Ralph Martin Helmut Bez
Malcolm Sabin (Eds.)

Mathematics of Surfaces XI

11th IMA International Conference
Loughborough, UK, September 5-7, 2005
Proceedings

Springer

Volume Editors

Ralph Martin
Cardiff University
School of Computer Science
5 The Parade, Roath, Cardiff, UK, CF24 3AA
E-mail: Ralph.Martin@cs.cf.ac.uk

Helmut Bez
Loughborough University
Department of Computer Science
Loughborough, UK, LE11 3TU
E-mail: h.e.bez@lboro.ac.uk

Malcolm Sabin
Numerical Geometry Ltd
26 Abbey Lane, Cambridge, UK, CB5 9EP
E-mail: malcolm@geometry.demon.co.uk

Library of Congress Control Number: 2005930889

CR Subject Classification (1998): I.3.5, I.3, I.1, G.2, G.1, F.2, I.4

ISSN 0302-9743
ISBN-10 3-540-28225-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28225-9 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11537908 06/3142 5 4 3 2 1 0

Preface

This volume collects the papers accepted for presentation at the 11th IMA Conference on the Mathematics of Surfaces, held at Loughborough University, 5th–7th September 2005. As with all earlier conferences in the series, contributors to this volume come from many countries. The papers presented here reflect the interest in a subject of relevance to mathematics, engineering, and computer science, especially in domains such as computer-aided design, computer vision, and computer graphics.

The papers in the present volume include eight invited papers, as well as a larger number of submitted papers. They cover a range of ideas from underlying theoretical tools to industrial and medical uses of surfaces. The latter category includes such diverse topics as surfaces in car design, and modelling of teeth, while the former includes papers on Voronoi diagrams, linear systems, estimation of curvatures on meshes, operators on meshes, intersection of subdivision surfaces, approximate parameterization, condition numbers, Pythagorean hodographs, artifacts in B-spline surfaces, Bézier surfaces of minimal energy, line subdivision, subdivision surfaces, level sets and symmetry, the topology of algebraic surfaces, curve analysis, interpolation with positivity, and conversion of cyclides to NURBS. Other papers concentrate on particular algorithms arising from applications, such as embedding graphs in manifolds, recovery of 3D shape from shading, finding optimal feedrates for machining, detection of creases in range data, and filling holes in range data.

We would like to thank all those who attended the conference and helped to make it a success. We are particularly grateful to Lucy Nye at the Institute of Mathematics and Its Applications for her hard work in organizing many aspects of the conference, and to Alfred Hofmann and Frank Holzwarth of Springer for their help in publishing this volume. Following this Preface is a list of distinguished researchers who formed the International Programme Committee, and who freely gave their time in helping to assess papers for these proceedings. Due to their work, many of the papers have been considerably improved. Our thanks go to all of them, and to other people who they called upon to help with refereeing.

June 2005

Ralph Martin,
Helmut Bez,
Malcolm Sabin

Organization

The Mathematics of Surfaces XI conference was organized by the Insitute of Mathematics and Its Applications (Catherine Richards House, 16 Nelson St., Southend-on-Sea, Essex, SS1 1EF, UK), and Loughborough University.

Organizing Committee

Helmut Bez	Loughborough University, UK
Ralph Martin	Cardiff University, UK
Malcolm Sabin	Numerical Geometry Ltd., UK

Program Committee

Bob Cripps	University of Birmingham
Gerald Farin	Arizona State University
Peter Giblin	University of Liverpool
Ron Goldman	Rice University
Tim Goodman	University of Dundee
Hans Hagen	University of Kaiserslautern
Edwin Hancock	University of York
Shimin Hu	Tsinghua University
Bert Jüttler	Johannes Kepler University
Myung-Soo Kim	Seoul National University
Tom Lyche	University of Oslo
Nick Patrikalakis	MIT
Jorg Peters	University of Florida
Hong Qin	State University of New York
Martin Rumpf	University of Duisberg
Georg Umlauf	University of Kaiserslautern
Wenping Wang	Hong Kong University
Joe Warren	Rice University
Mike Wilson	University of Leeds
Franz-Erich Wolter	University of Hannover
Guoliang Xu	Chinese Academy of Sciences

Table of Contents

Free-Form Surface Construction in a Commercial CAD/CAM System

Florian Albat and Rainer Müller

Tebis AG, Lademannbogen 128, D-22339 Hamburg, Germany
{Florian.Albat, Rainer.Mueller}@tebis.com

Abstract. In automobile industry, free-form surfaces often have to be constructed and even more often have to be modified. Frequently, a surface model is given as a triangular mesh, which is converted to (polynomial) spline surfaces (reverse engineering). We show some arising problems and how they are solved in our software. Furthermore, we present some open theoretical problems.

1 Surface Construction in CAD

There are two main types of surfaces in CAD: free-form surfaces and standard surfaces. The latter describe spatial objects, that can be defined uniquely by quite simple rules, often in terms of two-dimensional drawings. Typically, such objects consist of planes, cylinders and similarly simple surfaces. The sharp edges between the single surfaces are rounded by simulating a rolling ball (fillet surfaces). These models dominate in machine-building. The hood or roof of a car, however, is shaped in such a way, that it cannot be defined by easy rules like "cylinder of height 10cm with radius 3cm". Such surfaces are called free-form surfaces.

The automobile industry is not the only, but a very important field of industrial application for free-form surfaces. Many of our customers belong to this industry, both the car companies themselves and many suppliers of different kind. It is the main application field for surfaces of the highest quality requirements ('Class A', this implies e.g. curvature continuity). They are needed for the visible outer skin, that determines the potential buyer's impression of a car. But these high-quality surfaces represent only a minority of all constructed surfaces in a car, most of which lie invisible in the inside like the one shown in Fig. 1 and do not need to be of such a high quality. However, even such surfaces need to be G^1 continuous (within a tolerance of about $0.5°$), because they will finally be produced as sheet metals, and sheet metals cannot be shaped into sharp edges.

Besides sheet metals a car contains metal parts like the crank shaft, exhaust pipe and clutch shell plus plastic parts like the dashboard, which are molded. For all these parts CAD models must be constructed.

New objects account for only about 10% of construction work, roughly 90% of all constructions are modifications of existing models. So we will focus on such modifications. One reason for them is, that first versions of the fabrication

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 1–13, 2005.

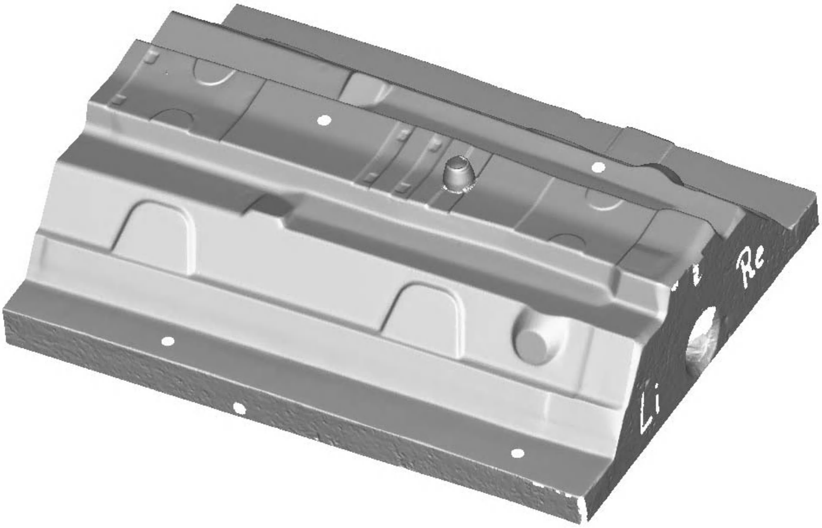


Fig. 1. Technical sheet metal from the inside of a car (scanned data)

tools are milled before the final design with all details is done. Another frequent reason is the compensation of a metal's springback.

Modifications can be either *real* or *virtual*. As real modifications we denote such, that are based on a really existing object. For example, the outer skin of a car is usually milled as a prototype, and the shape of this real model is modified by sanding off and spreading of material to satisfy certain aesthetic visions. This modified model is scanned, the scan points are triangulated and from this triangular mesh a new CAD model consisting of spline surfaces is constructed (reverse engineering). Virtual modifications we call such, that are performed directly on the CAD model without any real object being modified and scanned. A typical requirement is to raise a certain position of the roof for 1 cm. To do this, mostly neighbouring surfaces need to be adapted.

We will treat virtual modifications in the third section. Beforehand, we will elaborately treat real modifications, i.e. free-form surface construction by reverse engineering based on a triangular mesh. The question arises whether it is at all necessary to convert the mesh of scanned data into polynomial surfaces, or whether suitable new algorithms make it possible to work with meshes just as effectively. This question is interesting from a theoretical standpoint and not easily answered. However, it is (for the time being) irrelevant from a practical point of view, since industrial process chains are designed for work with 'smooth' surfaces. Changing to a continuous technique using meshes will not be possible until not only solutions for single steps exist, but really all design and production processes can be converted without suffering a loss in quality. Even then, the change will be done only when the savings gained outweigh the high cost of conversion. As a result, at least for the present and in the foreseeable future, it will be unavoidable in industrial applications to convert mesh data into polynomial surfaces. Hence, there is a need for easily useable software to do reverse engineering.

