendering methods can be combined with the above-mentioned geometric indices to obtain some suitable metric. For example, we have a hole-finding technique that utilizes pixel-color information to identify holes of arbitrary shapes and sizes.
Space walking Method	Space walking [38] is a method for navigating on topological manifolds while always projecting the maximal viewable area onto the screen plane. This solution is similar to a local projection pursuit approach because, at each step, as the observer rotates the model, the control system determines the transformation for orienting the local tangent space around the point of interest parallel to the screen display plane.

corresponding to random rotations can be evaluated for optimal views. Another method is to utilize an optimizer to find the best combination of high-dimensional rotations.

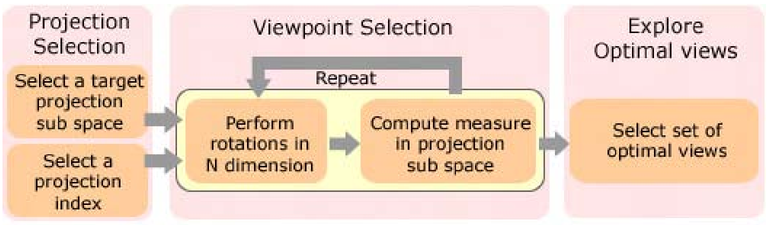

Projection Selection
Select a target projection sub space
Select a projection index
Viewpoint Selection
Repeat
Perform rotations in N dimension
Compute measure in projection sub space
Explore Optimal views
Select set of optimal views

Fig. 4. Finding optimal views by GPP. First, a candidate target projection space is specified. Next, different projections in the target space are obtained by repeatedly performing rotations in N dimensions and computing the measure of the indices. Finally, the projections with high values of the indices are selected.

4.2 N-Dimensional Optimization Methods

Once a set of indices has been specified, optimizations in the high-dimensional space are performed to find a series of maxima (or minima, depending on the type of index). The general optimization problem can be formalized as follows:

ND optimization problem: Given a function $f(\theta_i, i = 1, 2, \ldots, N(N-1)/2)$ that accepts $N(N-1)/2$ rotation angles, one in each of the canonical two-dimensional planar subspaces spanned by two orthogonal axes, and that returns the value of a projection index, find the set of orientation angles $\{\theta_i'\}$ such that for all $\theta_i \in [0, 2\pi]$, the function satisfies $f(\{\theta_i'\}) \geq f(\{\theta_i\})$.

An important consideration in the ND optimization problem is that search in an angle θ_i must be restricted to transformations that actually effect a change in the chosen projection space. Furthermore, as the dimensionality of the embedding space increases (e.g., $N \geq 5$), the full optimization tends to run very slowly, and finding all local optima in the search space is a challenge. The problem can be approached, e.g., by simulated annealing but that can also be very time consuming. While we plan to implement large-subspace optimization in future approaches, here we focus on an effective approximate sequential search method using low-dimensional subspaces. Our procedures for $N(N-1)/2$ one-dimensional subspace search and pairwise searches of the $N(N-1)(N^2-N-2)/8$ two-dimensional subspaces of the θ_i are thus as follows:

1D (2D) approximate optimization method: Compute the optimization measure $f(\{\theta_i\})$ for one choice of i (or (i,j)), and find the optimal point. Fix that point and repeat for the next i (or (i,j)). Repeat. The result will deviate from a true optimum but will give good estimate for beginning a less time-consuming search in the neighborhood of the resulting point. Variants include performing local subspace optimization incrementally in neighboring subspaces.

We utilize the Conjugate Gradient method for finding the local optima. We have found that almost-optimal views, or local optima, are equally interesting as they may reveal additional perceptual aspects of a geometric model that the mathematically optimal views may obscure. We therefore track all local minima found by the algorithm. While we have not accumulated extensive performance data, computing the 2D subspaces of a 2500-vertex surface in 9D takes about 3 minutes; further performance optimization strategies are being studied.

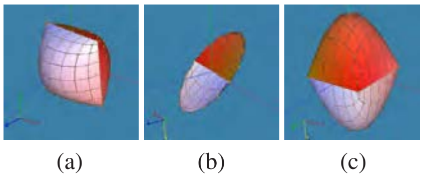


Fig. 5. Different projections of a 4D pillow-shaped object whose parametric equation is $\{\cos(\theta), \cos(\phi), \sin(\theta)\sin(\phi), \sin(\theta)\sin(\phi)\}$. The projections into the 3D space with axes $(1,2,3)$ are: (a) The default view (b) A flattened view. (c) An optimal view based on a maximal area projection found by rotations in subspaces $(2,3)$ and $(2,4)$.

4.3 Results of Optimal View Finding

We now discuss some results obtained by applying our methods to objects in high dimensions. Figure 5 shows different views of a pillow-shaped 4D object obtained by maximizing the area of the object projected into three dimensions.

Figure 6 contrasts the optimal views of a 4D-embedded torus based on two different projection indices: (a) projection edge length, and (b) projection area.

Figure 7 shows a more complex example, the cubic CP^2 torus, which consists of nine two-dimensional parametric patches embedded in nine dimensions. Figure 7(b) shows

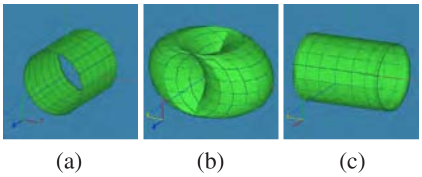


Fig. 6. Different projections of a 4D-embedded torus in the projection space with indices $(1,2,3)$ (a) The default view. (b) View found by maximizing edge-length between projected vertices; this view was found by rotations in subspaces $(1,2)$ and $(1,4)$. (c) A view based on maximal-area projection found by rotations in subspaces $(1,3)$ and $(1,4)$.

an interesting projection in the 3D target space with axes $(5,7,9)$. The optimal view finding method assisted in discovering the object's single hole (genus=1)[4].