Among others, this paper shall also show, that our point of view as a provider of commercial software differs in some points from that of more theoretically interested scientists. We have to sell the used algorithms in such a way, that our customers, of whom hardly any has studied mathematics or computer science, can use them as easily as possible. Some of the discussed points may be trivial for academically educated readers, but these are questions, which must be considered if one wants to have success in the market.

This paper does not want to give an overview of the vast literature about possible algorithms or similar, but is aimed to inform about practical experiences of different software users, which is not written in common scientific papers. Hence, we restrict our list of references to [1], which is a good short introduction to reverse engineering. More references can be found therein.

2 Surface Construction with Reverse Engineering

2.1 Structuring the Object

Fig. 2 shows a triangular mesh of the front part of a car. Because of the symmetry, it suffices to construct the left half of the car. A surface construction of this model is shown in Fig. 3. This model looks good, matches the mesh very well and is G^1 continuous, which are very important properties of a surface model from reverse engineering. This model was constructed with our software package RSC (*Rapid Surface Creation*).

In Fig. 3 the border lines of the single surfaces are drawn. We denote their entirety as *wire frame*. Many curves follow the object's *feature lines*, each of which separates different 'uniform' areas, e.g. the relatively flat part of the hood from the more curved fillet. As we will delve into later, it is very important for the quality of the approximation surfaces, that the feature lines can be found in the wire frame. In a classical construction without mesh, the feature lines are naturally incorporated in the process: the larger uniform surfaces, the so called main surfaces, are constructed first. Then they are intersected with each other, and the sharp edges are rounded by fillets. The border lines of fillets are feature lines.

When constructing the fillets, the main surfaces are trimmed by the fillet boundaries. Likewise, the wire frame in Fig. 3, which originated from reverse engineering, contains many trimmed surfaces, i.e. surfaces, whose border is not a quadrangle of isoparametric lines. This is the normal case, one can hardly find CAD constructions, which do not contain trimmed surfaces. Exceptions are constructions, that are made by some reverse engineering programs, who cannot handle trimmed surfaces, but only *natural surfaces*, i.e. quadrangular surfaces, whose boundaries are isoparametric lines. Of course, every wire frame can be subdivided into quadrangles, but such a face layout is not suitable for some subsequent process steps. As an example, fig. 4 shows on the left a part of the

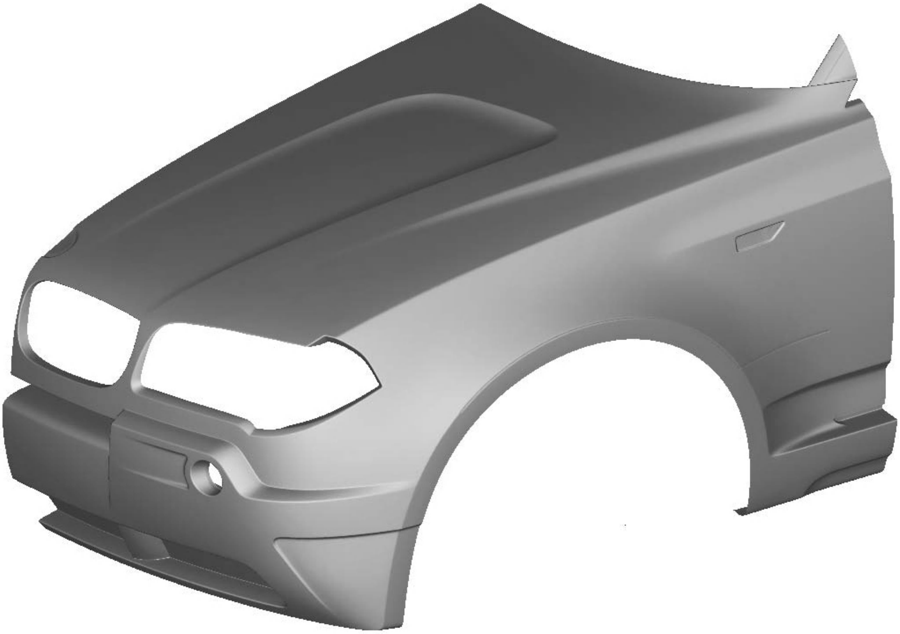


Fig. 2. Triangular mesh of a car

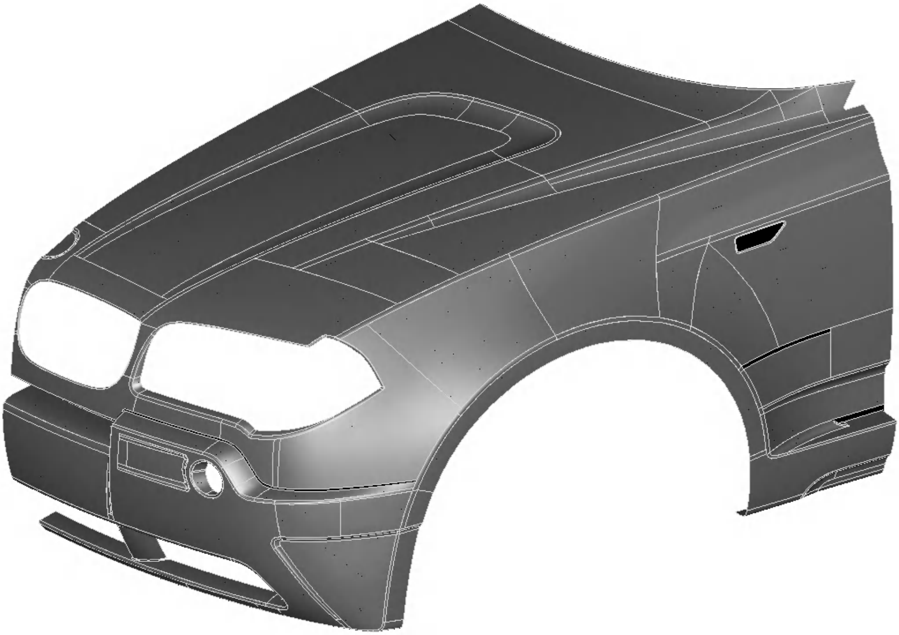


Fig. 3. Surface construction from the mesh in Fig. 2

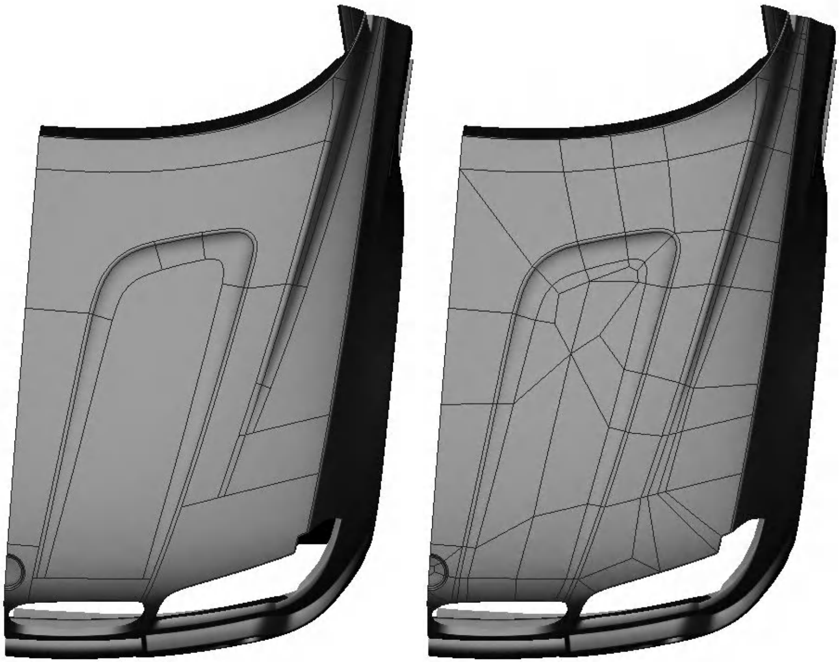


Fig. 4. Part of the wire frame from Fig. 3(left) and example for quadrangulation (right)

wire frame from fig. 3 and on the right a subdivision of this wire frame, so that it does only contain quadrangles.

The most obvious disadvantage of quadrangulation is the ‘unnaturalness’: the human user is disturbed by the many additional border lines. Another disadvantage is, that it is difficult to enlarge a main surface, when it was divided into several quadrangles. Such an enlargement is e.g. necessary, when a neighbouring approximated fillet surface has to be replaced by an exact fillet surface with bigger or smaller radius (reduction of radius).

Especially for the visible surfaces of a car (roof etc.) a surface calculated automatically by approximation of digitized data is hardly ever good enough for the high quality standards. Here is always manual postprocessing necessary, e.g. by manipulation of Bézier points. For that, a subdivision of the surfaces into quadrangles is completely unsuitable, one needs a face layout like in Fig. 3.

The designer’s great dream is of course an algorithm, that is given the mesh and creates a wire frame like that in Fig. 3. We are working on this and are optimistic, that there will be considerable progress soon, but until now, such an algorithm does not exist. As a surface model implies a wire frame, no automatic procedure exists, that constructs a surface model like the one shown directly from the mesh. Sure, there are commercial programs for this task, but the programs known to us can only handle quadrangles and therefore have the mentioned disadvantages.

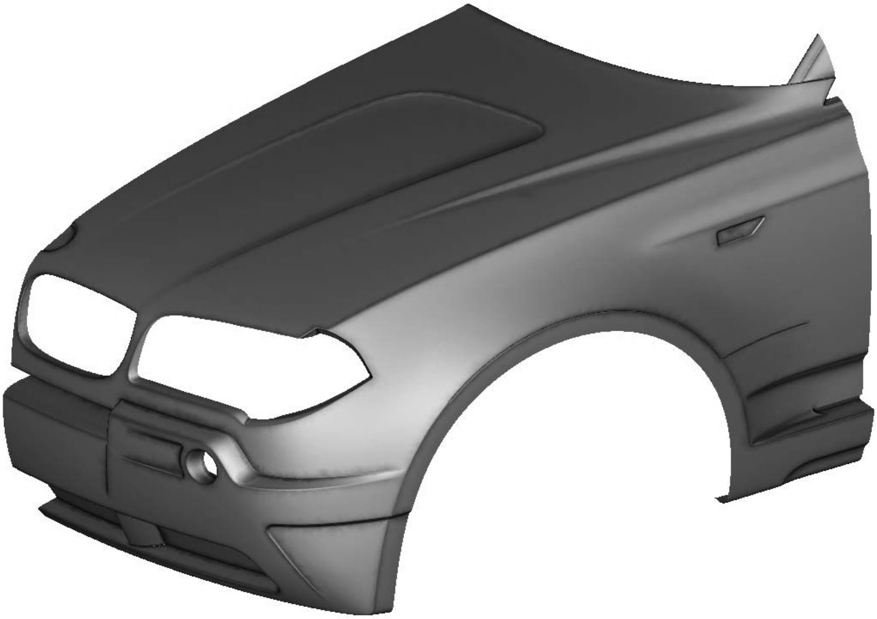


Fig. 5. Curvature representation of the mesh in Fig. 2

Until now, no program can create a wire frame nearly as well as the human designer, so the most reliable method, which is also implemented in our software, is to let the user construct the wire frame by himself. In general, it is difficult to construct spatial free-form curves, but in this application, it is much easier, as the curves are not absolutely free, but must lie on the mesh. By projecting the input automatically, our software enables the user to create curves directly on the mesh. This has proven to be both intuitive and reliable.

As support, we offer half-automatic functions for construction of feature lines, which e.g. recognize pair of fillet border curves or curves with constant (approximated) mesh curvature on positions selected by the user. For visualization of the object's structure it has proven useful to approximate the mesh's curvature and show it as a color spectrum, see Fig. 5.

2.2 Calculation of Surfaces

By construction of the wire frame the user decomposes the mesh into several facets, in which then surfaces can be calculated.

The surface, which is calculated for a facet, depends on the mesh and the facet's boundary curves, that define the approximated region. To achieve G^1 continuity, it also depends on surfaces in adjacent facets, if such exist. Letting the user specify all these parameters explicitly, is not only cumbersome, but also prone to error. A user-friendly program should ease this as far as possible. To do so, we use in our software a data structure 'RSC manifold', which contains besides geometric data also topological data, namely neighbourhood

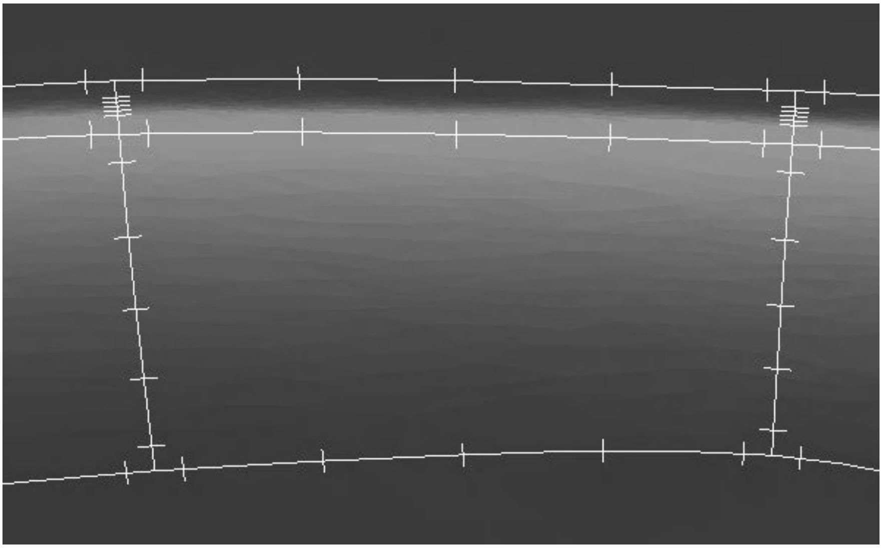


Fig. 6. Representation of a facet in form of stings

information. During the construction of the wire frame, for every new curve is checked, whether it is connected to a existing curve, and if so, how the curves are sorted around their common endpoint. From this is known later on, which curves together bound a facet, so that the user can work directly with facets. When he selects a facet, all aforementioned parameters are known to the program. To make a facet, in which no surface has been calculated yet, selectable in the graphic user interface, we draw around the facet's boundary stings towards the inner region (see Fig. 6). To perform a surface calculation, the user simply chooses an appropriate function and selects one or more facets.

There are different methods to calculate an approximation surface in a facet, which result in different types of surfaces. 'Classical' methods of reverse engineering are the calculation of optimally fitting planes, cylinders, cones or spheres, which we also offer. Here, we are particularly interested in free-form surfaces. As we have seen above, they can be divided into trimmed and natural surfaces.

Natural surfaces are relative simple, and their great advantage is, that they can be relatively easy connected G^1 continuous to their neighbouring surfaces. Connecting a trimmed surface to neighbours, is considerably more difficult (if the trim curve is not an isoparametic line). Hence, we renounce to match trimmed surfaces G^1 continuous, but if adjacent natural surfaces exist, we adapt them to a newly calculated trimmed surface. We do not allow two trimmed surfaces to be adjacent to each other. This is a restriction for the user, but as it is conform to the standard construction process, this restriction is mostly accepted as natural and not very objectionable.

Important for approximation with trimmed surfaces is the surface's shape outside the trimmed region, where no data points are available to approximate.

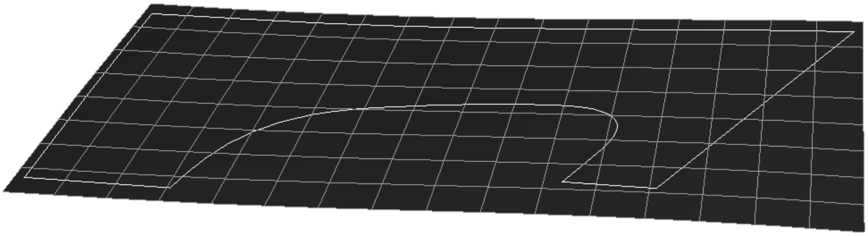


Fig. 7. Trimmed approximating surface with extrapolating region

Some approximation techniques result in surfaces with strong oscillations in this extrapolating region. Such surfaces are unsuitable, if one is interested in enlargement of a main surface or radius reduction (see above). An example for good shape in the extrapolating region is shown in Fig. 7 from the hood of the car in Fig. 3. The brighter closed curves are the facet's boundary, outside of which the rest of the calculated surface is shown. To illustrate the surface, a grid of isoparametric lines is drawn. One can see, that the surface looks smooth everywhere, not only in the approximated area. In the front on the right one can see, that the surface is slightly bent downwards, but this is absolutely acceptable.

The wire frame has great influence on the quality of the approximation surfaces. As an example, Fig. 8 shows the same part of two different surface models of a mesh. The wire frame of the upper model was simply created by projection of a regular grid, which is the fastest method to construct a wire frame. The wire frame of the lower model is based on the mesh's feature lines. The increase of quality is obvious.

2.3 Interactive Modifications of a Surface Model

When calculating surfaces, it sometimes turns out, that the surface quality is not as good as expected, because the wire frame is not good enough in this region. Then functions for interactive modeling are useful, so that the user does not have to construct the curves completely new, but can e.g. modify a single curve or move a common endpoint across the mesh. The internal knowledge of neighbourhood information is needed, so that common endpoints remain common and surfaces in the affected facets can be recalculated automatically.

Especially useful is the interactive remodeling of the wire frame, when an earlier CAD model is at hand, whose curves can mainly be used. This is often the case, as many of the frequent changes in the model are only local and do not change the global structure. Then no complete wire frame needs to be constructed, but the user must only modify the regions, where the previous model does no longer match the changed real object.

The sheet metals of a car are produced by letting a stamp like that in fig. 9 press a flat sheet metal into a matching counterpart (essentially a offset surface of the stamp). Unfortunately, the sheet metal does not exactly remain in the stamp's shape, but it springs a little back towards its former planar shape.