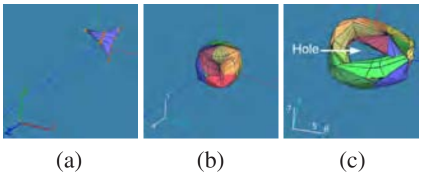


Fig. 7. Different projections of a two-dimensional manifold, actually a torus, embedded in nine-dimensional space: (a) The default view in projection space with indices $(1,2,3)$. (b) Default view in the projection with indices $(5,7,9)$. (c) A view in the space $(5,7,9)$ that exposes a hole; the view was found by maximizing the index based on distances between vertices in 3D, a point spread measure, and the area of the projected 3D surface. The exposure of the hole was maximized by exploring the neighborhoods of subspaces $(4,9)$, $(6,9)$ and $(8,9)$.

5 Visualizing the Search Space of Optimal Views

The second significant part of the framework is the visualization of the search space of all possible projections. We refer to this as the *meta visualization* because it provides additional information on the measures of projection indices. Figure 8 shows a snap shot of the visualization interface and the meta visualizations.

The meta visualizations provide a quick overview of the richness of different subspaces in the form of thumbnail charts as shown in Figure 8(b). The thumbnails are generated by automatically roaming through all the subspaces, e.g., by performing rotations in the subspaces and recording values of indices. For objects in very high dimensions, there are possibly hundreds of such thumbnails, and to speed up computation, the user can either reduce the resolution of the charts or specify what subspaces are to be explored.

The thumbnails can be magnified into higher resolution charts by clicking on them in the thumbnails panel. An example is shown in 8(c), in which the horizontal and

[4] While the properties of the object were known beforehand, optimal view-finding aided in finding the large hole, which is very tedious to discover manually.

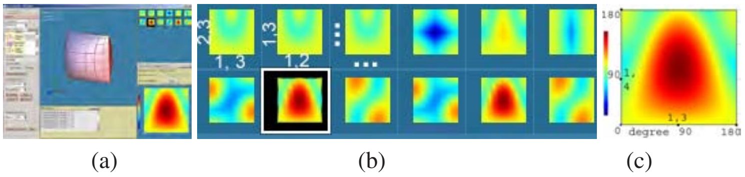


Fig. 8. The visualization interface. (a) The visualization application. (b) The closeup of the thumbnail charts representing the measure of a projection index in all 2D subspaces. (c) A magnified view of the thumbnail for subspace $(1,3)$ and $(1,4)$ (thumbnail with black background). The palette corresponds to the values of the index: dark red = high value, dark blue = low value.

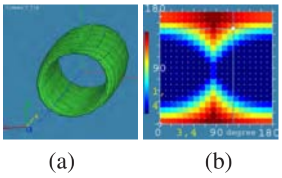


Fig. 9. (a) The hole in a 4D-embedded torus. (b) Its meta visualization in one 2D subspace, (1,4) combined with (3,4). The dark red areas correspond to the hole in the screen plane. The hole was identified by comparing the pixel colors for the rendered background and foreground.

vertical axes correspond to rotations in two different subspaces, respectively. Clicking and dragging anywhere on the magnified chart applies the corresponding rotations to the object in the main window. This allows quick investigation of optimal views and their neighborhoods. The chart in Figure 8(c) shows a well-defined region of maxima of the index; other meta visualizations can of course be much more complex. Another example in shown in figure 9, which shows a 4D torus in a projection that exposes its hole, and the meta visualization chart that depicts the associated subspace.

6 Conclusion and Future Work

We have presented a framework for visualizing and exploring high-dimensional geometry using a general philosophy that we refer to as geometric projection pursuit (GPP). We present several relevant examples of the method targeting the fact that understanding geometric objects in high dimensions has unique and specific requirements in terms of visualization and exploration. Projection search based on optimization of geometry-driven indices is useful in exposing features of the object that are relevant to their geometric and topological properties, and to goal-specific tasks requiring the elucidation of such properties. While we have given examples of some families of simple geometry-based projection indices, additional arbitrarily complex problem-specific measures can easily be added to our system. We plan to experiment further with indices that expose characteristics like symmetry, occlusion, branch-cuts and non-rigid deformations. It may also be possible to extend the framework to non-geometric data sets and thus to enhance techniques such as N-dimensional brushing [39]. Another extremely promising area for future work is the development of intelligent navigational methods to create smooth transitions among the extensive set of optimal views found by the system.

This research was supported in part by National Science Foundation grants CCR-0204112 and IIS-0430730.