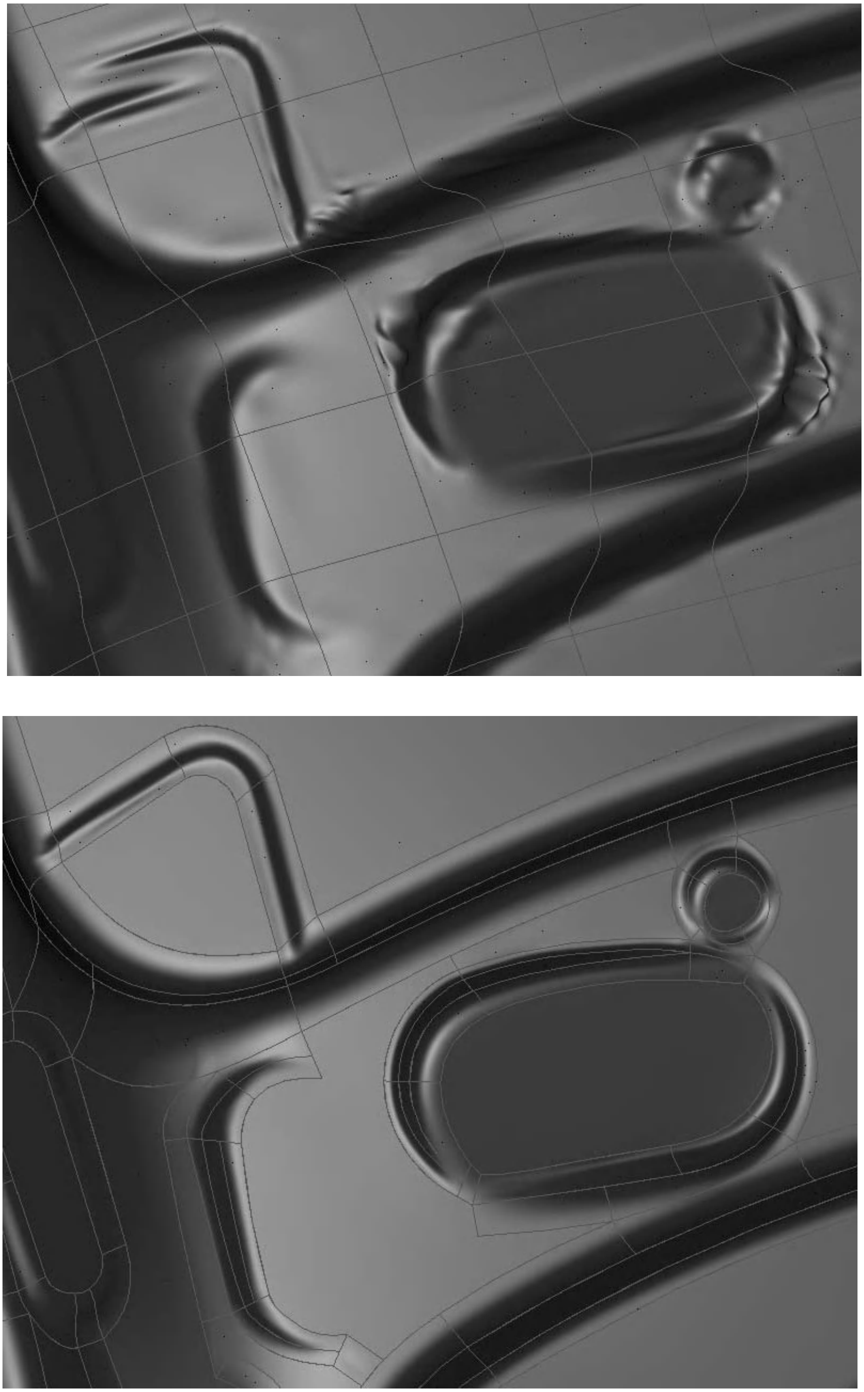


Fig. 8. Surface models with different wire frames

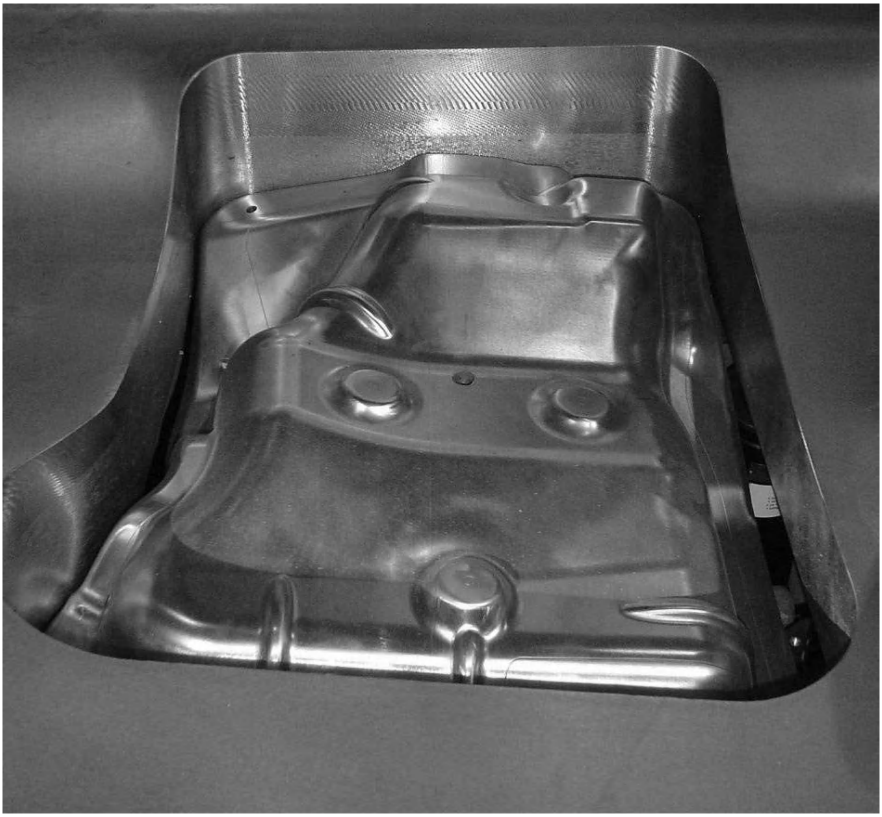


Fig. 9. Stamp for production of sheet metals

Therefore, the stamp must not have the shape, which the metal is supposed to get, but it must be shaped in such a way, that the metal after springing back has the desired shape. This compensation of springing back, known as bending, is a big challenge in CAD, which is still not completely solved. It can be simulated more or less well, and then it can be seen as a modification of the desired surface model. When the bending is simulated by finite elements, one gets the necessary stamp form as a FEM model, i.e. as an triangular mesh. For further treatment, it is not important, whether the mesh originated from scanned data or from a FEM model.

3 Virtual Modifications

Many modifications, until a car design is finished, originate from technical requirements, for which e.g. the middle part of the roof must be raised by one centimeter. Of course, the look of the car shall be changed as little as possible, so the roof must be deformed uniformly. The neighboring parts (door, pillar, . . .)

must be modified accordingly. One reason for the modification can be, that the assembling of test metals has shown large gaps.

For a real modification, the model-maker raises the roof by spreading clay or some other material, and he achieves a 'uniform deformation' by not spreading a single lump at the reference position but a whole layer on the roof and perhaps the sideways parts, which becomes thinner towards its boundary. For a virtual modification, this must somehow be simulated. It must be defined, how far the layer of clay reaches, i.e. what is the domain of the deformation, and how the layer becomes thinner towards the boundary.

Up to now, we do only offer first approaches for such problems, which are nevertheless very useful, according to our customer's feedback. We will treat this much more elaborately in the future. Questions are not only, how the surfaces are calculated, but also how the user is enabled to specify his requests as easy as possible to the software.

4 Open Problems

4.1 Recognition of Structure

The user's substantial work in reverse engineering is the definition of the face layout, which in our software is done by the construction of the wire frame. Hence, here is the biggest potential to reduce the amount of work. The aim is an algorithm, that reliably produces curves, as they can be seen in fig. 3. During the preparation of this paper, we have made considerable progress in this area, and we hope to present this soon.

4.2 Modeling of Curvature

An extremely time-consuming part in the automobile construction process is the 'polishing' of a car's visible surfaces to meet highest aethetic criteria ('Class A surfaces'). This is done by interactive modeling of Bézier points with special respect to the surface's curvature. While the connection between the motion of a control point and the change of the surface's shape is quite intuitively predictable, the effect on the curvature in contrast is only hard to comprehend. The results of this technique are often not as good as desired.

Hence, we are looking for possibilities to let the user manipulate the curvature as intuitively as possible. Besides something like 'curvature Bézier points' it would also be helpful, if one could impose constraints to a surface like that the curvature of isoparametric lines has to be monotonic or have no zeros.

A special case for curvature constraints is the construction of special blend surfaces, which in opposite to the standard rolling ball fillet are connected G^2 continuous to the blended surfaces and whose lateral isoparametric lines shall have exactly one maximum of curvature, where the curvature shall furthermore have a prescribed value.

4.3 Surface Fairing

The abovementioned demands on a surface's curvature have to be specified explicitly by the designer. But of course, the surfaces, which are calculated without special effort to modeling or configuration shall also look as good as possible. Improvement of surface quality is generally denoted as surface fairing. The question in the beginning is: when is a surface 'fair'? The most common and most easy to manage approach is to define fairness by a functional of the surface's coefficients. Then, the approximating surface does not minimize a pure sum of squared distances but a weighted sum of squared distances and the fairing functional. If the fairing functional is quadratic, the calculation of the optimal surface requires only the solution of a linear equation system, exactly like the pure approximation without fairing. Many fairing functionals have been proposed, and it is not clear, which is the most suitable.