References

1. Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer Series in Statistics. Springer, New York, NY, USA (2002)
2. Kruskal, J.B., Wish, M.: Multidimensional Scaling. Sage University Paper series on Quantitative Application in the Social Sciences. Sage Publications, Beverly Hills and London (1978)
3. Dasgupta, S.: Experiments with random projection. In: UAI 2000. Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, USA, pp. 143–151. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000)
4. Fern, X., Brodley, C.: Random projection for high dimensional data clustering: A cluster ensemble approach. In: ICML 2003. Twentieth International Conference on Machine Learning (2003)
5. Asimov, D.: The grand tour: a tool for viewing multidimensional data. SIAM J. Sci. Stat. Comput. 6, 128–143 (1985)
6. Cook, D., Buja, A., Cabrera, J., Hurley, H.: Grand tour and projection pursuit. J. of Computational and Graphical Statistics 2, 225–250 (1995)
7. Young, F.W., Rheingans, P.: Visualizing structure in high-dimensional multivariate data. The IBM Journal of Research and Development 35 (1991)
8. Huber, P.J.: Projection pursuit. The Annals of Statistics 13, 435–475 (1985)
9. Friedman, J.H.: Exploratory projection pursuit. J. of the American Statistical Association 82, 249–266 (1987)
10. Mihalisin, T., Timlin, J., Schwegler, J.: Visualizing multivariate functions, data, and distributions. IEEE Comput. Graph. Appl. 11, 28–35 (1991)
11. van Wijk, J.J., van Liere, R.: Hyperslice: visualization of scalar functions of many variables. In: VIS 1993. Proceedings of the 4th conference on Visualization 1993, pp. 119–125 (1993)
12. Liere, R., Wijk, J.J.: Visualization of multi-dimensional scalar functions using hyperslice. Technical report, Computer Science Department, CWI, Amsterdam, The Netherlands (1994)
13. dos Santos, S.R., Brodlie, K.W.: Visualizing and investigating multidimensional functions. In: VISSYM 202, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, 173 (2002)
14. Simionescu, P.A., Beale, D.: Visualization of hypersurfaces and multivariable (objective) functions by partial global optimization. The Visual Computer 20, 665–681 (2004)
15. Wegenkittl, R., Löffelmann, H., Gröller, E.: Visualizing the behavior of higher dimensional dynamical systems. In: VIS 1997, pp. 119–126. IEEE Computer Society Press, Los Alamitos, CA, USA (1997)
16. Abbott, E.A.: Flatland: A Romance of Many Dimensions. Dover Publications Ltd. (1992)
17. Noll, A.M.: A computer technique for displaying n-dimensional hyperobjects. Commun. ACM 10, 469–473 (1967)
18. Banchoff, T.: Discovering the Fourth Dimension. Prime Computer, Inc., Natick, MA (1987)
19. Banchoff, T.: Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions. W. H. Freeman & Co., New York, NY, USA (1990)

20. Phillips, M., Levy, S., Munzner, T.: Geomview: An interactive geometry viewer. Notices of the Amer. Math. Society 40, 985–988 (1993)
21. Hanson, A.J., Munzner, T., Francis, G.: Interactive methods for visualizable geometry. Computer 27, 73–83 (1994)
22. Hoffmann, C.M., Zhou, J.: Some techniques for visualizing surfaces in four-dimensional space. Comput. Aided Des. 23, 83–91 (1991)
23. Hollasch, S.: Four-space visualization of 4D objects. Master's thesis, Arizona State University (1991)
24. Banks, D.: Interactive manipulation and display of surfaces in four dimensions. In: SI3D 1992. Proceedings of the 1992 symposium on Interactive 3D graphics, pp. 197–207. ACM Press, New York (1992)
25. Roseman, D.: Twisting and turning in four dimensions, Video animation, Department of Mathematics, University of Iowa, and the Geometry Center (1993)
26. Hanson, A.J., Ishkov, K.I., Ma, J.H.: Meshview: Visualizing the fourth dimension. Technical report, Indiana University (1999)
27. Hanson, A.J.: Graphics Gems IV. In: Geometry for n-dimensional graphics, pp. 149–170. Academic Press, London (1994)
28. Hanson, A.J., Heng, P.A.: Visualizing the fourth dimension using geometry and light. In: VIS 1991, pp. 321–328. IEEE Computer Society Press, Los Alamitos, CA, USA (1991)
29. Carey, S.A., Burton, R.P., Campbell, D.M.: Shades of a higher dimension. Computer Graphics World, 93–94 (1987)
30. Steiner, K.V., Burton, R.P.: Hidden volumes: The 4th dimension. Computer Graphics World, 71–74 (1987)
31. Duffin, K.L., Barrett, W.A.: Spiders: a new user interface for rotation and visualization of n-dimensional point sets. In: VIS 1994. Proceedings of the conference on Visualization 1994, pp. 205–211. IEEE Computer Society Press, Los Alamitos, CA, USA (1994)
32. Hanson, A.J.: Graphics Gems V. In: Rotations for n-dimensional graphics, pp. 55–64. Academic Press, Cambridge, MA (1995)
33. Hanson, A.J., Zhang, H.: Multimodal exploration of the fourth dimension. In: Proceedings of IEEE Visualization, pp. 263–270. IEEE Computer Society Press, Los Alamitos (2005)
34. Zhang, H., Hanson, A.J.: Physically interacting with four dimensions. In: ISVC. 4291, pp. 232–242 (2006)
35. Zhang, H., Hanson, A.J.: Shadow driven editing (accepted for publication). In: Proceedings of IEEE Visualization, IEEE Computer Society Press, Los Alamitos (2007)
36. Vázquez, P.P., Feixas, M., Sbert, M., Llobet, A.: Viewpoint entropy: a new tool for obtaining good views of molecules. In: VISSYM 2002. Proceedings of the symposium on Data Visualisation 2002, pp. 183–188. Eurographics Association, Aire-la-Ville, Switzerland (2002)
37. Polonsky, O., Patane, G., Biasotti, S., Gotsman, C., Spagnuolo, M.: What's in an image: Towards the computation of the "best" view of an object. The Visual Computer 21, 8–10 (2005)
38. Hanson, A.J., Ma, H.: Space walking. In: VIS 1995. Proceedings of the 6th conference on Visualization 1995, p. 126. IEEE Computer Society Press, Los Alamitos (1995)
39. Martin, A.R., Ward, M.O.: High dimensional brushing for interactive exploration of multivariate data. In: VIS 1995. Proceedings of the 6th conference on Visualization 1995, p. 271. IEEE Computer Society Press, Los Alamitos (1995)

Direct Extraction of Normal Mapped Meshes from Volume Data

Mark Barry and Zoë Wood

California Polytechnic State University

Abstract. We describe a method of *directly* extracting a simplified contour surface along with detailed normal maps from volume data in one fast and integrated process. A robust dual contouring algorithm is used for efficiently extracting a high-quality "crack-free" simplified surface from volume data. As each polygon is generated, the normal map is simultaneously generated. An underlying octree data structure reduces the search space required for high to low resolution surface normal mapping. The process quickly yields simplified meshes fitted with normal maps that accurately resemble their complex equivalents.

1 Introduction

Volume data is used in scientific and medical imaging as well as in the process of scanning physical objects [1]. This data structure has the benefit that it can describe surfaces, spatial extents and interior structures. The most popular method for viewing volume data requires extracting a *contour* surface or isosurface. Typically an extracted contour surface of high detail will contain a very large number of polygons. Approximating the original surface by reducing the extracted mesh's resolution is desirable.

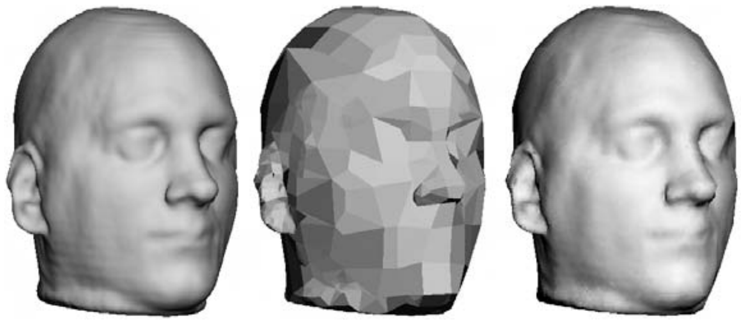


Fig. 1. A human head extracted from a 130 x 128 x 128 volume. All meshes are dual contour surfaces. Left: 56,637 quads. Center: 1,406 quads. Right: same as center but with normal maps, generated with our algorithm in about 1 second.

A technique known as *normal mapping* is one way to achieve the appearance of fine detail on a low resolution version of the surface. Current normal mapping algorithms are applicable to volume data only after a surface has been extracted, requiring a lengthy, multi-step process. We demonstrate a novel method of greatly shortcutting the current process by directly extracting a simplified surface along with normal maps in one integrated process, using the volume data structure to our advantage. The process is very fast - capable of extracting normal mapped meshes of various resolutions in the time of one second on average, giving the user the ability to quickly choose a mesh at the desired level of detail. A low resolution mesh extracted using our algorithm can render almost fourteen times faster then its high resolution equivalents. Yet even normal mapped low resolution meshes, with 92% fewer polygons, appear nearly identical to their high resolution equivalents (see Figure 5).

2 Previous Work

2.1 Isosurface Extraction

The *Marching Cubes* (MC) algorithm [2] can generate a high-quality mesh that captures a contour's fine details, but commonly generates very large meshes, ranging in the millions of polygons. The *Extended Marching Cubes* algorithm [3] presents an improvement over the MC algorithm by generating additional contour vertices *within* cubes, which results in higher-quality meshes that capture features such as sharp edges. The downside of these methods is the generation of a high resolution mesh in order to display the contour surface's finest details.

2.2 Adaptive Contouring

Adaptive polygonization or adaptive contouring is one method of dealing with high resolution meshes produced by MC. An *octree* structure [4] [5] [6] can be generated to adaptively represent the volume data. The challenge of adaptive contouring on simplified octrees is that the resulting polygonal mesh is not "water-tight". The problem of generating a closed polygonal mesh from a simplified octree has been extensively studied [7] [8] [5]. In general the solutions proposed require restrictions on how the octree is simplified.

Ju et. al. [9] present an alternative to previous adaptive contouring methods using dual contouring. Dual contouring methods generate a contour vertex *within* a cube's interior. The algorithm presented by Ju et. al. is an *adaptive* dual contouring method that produces a "crack-free" contour surface from a simplified octree. Unlike other adaptive contouring methods, this dual contouring method does not impose restrictions on how an octree is simplified nor requires any sort of crack patching. In addition, it performs as well as [3] in terms of preserving distinct features such as sharp edges. In order to adaptively simplify the octree data structure that represents the volume data, the authors use the quadric error functions (QEFs) introduced in [10]. Fixes and improvements to [9] are discussed in [11], [12], and [13]. Due to its ability to directly extract adaptive high quality meshes from volume data, we use this algorithm to create our low resolution meshes.

2.3 Simplification and Normal Maps

A popular approach to dealing with high resolution output meshes from MC involves first extracting a high resolution surface and later simplifying it [14] [10] [15]. The low resolution simplified meshes are often a gross approximation of the original mesh's high resolution appearance. One common approach to achieve the appearance of a high resolution mesh using simplified geometry is through the use of a normal map [16] [17]. One of the most popular methods to generate normal maps is presented in [18]. The process first assumes that a high resolution mesh has been simplified into a low resolution approximation. For each polygon in the low resolution mesh, a texture map is created and sampled. For each sample, a ray is cast to determine the nearest corresponding point on the high resolution mesh. This is a costly operation that can involve searching every polygon in the high resolution mesh for a possible intersection with the ray. In [18], the search is optimized by partitioning the search space into cells. Note that creating this spatial partitioning data structure essentially requires the re-creation of a volume data structure if the mesh came from a volume.