For extrapolation of a trimmed surface, it must be fair not only in the approximated area, but also in the outside. This cannot be guaranteed with the approach of fairing functionals, at least not with the functionals we have tested. So we needed to use a different fairing technique, which has certain disadvantages and is surely not optimal.

4.4 G^1 Continuity

Connecting two natural surfaces G^1 continuous along a common boundary, is relatively easy. Connecting a natural surface to a trimmed surface is more difficult, but feasible within the tolerance needed for our applications, which is less than 0.005 mm distance and less than 0.5° angle. Connecting two trimmed surfaces to each other, when the common boundary is for neither of them an isoparametric line, is much more difficult. One possibility is to provide a high number of degrees of freedom by increasing the number of segments, but high segment numbers have to be avoided especially for the trimmed main surfaces. As we do not know a satisfactory method, we impose the abovementioned demand, that the designer is not allowed to construct two adjacent trimmed surfaces. This is not a strong restriction, but of course it would be better, if we could abstain from it.

We want to emphasize, that G^1 continuity is important, not C^1 continuity. The latter is relatively restrictive and hardly realizable for trimmed surfaces.

5 Conclusion

Important for free-form surface construction by reverse engineering or by free construction and modification is:

- use of spline surfaces,
- structuring of the object,
- use of trimmed surfaces and
- use of neighbourhood information.

Our approach for reverse engineering is letting the designer construct a wire frame, whose facets can be approximated by different types of surfaces (trimmed surfaces, natural surfaces with G^1 continuous connection, planes, ...). For the ease of handling we use automatic recognition and use of topological information.

Our software is successfully sold and used, but of course it is not so perfect, that we do not want to improve it. Many points are rather technical details, but some points are of general, also theoretical interest. For our purposes, the most important of them are:

- Modeling of a surface's curvature,
- recognition of feature lines on a triangular mesh.

We hope to stimulate the interest of scientists, who are interested in a real practical application of their results.

Reference

1. Varady, T., Martin, R.: Reverse Engineering, Chapter 26 in Kim, H.-S., Hoschek, J., Farin, G. (eds.): Handbook of Computer Aided Geometric Design, Amsterdam, Elsevier, 2002.

Polyhedral Gauss Maps and Curvature Characterisation of Triangle Meshes

Lyuba Alboul and Gilberto Echeverria

Sheffield Hallam University, Sheffield S1 1WB, UK
{L.Alboul, E.Echeverria}@shu.ac.uk
http://www.shu.ac.uk/scis/artificial_intelligence

Abstract. We design a set of algorithms to construct and visualise unambiguous Gauss maps for a large class of triangulated polyhedral surfaces, including surfaces of non-convex objects and even non-manifold surfaces. The resulting Gauss map describes the surface by distinguishing its domains of positive and negative curvature, referred to as *curvature domains*. These domains are often only implicitly present in a polyhedral surface and cannot be revealed by means of the *angle deficit*. We call the collection of curvature domains of a surface the *Gauss map signature*. Using the concept of the Gauss map signature, we highlight why the angle deficit is sufficient neither to estimate the Gaussian curvature of the underlying smooth surface nor to capture the curvature information of a polyhedral surface. The Gauss map signature provides shape recognition and curvature characterisation of a triangle mesh and can be used further for optimising the mesh or for developing subdivision schemes.

1 Introduction

In many applications a physical object is represented by discrete data, obtained by some measurement system. A polyhedral model (i.e. a triangular mesh, a piecewise linear surface) is an easily obtainable preliminary sketch of the given object. Triangular or polygonal meshes are commonly used in modern computer-related applications to represent surfaces in three-dimensional space. Therefore, there is a substantial need for accurate estimates of geometric attributes that are directly computed from a mesh, such as surface area, normal vectors, and curvatures. In recent years significant efforts have been made to define the analogues of differential geometry concepts on meshes that imitate those of a smooth surface [1, 5, 8, 13, 14]. Among those concepts surface curvatures are particularly important, as they are basic measures to describe the local shape of a smooth surface. However, the surface of a triangle mesh is not smooth, and there is still no consensus about the most appropriate way of estimating such geometric quantities as curvatures. On the other hand, methods are being developed to capture curvature information without referring to higher-order formulas of differential geometry. These methods are based on the discrete curvature concepts and are of growing interest for geometric modelling [13, 2, 8, 14]. Discrete curvatures can

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 14–33, 2005.

be computed directly on triangle meshes. The principal difference between a polyhedral and a smooth surface is that the discrete curvatures in a polyhedral surface are concentrated around the vertices and along the edges. If we think of the surface of a triangle mesh as an approximation of a smooth surface, then, informally speaking, curvatures in the domain of the underlying smooth surface are ‘glued’ together in the corresponding domain of the triangle mesh.

Therefore, measures of curvature in a piecewise linear setting should be analogues of *integral formulae* for curvature in a ‘smooth’ setting and should preserve integral relations for curvature, such as the Gauss-Bonnet theorem [6, 3]. Such analogues exist and were introduced long ago in relation to the theory of non-regular surfaces (see an overview in [1]). These analogues were discussed in detail in [6], where the authors also compare discrete curvatures with their smooth counterparts.

In the last five or six years the number of papers that explore discrete curvatures in one or another context has increased significantly. Much attention is paid to the discrete Gaussian curvature, known also as the *angle deficit*. It has also been referred to as *angular defect*. The concept was brought to the attention of the geometric modelling community in 1984 [7], where the author listed several applications of the angle deficit in surface modelling, mostly in the the context of the mechanics of thin-shell structures. Nowadays, the angle deficit is used to evaluate curvature information directly from a mesh, as well as to estimate the Gauss curvature and derive principal curvatures of the underlying smooth surface, assuming that the mesh samples the surface in a certain way [15, 14]. In [5] the problem of the correct estimation of the Gauss curvature is investigated in detail, and they argue, on the basis of several approximation results, that approaches based on the use of normalized angular deficits are often erroneous, and can be applied correctly only if the geometry of meshes is precisely controlled. We agree with them, and in this paper we highlight why the angular deficit is sufficient neither to estimate the Gaussian curvature of the underlying smooth surface nor to capture the curvature information of a polyhedral surface. Loosely speaking, the reason is that there are more curvatures for polyhedral surfaces than for smooth ones [6, 1, 2]. This fact is still not fully acknowledged, but without addressing it, it is impossible to develop correct curvature estimates.

Besides the need to derive correct curvature estimates directly from polygonal meshes, there is also a need for visualisation of the geometry of an object in order to explore complex shapes and emphasize hidden details. We propose an approach that addresses both needs, and that enables us to correctly and consistently describe and visualise complex 3D shapes based on curvature properties. In the smooth case, the *Gauss map* and related *shape operator* completely determine the shape of the original surface S [10]. Therefore, efforts have been made to use analogues of the Gauss map to explore shape characteristics of a complex polyhedral surfaces. For example, an analogue such as the *extended Gaussian image* is used in Computer Graphics and Vision to compare objects and to illuminate the structure of surface shape [11, 12]. However, the extended

Gaussian image and its generalisations construct only normals to the faces of the polyhedral surface without indicating their connectivity. There exist few attempts to create the Gauss map directly from the mesh, but the results are still scarce and ambiguous for non-convex objects [12]. In [1, 13, 9] some theoretical considerations regarding Gauss maps for triangulations are given, but only very schematically, and no algorithm has yet been presented.

Our method constructs a polyhedral analogue of the Gauss map directly from a polygonal mesh and uses this map to characterise surface shape. We believe that such an algorithm is developed here for the first time. We also give a definition of the polyhedral Gauss map and show its conformity with the smooth case. Our approach enables us to construct unambiguous Gauss maps for polyhedral surfaces of complex shape, of various genera, self-intersecting and even non-manifold surfaces. The algorithm is relatively straightforward for simple embedded vertices, but a sound computation of more complex vertices requires thorough analysis of the geometry of a vertex as well as ingenious computational techniques. Nevertheless, construction of the Gauss map of a polyhedral surface is a computationally low-cost method, as all operations are linear. The resulting Gauss map provides a description of the surface by determining its domains with respect to incorporated curvatures. These domains are often only implicitly present in a polyhedral surface, and cannot be determined by the sign of the angle deficit only. Each domain can be split up into uniquely determined subdomains; therefore each surface can be associated with the collection of these sub-domains, denoted as the *Gauss map signature*, abbreviated as GMS. The GMS method besides shape recognition and description can be used for optimisation of the underlying model or for developing subdivision schemes. The method provides also a better insight into the geometric structure of complex triangle meshes, by describing various vertex types, some of them with a very complex GMS. A good understanding of the geometry of meshes is a step towards more robust mesh manipulation algorithms. Finally, the proposed GMS is computationally inexpensive, can be viewed dynamically, and is effective in visualising curvature features of complex polyhedral surfaces.

In the rest of the paper, Section 2 recalls the necessary background on polyhedral and global differential geometry regarding curvatures. Section 3 defines the polyhedral Gauss map, lists some of its properties and describes the algorithm to construct it. Section 4 presents Gauss map visualisations of various single vertices and complex polyhedral surfaces. Section 5 concludes the paper.