In contrast, our method avoids the initial step of extracting of a high resolution mesh. By using adaptive contouring, the algorithm is designed to directly extract the final simplified mesh. Another advantage to using our algorithm is a limited search space for mapping fine to coarse features, which is naturally implemented using the existing octree data structure. Collecting normals to create a normal map is limited to "searching" only four octree cubes that a polygon spans (see Section 3.2).

The authors of [19] present a method of constructing a progressive mesh such that all meshes in the progressive mesh sequence share a common texture parameterization. Thus a normal mapped progressive mesh can easily be created. Our work generates a new set of normal maps for each mesh extracted, while the progressive mesh approach uses a single normal map for all mesh resolutions. The single parameterization approach may be more efficient in the end but takes a substantially longer time to construct for all mesh resolutions.

3 Algorithm Overview and Contributions

We present a method to extract geometrically simplified, low polygon meshes with their accompanying normal maps from volume data. The algorithm follows these simple stages:

1. Create an adaptive octree from the original volume using QEF.
2. Extract the dual contour using [9].
3. For each low resolution polygon, create the associated normal map using the octree data structure.

Steps one and two follow the work of [9] and thus are not presented here in any detail. Step three is our main contribution as we know of no other method to directly extract normal mapped meshes from volume data. Not only is our method novel in its application to the volume domain, but our method of generating normal maps by gathering the fine data in a fine-to-coarse approach (versus a coarse-to-fine approach) is likewise new (see Section 3.3). We discuss the details of the algorithm in the following sections. For complete details on the algorithm, please see [20].

3.1 Dual Contouring of Volume Data With Normal Map Extraction

Our goal is to extract a geometrically simplified surface with a normal map directly from volume data. In addition to being a robust adaptive contouring algorithm, dual contouring is ideally structured for easy extraction of normal maps as well. Thus we use the dual contouring method presented in [9] for both the creation of the octree using QEF and the surface extraction. Dual contouring builds the final mesh by connecting four minimizing vertices found within four neighboring cubes in the octree. If the adaptive structure of the octree includes neighboring cubes of different levels in the hierarchy, triangles are generated. The implementation treats triangles as quads having two vertices fused together. We therefore refer to all surface polygons as *quads*. To make up for the coarse appearance of the mesh alone, normal maps are created and applied to quads as they are generated.

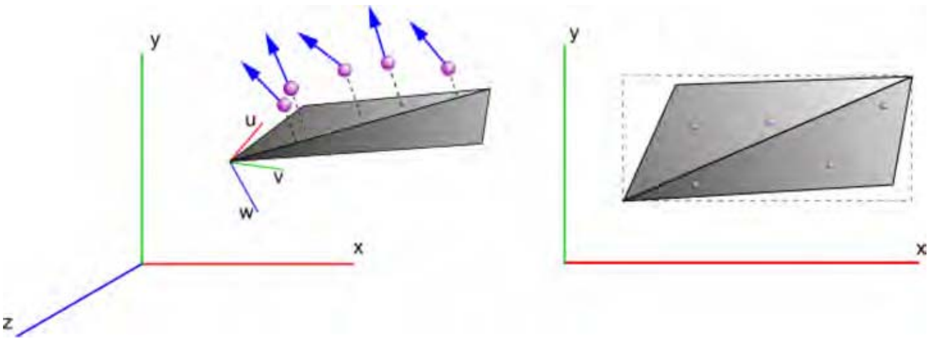


Fig. 2. Projection of fine-level contour vertices and their associated normals onto a contour quad. The dotted bounding box on the right represents the normal map. Note that there are typically *many* more fine-level samples then shown here. See Figure 3.

3.2 Generating Normal Maps

A normal map can be generated and stored in many different polygonal formats; triangles, rectangles, packed charts, etc. We choose to generate a normal map per quad. Normal map generation starts with the creation of a rectangular map.

The dimensions of the rectangle are calculated based on the size of the quad. Because the quad is defined in three-dimensional space and the normal map is only two-dimensional, the quad must be projected onto a two dimensional space.

The quad's orthonormal basis is computed and used to calculate the projection (see Figure 2). The orthonormal basis matrix is composed of three mutually perpendicular row vectors: $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$. We use the cross product of the quad's diagonals to compute $\boldsymbol{w}$, leaving the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ to be computed. There are no constraints on the orientation of $\boldsymbol{u}$ nor $\boldsymbol{v}$, as long as all three vectors are mutually perpendicular. Choosing values for $\boldsymbol{u}$ and $\boldsymbol{v}$ translates into how the quad will be oriented on the normal map and determines the normal map's dimensions required to bound the quad. For the most efficient use of space, the $\boldsymbol{u}$ vector can be oriented parallel with the quad's longest edge. The $\boldsymbol{v}$ vector can then be computed from $\boldsymbol{w} \times \boldsymbol{u}$. Note that not all quads are planar, however, we have found that using the cross product of the quad's diagonals is a reasonable mapping for our algorithm.

The x-y-z coordinate of each of the quad's vertices is multiplied with the basis matrix to perform the projection. After the projection, the z-component of the quad's vertices can be dropped and the coordinates treated as two-dimensional. The two-dimensional quad is bounded with a rectangle representing the normal map. Next the normal map is filled with the appropriate normal vector values. Sampling is chosen to match the sampling rate of the volume data, thus approximately one texel is created on the normal map per finest level voxel spanning the quad.