2 Basic Concepts and Definitions

2.1 Polyhedral Surfaces

By a polyhedral surface we understand a triangulated polyhedral surface. Designating $\mathbf{V}$ as a finite point set in three-dimensional space, $\mathbf{V} = \{V_i, i = 1, \ldots, n\}$, we denote by $\mathbf{P}(\mathbf{V})$ a polyhedral surface with the vertex set $\mathbf{V}$. The term *polyhedron* refers to a closed polyhedral surface. In such a setting a polyhedron is bounded, but might be non-simple, i.e. non-homeomorphic to a sphere. Also it

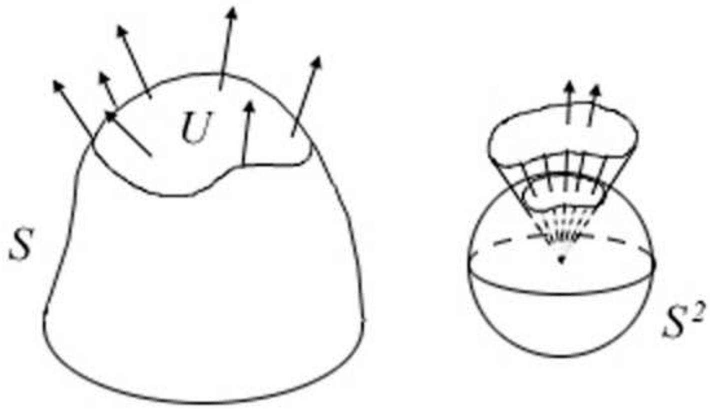


Fig. 1. Gauss map of a smooth surface

might be multi-connected and self-intersecting, and its interior volume is not necessarily part of the polyhedron. Therefore, a polyhedron is not necessarily a solid body. Given a polyhedron $\mathbf{P}(\mathbf{V})$, the set of its vertices is denoted by V, the edges by E, and the faces by F.

Definition 1. *The star* $\mathbf{Str}(\nu)$ *of a vertex* ν *is the union of all the faces and edges that contain the vertex, and the link* $\mathbf{Lnk}(\nu)$ *of the star (the boundary of the star) is the union of all those edges of the faces of the star* $\mathbf{Str}(\nu)$ *that are not incident to* ν.

A surface of a triangle mesh is an example of a polyhedral surface. Therefore, all properties of a polyhedral surface discussed below are applicable to a mesh.

2.2 Integral Curvature Relations: Smooth Towards Discrete

In this paper we are interested only in discrete curvatures related to the integral Gaussian curvature, i.e. those that are supported on the vertices. In what follows we give a brief comparative analysis between the known integral relations for curvature for smooth surfaces and their discrete counterparts.

Integral Gaussian Curvature and the Angle Deficit. For a domain U of smooth surface S the Gauss map $N(U)$ is the map assigning to each point $p \in U$ the point on the unit 2-sphere $S^2 \in \mathbf{R}^3$, by ‘translating’ the unit normal vector $N(p)$ to the origin [10]. The end-points of normals, therefore, will cover a certain region on S^2 (see Figure 1).

Given a neighbourhood $U(p)$ on S, the ratio of the area $N(U(p))$ to the area of $U(p)$ can be considered as a measure of the amount of curvature of the surface S near the point p. Then the Gaussian curvature $K(p)$ is defined by setting

$$|K(p)| = \lim_{U(p)\downarrow p} \frac{\text{Area}(N(U(p)))}{\text{Area}(U(p))} \tag{1}$$

where the limit is taken as the neighbourhood $U(p)$ contracts down to the point p.

If a neigbourhood $U(p)$ is sufficiently small such that the map $N(U(p))$ is one-to-one and orientation-preserving (outward normals at corresponding points on S and S^2 correspond), then the area $N(U(p))$ is considered positive, and the corresponding region $U(p)$ is said to be strictly *convex* and $K(p) > 0$. If the map $N(U(p))$ is one-to-one but orientation reversing, then the area $N(U(p))$ is considered to be negative, p is a saddle point and $K(p) < 0$. Of course, different regions of S can be mapped to the same region on the unit sphere, which results in multiplicities of the Gauss map.

Therefore for a region $U(p)$, for which the map $N(U(p))$ might not be one-to-one, under the integral Gaussian curvature one understands the algebraic area of the image $U(p)$ under the Gauss mapping:

$$K_{int} = \int_U K dA. \tag{2}$$

For an entire closed smooth surface the previous formula turns into the mathematical expression of the Gauss-Bonnet theorem:

$$\int_S K dA = 2\pi\chi(S), \tag{3}$$

where $\chi(S)$ is the Euler characteristic of S. The discrete analogue of the expression 2 is known as the *angle deficit*, and it measures curvature ω around vertex ν:

$$\omega = 2\pi - \theta, \tag{4}$$

where $\theta = \sum \alpha_i$ is the total angle around vertex ν, and α_i are those angles of the faces in the $\mathbf{Str(\nu)}$ that are incident to ν. This is a polygonal analogue of the Gauss-Bonnet theorem [3]. We refer to ω as the *discrete Gaussian curvature.* For any point $x \in \mathbf{P}(\mathbf{V})$, except vertices, ω is identically equal to zero. For a domain $U \subseteq \mathbf{P}(\mathbf{V})$ the total curvature Ω_U is determined as

$$\Omega_U = \sum_{\nu \in U} \omega_\nu. \tag{5}$$

For an oriented closed polyhedral surface $\mathbf{P}(\mathbf{V})$ of genus g the total curvature $\Omega_{P(V)}$ is equal to $(1 - g)4\pi$, so the analogue of the integral relation for the Gaussian curvature is preserved [1, 4, 16].

Integral Absolute Curvature and its Discrete Analogue. The following measure which we determine is an analogue of the integral absolute curvature for a polyhedral domain. The most obvious candidate for this measure seems to be the sum of absolute values of the angle deficits around the vertices in the domain.

However, in Figure 2 we can see that in both polyhedra all curvatures ω_{ν_i} are positive. In the depicted polyhedra they are actually equal for every corresponding vertex, i.e. $\forall\ i\ \omega_{\nu_i \in P_1} = \omega_{\nu_i \in P_2}$. Therefore, we have:

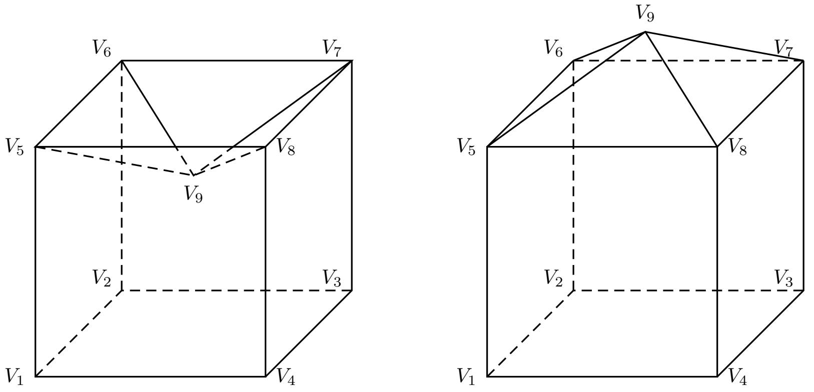


Fig. 2. Two polyhedra

$$\Omega(P_1) = \Omega(P_2) = \sum_{\nu \in P_1} |\omega_\nu| = \sum_{\nu \in P_2} |\omega_\nu| = 4\pi. \tag{6}$$

The left polyhedron is non-convex, but the above equation does not reflect this fact. For a closed non-convex smooth surface S the total absolute curvature $K_{abs} = \int_S |K| dA$ is greater than 4π; therefore, $\sum_{\nu \in P} |\omega_\nu|$ is not an appropriate analogue of K_{abs}. The problem is that the curvature ω around a vertex may consist of positive and negative 'parts' that are 'glued' together; and the task is to separate them. If vertex ν belongs to the boundary of the convex hull of its star (i.e. the convex hull of ν and all vertices in its star), then we can single out another star $\mathbf{Str}^+(\nu)$ with ν as the vertex and those edges of $\mathbf{Str}(\nu)$ that belong to the boundary of the convex hull. The edges of $\mathbf{Str}^+(\nu)$ will determine the faces of $\mathbf{Str}^+(\nu)$. We refer to $\mathbf{Str}^+(\nu)$ as the *convex cone* of vertex ν. Then the *positive (extrinsic) curvature* ω^+ is defined as

$$\omega^+ = 2\pi - \theta^+ \tag{7}$$

where θ^+ is the total angle around ν in $\mathbf{Str}^+(\nu)$. The positive curvature ω^+ is equal to zero, if the vertex ν and all the vertices in $\mathbf{Lnk(\nu)}$ lie in the same plane. If the convex cone around ν doesn't exist, i.e. ν lies inside the convex hull of $\mathbf{Str}(\nu)$, then ω^+ is, by definition, equal to zero.

This definition of ω^+ is also in conformity with the smooth case. Indeed, suppose that $\mathbf{SUP}_U(N)$ is the number of local supporting planes to the region U of a surface S, then the integral positive curvature $\int_U K^+ dA$ can be defined as the integral $\int \mathbf{SUP}_U(N) d\sigma_N$ taken over the unit 2-sphere $S^2 \in \mathbf{R}^3$.