The normal maps act to fill in the fine detail that the simplified contour surface lacks. The finest detail that can be obtained comes from the contour vertices generated at the leaves of the octree. Note that each finest level voxel has a gradient computed using a finite difference, and finest level vertices and normals are computed as in [9]. The process of generating a normal map involves capturing the normals from these fine-level contour vertices. Recall that contour vertices are contained within an octree cube and surface polygons are formed by connecting four minimizing vertices, thus each polygon spans four octree cubes. Normals for a low resolution quad are obtained by collecting all the fine-level contour vertices contained within four spanning cubes in the octree. Each cube calls a recursive function for each of its child cubes, traversing down the octree of sub-cubes until reaching a leaf cube. At a leaf cube, contour vertices, if any, are extracted and returned in an array. As all calls to the recursive function return, the arrays of fine-level contour vertices are combined. These fine-level contour vertices are then projected onto the quad's normal map.

3.3 Normal Map Sampling

The projection step samples normals across the normal map but may leave remaining texels' normal values undefined. One method to completely fill the normal map would be to interpolate the normal values at the projected points across undefined texels. Many methods of interpolating scattered data are available [21] but we found that using a very simple sampling method was sufficient for all

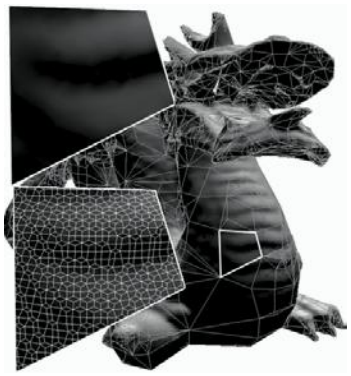


Fig. 3. Close-up view of one normal map of the dragon, shown at a coarse level (16K) and the finest level (225K). The finest level view shows the wireframe projected onto the normal map extracted for the coarse mesh. The normal map shown is approximately 30 by 30 texels, closely matching the resolution of the finest level mesh.

example meshes shown. This approximation works by first initializing the entire normal map with the contour quad's normal. After initialization, the fine-level vertices are projected onto the normal map as before, overwriting any initialized texels. Such a simple mapping works well due to the QEF used to generate the low resolution surface, where a low resolution polygon closely matches the high resolution surface. Any areas where the surface may fold back on itself should remain at a high resolution during simplification and thus not cause problems with normal map creation. As shown in figure 3, the normal map produced by this sampling method smoothly matches the original data. Other interpolation methods or even the pull-push method of [22] could be explored in future work, however, our simplified approximation works well due to choosing a sampling rate which matches the resolution of the volume and the qualities of the QEF surface simplification.

Note that most previous work [18] [19] for generating normal maps use a ray shooting technique, where texel samples on the coarse polygon are sampled by searching for an associated point on the high resolution mesh. In this way, these approaches generate a densely sampled normal map for the coarse mesh, where many of the sampled normals will be an interpolated normal from an interior point on the face of the fine mesh. We take the opposite approach by projecting the high resolution sample points to the coarse mesh. This approach is arguably faster, not only because of the octree data structure but because of the number of samples being mapped from fine to coarse (versus the many texel samples generated across the coarse face needing an associated mapping to the fine mesh). Our approach could be considered as less accurate, however, the only sample points that are lost are the interpolated normals from the interiors of the high resolution faces. We found that our fine-to-coarse approach performed not only very quickly but also with visually pleasing results.

The following outlines the main steps in generating normal maps:

1. Use the dual contouring algorithm to extract quads.
2. For each quad created, generate its normal map.
   (a) Calculate the quad's orthonormal basis.
   (b) Transform/project the quad's vertices into two-dimensional space by using the quad's orthonormal basis.
   (c) Collect all fine-level contour vertices (position and normal) for the quad.
      i. For each of the four cubes which the quad spans, recursively traverse the octree to the leaves and append vertices to a common array.
   (d) Project all fine-level contour vertices onto the quad using the quad's orthonormal basis, (associated normals are unchanged).
   (e) Bound the projected quad with a rectangular normal map.
      i. Find the max and min x and y values of vertices in the array.
      ii. Allocate normal map texels of size x extent by y extent. Initialize all texels to quad's normal.
      iii. For each projected contour vertex: Copy its normal into the normal map using integer-rounded x and y of its position.

4 Results

One important measure of success is how well the low resolution normal mapped surface resembles the original high resolution surface without normal maps. Figures 1, 4 - 8 compare a low resolution and normal mapped mesh to its high resolution equivalent, demonstrating the excellent results from using our algorithm. Another factor to consider is the speed at which a mesh, along with its normal maps, can be extracted from the volume. The longest extraction time for our algorithm was for the dragon mesh from a 356 x 161 x 251 volume, along with normal maps, at the highest resolution. This operation takes a little over 3 seconds. Extracting simpler meshes (which would be the more frequent request) takes less time - an average of 1 second for the meshes shown in the following figures. This speed makes it easy for a user to quickly switch between varying levels of mesh resolution. A user may wish to begin with a low resolution mesh for fast display and cursory examination. Then as the user desires more geometric accuracy, a higher resolution mesh may be quickly extracted. It is very important to emphasize that a low resolution mesh is *directly extracted*.

A fast rendering time is one of the main motivations for using normal mapped surfaces. Table 1 compares the rendering performance of high resolution meshes to low resolution, normal mapped, meshes. It is clear that rendering low resolution normal mapped meshes is a lot faster, yet they retain a significant amount of detail as shown in Figures 1, 4 - 8. For the dragon example, even on the lowest resolution meshes, the scales are still clearly defined. The flat-shaded meshes alone do not come near to displaying that kind of detail. At the lowest resolutions, maintaining correct topology of the original high resolution mesh begins to fail. In the extreme example of Figure 6 (Right), the flat-shaded mesh alone grossly resembles the dragon model; applying normal maps restores most of the

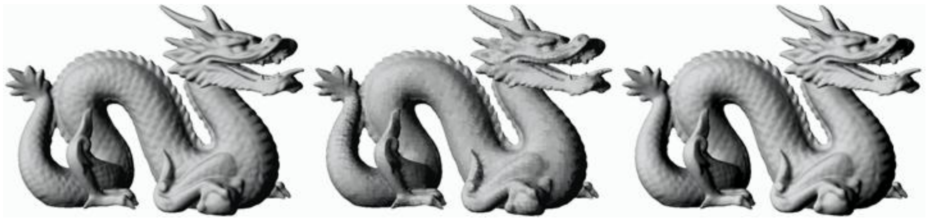


Fig. 4. All dragon models extracted from a 356 x 161 x 251 volume. All meshes are dual contour surfaces. Left: high resolution – 225,467 quads. Center: low resolution – 43,850 quads. Right: same as center but with normal maps applied.

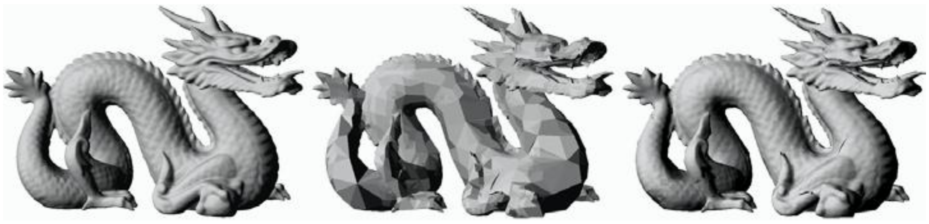


Fig. 5. Left: 225,467 quads. Center: 16,388 quads. Right: same as center but with normal maps applied.

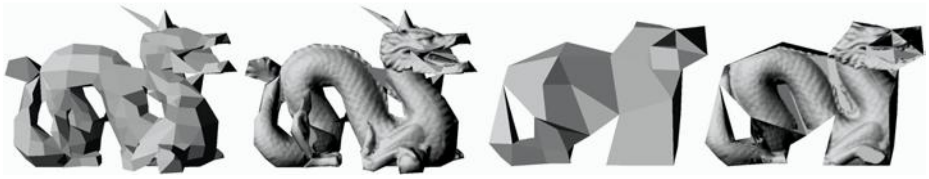


Fig. 6. Left two: flat-shaded and normal mapped dual contour surface (558 quads). Right two: flat-shaded and normal mapped dual contour surface (65 quads).

finer details. Figures of the mouse embryo and human head demonstrate that the process works equally well for medical imaging applications.

4.1 Limitations

Though we describe a method that is fast and yields excellent visual results, there is always room for improvement. One inefficiency in the system as it stands is that each polygon allocates its own normal map. Ideally the collection of normal maps could be packed into one or a few *atlases* [18] [19]. Though the result would be a more efficient use of memory space, the packing process is computationally intensive. The memory occupancy for the normal data stored in the normal maps created by the current algorithm for the dragon models shown in figure 5 and 6 range from 7 megabytes for the 225K mesh to 2 megabytes for the 65 face mesh.

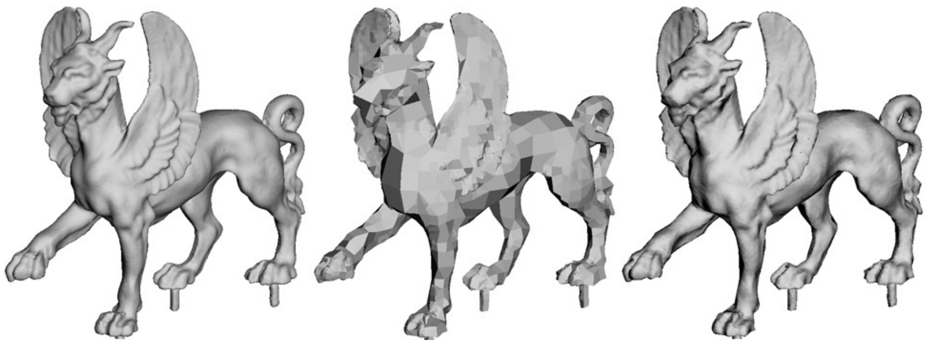


Fig. 7. A mythical creature extracted from a 316 x 148 x 332 volume. All meshes are dual contour surfaces. Left: 150,823 quads. Center: 10,950 quads. Right: same as center but with normal maps.

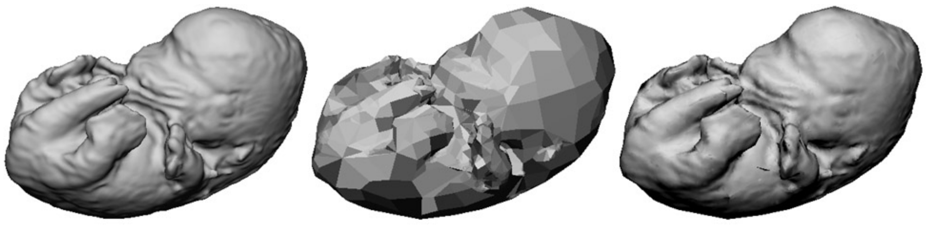


Fig. 8. A mouse embryo extracted from a 256 x 128 x 128 volume. All meshes are dual contour surfaces. Left: 64,896 quads. Center: 3,035 quads. Right: same as center but with normal maps.