The *negative (extrinsic) curvature* ω^- of ω is determined as

$$\omega^- = \omega^+ - \omega \tag{8}$$

The *absolute (extrinsic) curvature* ω_{abs} is defined as

$$\omega_{abs} = \omega^+ + \omega^- \tag{9}$$

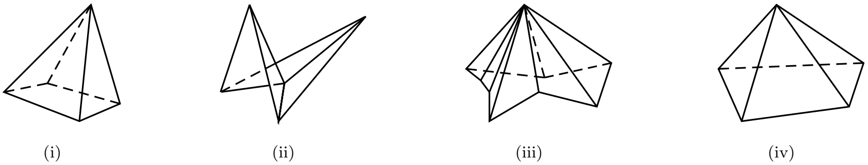


Fig. 3. Examples of vertices: convex (i), saddle (ii), and mixed (iii) with its convex cone (iv)

Three basic types of vertices for an embedded polyhedral surface are then distinguished ([6, 1]): *convex* ($\omega^+ = \omega$), *saddle* ($\omega^- = -\omega$) and *mixed* ($\omega^+ > 0$, $\omega^+ \neq \omega$) (see Figure 3).

The *total absolute extrinsic curvature* Ω_{abs} is defined then as the sum of absolute extrinsic curvatures of all the vertices of a polyhedral surface P:

$$\Omega_{abs} = \sum_i \omega_{abs}(\nu_i) = \sum \left[\omega^+(\nu_i) + \omega^-(\nu_i)\right] \qquad (10)$$

The word ‘extrinsic’ in the introduced curvatures is due to the fact that they reflect how a vertex is embedded in space, while the angle deficit, computed only by using the angles around a vertex remains an intrinsic measure. The total absolute extrinsic curvature Ω_{abs} takes different values on the polyhedra that are depicted in Figure 2. It is equal to 4π on the right polyhedron, as it represents a convex body, and is greater than 4π on the left polyhedron, which is not convex. Therefore, the total absolute extrinsic curvature of a polyhedral surface is an adequate analogue of total absolute curvature of a smooth surface.

Remark 1. The curvature measures can also be expressed in terms of the number of critical points. Details can be found in [3, 6, 1].

3 The Polyhedral Gauss Map and Its Construction

Separation of the positive and negative parts of the curvature for a mixed vertex can also be carried out using an analogue of the Gauss map for a polyhedral surface, which we call the *polyhedral Gauss map.* For a polyhedral surface curvatures are concentrated around vertices, so we need to be able to construct the Gauss map for a vertex, and the union of the Gauss maps for all vertices is the Gauss map of a polyhedral surface. Assuming that the surface is oriented, we can construct an outward unit normal to any point of a face except at vertices and edges, and all these normals are parallel to each other. By translating them to the same origin, we get a unique unit vector. Applying the same procedure to each face of $\mathbf{Str}(\nu)$ we get a bundle of unit vectors. The end-points of these vectors lie on the unit sphere. Without loss of generality, we assume that no two neighbouring faces lie in the same plane, then each end-point corresponds to a face and can be considered as the image of the face. By analogy with the smooth case (see Subsection 2.2), we can define the Gauss image at a vertex as follows

Definition 2. *The Gauss image of a vertex is defined as the algebraic area of the region on the unit sphere inscribed by the closed line, formed by geodesic arcs, that connects the images of neighbouring faces in the star of the vertex.*

We will refer to this closed line as the *spherical indicatrix* of a vertex ν. The bundle of unit normals, corresponding to $\mathbf{Str(\nu)}$ is called the *normal star* of ν.

The validity of our definition follows from the following observations:

1. If $\mathbf{Str(\nu)}$ is convex, then from the theory of convex polyhedra it is known that the area of the spherical polygon, cut out by the normal star of ν) in the unit sphere, gives the measure of curvature around ν (equal to $\omega(\nu)$).
2. If $\mathbf{Str(\nu)}$ is the star of a mixed vertex, then the Gauss image of ν is not a convex spherical polygon, moreover the spherical indicatrix has self-intersections, partitioning the Gauss image into several simple polygons, possibly overlapping. However, the *convex normal star*, i.e. the normal star of $\mathbf{Str}^+(\nu)$, will give us the measure of the positive curvature ω^+. This is in conformity with the smooth case.
   Indeed, if ν is a mixed vertex then the areas of the Gauss image that represent the parts of negative curvature, are not overlapping with the area of positive curvature. The area of the positive curvature in this case represents a convex simple polygon, which is unique. Note that all the edges of $\mathbf{Str}^+(\nu)$ are also the edges of $\mathbf{Str(\nu)}$, and if $\mathbf{Str(\nu)}$ is not convex, then its deviation from the convex star occurs at an edge of $\mathbf{Str}^+(\nu)$ (see Fig. 3, (iii) and (iv)). We call such an edge the *deviation edge*. The deviation edges occur always in pairs. The dihedral angle at the *deviation edge* in $\mathbf{Str(\nu)}$ is sharper than the corresponding dihedral angle in $\mathbf{Str}^+(\nu)$. Therefore the end-points of the outward normals to the faces that are adjacent to the deviation edges and not the faces of the convex star, will lie outside the spherical polygon formed by the convex normal star. The intersection points of the spherical indicatrix, that separate the positive area of the Gauss map from negative parts, correspond to those faces of $\mathbf{Str}^+(\nu)$ that are not faces of $\mathbf{Str(\nu)}$.
3. An orientation of the contour around the vertex on a polyhedral surface induces the orientation on the spherical indicatrix, i.e. the boundary of the corresponding Gauss image. Thus we can evaluate the curvature around the vertex by computing the area of the spherical image with the sign + (plus) in the case that the orientation is preserved, and with the sign − (minus) otherwise.

In Fig. 4 a convex vertex and a simple mixed vertex are presented together with their corresponding spherical indicatrices.

The above observations are valid for vertices of an embedded (immersed) polyhedral surface $\mathbf{P(V)}$, examples of which were given in Section 2. More complex cases are discussed in the following subsection.

Remark 2. The Gauss image defined above can also be used to determine the Mean curvature H of a polyhedral surface $\mathbf{P}(\mathbf{V})$. The Mean curvature H is

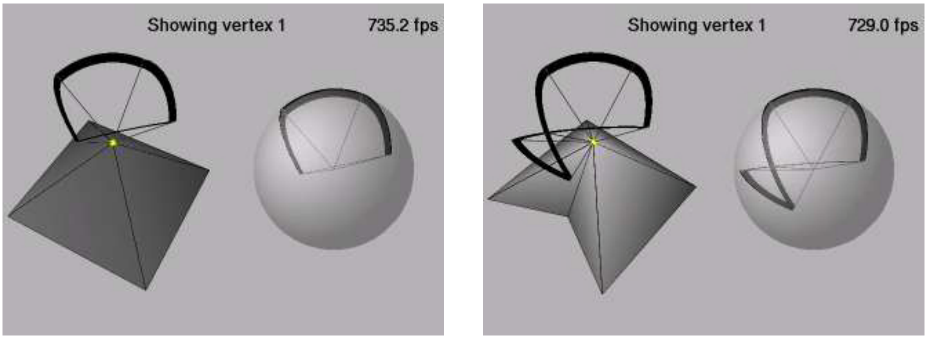


Fig. 4. Left: A convex vertex and its spherical indicatrix. Right: A mixed vertex and its spherical indicatrix

determined along the edges, and for an edge e $H(e)$ equals to (half) the oriented exterior angle $\beta(e)$ between the faces adjacent to e. The absolute value of $\beta(e)$ is equal to the length of the geodesic arc that connects two normals to the faces adjacent to e. In this paper we are interested only in analogues of the Gaussian curvature, and therefore we do not consider the Mean curvature H.

In the next subsection we outline the main steps in our algorithms to construct the Gauss map, to determine the orientation of the spherical indicatrix, to compute and to visualise curvatures directly from the Gauss image.

3.1 Construction of the Polyhedral Gauss Map

To characterise a polyhedral model (i.e. triangle mesh) $\mathbf{P}(\mathbf{V})$ we have developed algorithms that have the following functionalities:

1. *Determination of the Gauss Map of* $\mathbf{P}(\mathbf{V})$, which consists of
   (a) constructing the Gauss map for each of the vertices in V,
   (b) revealing the *Gauss map signature* of each vertex, and computing curvatures incorporated in the Gauss map signature,
   (c) computing the total amount of curvatures regarding its signs, of $\mathbf{P}(\mathbf{V})$;
2. *Curvature Visualisation*, which displays a graphical representation of the Gauss map for a single vertex as well as for the whole surface.

To construct directly the Gauss map for a given vertex, we first construct the outward vector normals for each of the faces in the star of the vertex, 'translate' them in the same origin, and then draw geodesics arcs between them to obtain a graphic image. The image, in most cases, consists of several spherical simple polygons, which are sometimes overlapping. We are able to identify each simple spherical polygon in the Gauss image of a vertex, determine the orientation of each polygon and thus its sign. Therefore, we are able not only to separate $\omega(\nu)$ for a vertex ν into positive and negative parts $\omega^{+}(\nu)$ and $\omega^{-}(\nu)$, but into subparts of the same sign. The number of subparts together with their signs represents the *Gauss map signature* of a vertex. Each subpart of the negative sign represents a potential (hidden) saddle region.