Table 1. Performance data for the examples shown in Figures 1, 4 - 8

	Quad Count			Render Time (ms)		
	Hi-Res	Lo-Res	Reduction	Hi-Res	Lo-Res	Speedup
Figure 1	56,637	1,406	97.5%	91	3	30.3
Figure 4	225,467	43,850	80.6%	360	90	4.0
Figure 5	225,467	16,388	92.7%	360	26	13.8
Figure 6 (L)	225,467	558	99.8%	360	1	360
Figure 6 (R)	225,467	65	99.97%	360	0.3	1200
Figure 7	150,823	10,950	92.7%	245	22	11.1
Figure 8	64,896	3,035	95.3%	103	6	17.2

As mentioned in Section 3.2 we use the cross product of the quad's diagonal to create our planar mapping. Our results on numerous input meshes show that such a mapping is sufficient for our application, however, other parameterizations could be explored. In addition, our normal map sampling method is very simple, more complex methods for interpolating normal data could be explored.

5 Conclusion and Future Work

We describe a method of directly extracting normal mapped contour surfaces from volume data. This method greatly shortcuts the current multi-step process of extracting a high resolution mesh, simplifying it, and generating normal maps. The process of generating normal maps is greatly streamlined due to the use of an octree data structure and the dual contouring algorithm. The visual results of this process are of excellent quality - the low resolution normal mapped meshes closely resemble their high resolution equivalents. This process is also very fast - generating and displaying a normal mapped surface in less than a few seconds.

Future work includes extending our algorithm for use in computer games. Destructible game objects could be implemented similar to the work presented in [9]. Additionally, though we use dual contouring's QEF metric for mesh simplification, the use of other error metrics could be explored. The QEF metric aims to preserve the most prominent features such as sharp edges. Perhaps preserving certain sharp edges would not be necessary if they were instead captured in a normal map. Finally, blending normals across large quad boundaries could be improved as small inconsistencies are occasionally seen.

References

1. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: SIGGRAPH 1996. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 303–312. ACM Press, New York (1996)
2. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH 1987. Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 163–169. ACM Press, New York (1987)
3. Kobbelt, L.P., Botsch, M., Schwanecke, U., Seidel, H.P.: Feature sensitive surface extraction from volume data. In: SIGGRAPH 2001. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 57–66. ACM Press, New York (2001)
4. Meagher, D.: Geometric modeling using octree encoding. In: Computer Graphics and Image Processing, pp. 129–147 (1982)
5. Wilhelms, J., Gelder, A.V.: Octrees for faster isosurface generation. ACM Trans. Graph. 11, 201–227 (1992)
6. Shekhar, R., Fayyad, E., Yagel, R., Cornhill, J.F.: Octree-based decimation of marching cubes surfaces. In: VIS 1996. Proceedings of the 7th conference on Visualization 1996, p. 335. IEEE Computer Society Press, Los Alamitos, CA, USA (1996)
7. Cignoni, P., Ganovelli, F., Montani, C., Scopigno, R.: Reconstruction of topologically correct and adaptive trilinear isosurfaces. Computers and Graphics 24, 399–418 (2000)

8. Frisken, S.F., Perry, R.N., Rockwood, A.P., Jones, T.R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. In: SIGGRAPH 2000. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 249–254. ACM Press/Addison-Wesley Publishing Co., New York (2000)
9. Ju, T., Losasso, F., Schaefer, S., Warren, J.: Dual contouring of hermite data. In: SIGGRAPH 2002. Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 339–346. ACM Press, New York (2002)
10. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: SIGGRAPH 1997. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 209–216. ACM Press, New York (1997)
11. Zhang, N., Hong, W., Kaufman, A.: Dual contouring with topology-preserving simplification using enhanced cell representation. In: VIS 2004. Proceedings of the conference on Visualization 2004, pp. 505–512. IEEE Computer Society Press, Los Alamitos (2004)
12. Schaefer, S., Ju, T., Warren, J.: Manifold dual contouring. IEEE Transactions on Visualization and Computer Graphics (2007)
13. Schaefer, S., Ju, T., Warren, J.: Intersection-free contouring on an octree grid. In: Proceedings of Pacific Graphics, IEEE Computer Society Press, Los Alamitos (2006)
14. Heckbert, P.S., Garland, M.: Survey of polygonal surface simplification algorithms. Technical report (1997)
15. Hoppe, H.: Progressive meshes. In: SIGGRAPH 1996. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 99–108. ACM Press, New York (1996)
16. Krishnamurthy, V., Levoy, M.: Fitting smooth surfaces to dense polygon meshes. In: SIGGRAPH 1996. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 313–324. ACM Press, New York (1996)
17. Cohen, J., Olano, M., Manocha, D.: Appearance-preserving simplification. In: SIGGRAPH 1998. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 115–122. ACM Press, New York (1998)
18. Cignoni, P., Montani, C., Scopigno, R., Rocchini, C.: A general method for preserving attribute values on simplified meshes. In: VIS 1998. Proceedings of the conference on Visualization 1998, pp. 59–66. IEEE Computer Society Press, Los Alamitos, CA, USA (1998)
19. Sander, P.V., Snyder, J., Gortler, S.J., Hoppe, H.: Texture mapping progressive meshes. In: SIGGRAPH 2001. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press, New York (2001)
20. Barry, M.: Direct extraction of normal maps from volume data. Technical Report CPSLO-CSC-07-01, California Polytechnic State University (2007)
21. Amidror, I.: Scattered data interpolation methods for electronic imaging systems: a survey. Journal of Electronic Imaging 11, 157–176 (2002)
22. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.: Texture mapping progressive meshes. In: SIGGRAPH 1996. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, p. 43. ACM Press, New York (1996)

Author Index