The advantage of our method is that it allows the determination of incorporated curvatures of various types of vertices, including all the above-mentioned ones and much more complex ones such as: the *monkey saddle*, other types of *generalised saddles*, and vertices with *self-intersections*, which do not fit exactly in the category ‘mixed’, described in the previous section. Eventually, we can determine the curvatures of a vertex of any type (of an oriented polyhedral surface P). It is also possible to display the Gauss Map for all the vertices of the object simultaneously, or select only one of the vertices for its Gauss Map to be shown exclusively (or, correspondingly, to visualise the Gauss map of a region on the surface). The method is interactive, and we can visualise the regions of positive curvature separately from the regions of negative curvature.

3.2 Algorithm Description

Input data are taken from text files in the Alias/Wavefront format OBJ. An OBJ file describes a polyhedral model by the list of the coordinates of the vertices, and by the indices of vertices that compose the faces. Vertices are described by their three coordinates in space, and are automatically given an index number, beginning with 1. The faces are described as a list of vertex indices that delimit a polygon. The order of the vertices is generally counter clockwise, and thus the algorithm assumes this ordering.

Vertices and faces read from the OBJ file are stored in arrays. Before computing the Gauss Map, some more information is obtained from the raw data. The normal vector of each face is obtained by the cross product. The faces that belong to the star of a vertex are stored in lists. For each vertex, its faces are ordered so that they will be visited in counter clockwise order.

The process of obtaining the Gauss Map for a single vertex as follows:

- Take the unit normal vectors for all the faces in the star of the vertex, $\mathbf{Str}(\nu)$.
- The normals are kept in the same order as the faces are visited in counter clockwise order.
- All the normals are translated to the same origin, forming the normal star of the vertex. Their endpoints will lie on the surface of a unit sphere.
- The normal’s endpoints are joined together according to their order. Since they lie in the surface of a sphere, the shortest line between two of them is a segment (arc) of a great circle of the sphere.
- The arcs thus defined form the spherical indicatrix, and will delimit an area on the surface of the sphere, which is composed of one or more spherical polygons.
- If the spherical indicatrix is self-intersecting, then there are more than one spherical polygons. In this case, new vectors must be added, that will have their end-points at the location where two arcs intersect. The set of the face normal vectors plus these new intersection vectors is called the *extended star* of the vertex.

- The new vectors identify points of separation between the areas of the Gauss Map. These points are common points for two or more spherical polygons.
- The orientation of each of these spherical polygons is determined by following the corresponding loop of the spherical indicatrix, in the order already established for the normals (by visiting the faces around the vertex in counter clockwise order). The orientation will be positive if the walk along the loop is counter clockwise, and negative if the loop is clockwise. Accordingly, the area of a polygon of the positive orientation is considered as positive, and of the negative orientation as negative.
- The Gauss image of a single vertex may have all of its area positive, or all negative, or have spherical polygons with both orientations. The vertex is then classified, in accordance with its Gauss Map, as positive, negative or mixed, respectively. These classification differs from the classification, given in Section 2. A convex vertex is positive, but the positive vertex may have self-intersections.
- The area of the Gauss Map is computed as a sum of the areas of the individual spherical polygons, obtained with the well-known *spherical excess formula*:

  $$\text{Area} = (\sum \alpha_i - (n-2)\pi), \tag{11}$$

  where α_i is an internal angle of the polygon; n is the number of the sides of the polygon.

  The area of each individual spherical polygon will be less than 2π in most cases.

Below is the description of the algorithm

```
For each vertex $v$ in $\mathbf{V}$
{
    face-list → Get the list of faces for the current vertex.
    Order the faces around the current vertex in CCW
        (counter clockwise) orientation.
    normal-list → List of the normals from face-list
    extended-normal-list → Find arc intersections in
        normal-List
    If there are any arc intersections
    {
        polygon-list → Divide into areas using
        extended-normal-list
    }
    Else check for the special case of the ‘monkey saddle’
    {
        polygon-list → Check for duplicate normals in
        normal-list
    }
    For each spherical polygon $p$ in polygon-list
```

```
        {
            TotalGaussMapArea → TotalGaussMapArea +
            spherical polygon area of $p$
            Count CCW, CW (clockwise) and FLAT polygons
        }
        Classify the type of vertex.
        If all polygons are CCW
            Vertex $v$ is positive
        Else if all polygons are CW
            Vertex $v$ is negative
        Else if all polygons are FLAT
            Vertex $v$ is flat
        Else
            Vertex $v$ is mixed
    }
```

Special Cases. There are several special cases, but they belong mostly to one of the three types, listed below, or a combination of these types; each case is dealt with in a different way:

- **A vertex with pairwise coinciding normals, related to non-adjacent faces**. An example is a vertex of the configuration commonly known as the *monkey saddle*, with its 6 faces equally sized. It has three pairs of pairwise coinciding normals. This gives us three pairs of identical vectors, and the Gauss image has, therefore, two overlapping identical areas. But in its spherical indicatrix there is no clear intersection of the arcs between the normal vectors. This case is solved by setting that coinciding normal vectors are not equal among them if no arc intersections are identified.
- **A vertex with the spherical indicatrix, self-intersecting several times at the same point**. An example is a vertex that can be constructed by adding two more faces to a monkey saddle. The star of such a vertex has four upward slopes and four downward slopes. We call such a vertex a *tulip*. The arcs of its spherical indicatrix intersect themselves four times precisely at the same point. This new point will be common to four different spherical polygons, so we must keep several copies of this point.
- **The area of a simple spherical polygon is greater than 2π.** A case of this type arises only if two simple spherical polygons are 'glued' together, i.e. have the common boundary. This requires additional checks. The main check consists of determining whether the star of the vertex is a saddle, which can be done by computing the convex star of the vertex. If the vertex is a saddle than the vertex lies inside the convex hull. On the basis of this check an appropriate sign is assigned to the area of the polygon.

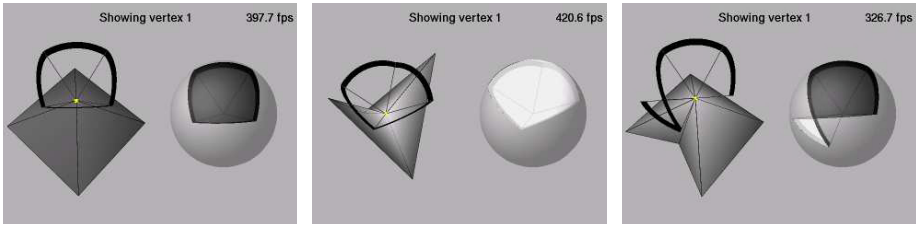


Fig. 5. Gauss images of the basic types of vertices: convex, saddle and mixed

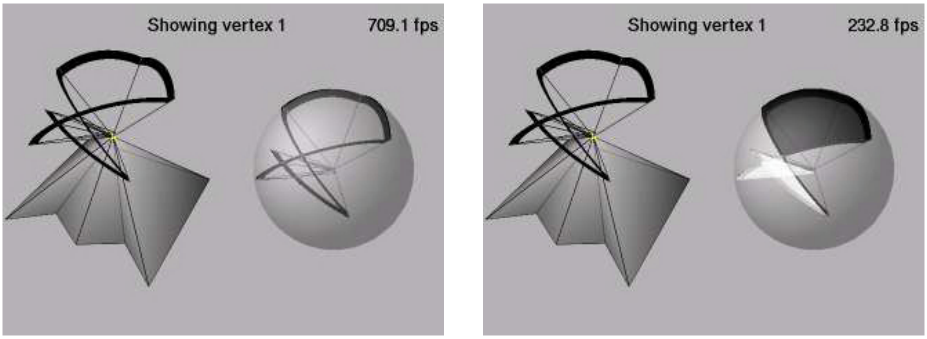


Fig. 6. A complex mixed vertex and its spherical indicatrix (left) and Gauss image (right)

4 Results—Examples of Curvature Visualisation

Examples of Gauss map visualisations are given below. The display of the Gauss Map is done in two different views, or scenes, and is implemented using OpenGl. The left scene shows the model of the original object and, in the right scene, the areas for the Gauss Map are drawn on top of a sphere. Positive areas are shown in black, while negative areas are displayed in white.

Gauss Images of Vertices. In this subsection we present examples of Gauss images of several types of vertices. The Gauss images of three basic types of vertices, described in Subsection 2.2, are given in Fig. 5.

In Fig. 6 the Gauss image of a more complex mixed vertex is shown. In the left picture the image of the vertex is presented together with its spherical indicatrix, and in the right picture the parts of positive and negative area are indicated. The part of positive area is clearly separated from the parts of negative area, which are overlapping. We can single out two simple spherical polygons of negative area, which, loosely speaking, correspond to two 'hidden' saddle domains in the vertex.

In Fig. 7 two views of the Gauss image of the monkey saddle, described in the paragraph 'Special Cases', are given. The Gauss image looks similar to the Gauss of the saddle vertex, but consists eventually of two completely overlapping spherical polygons.

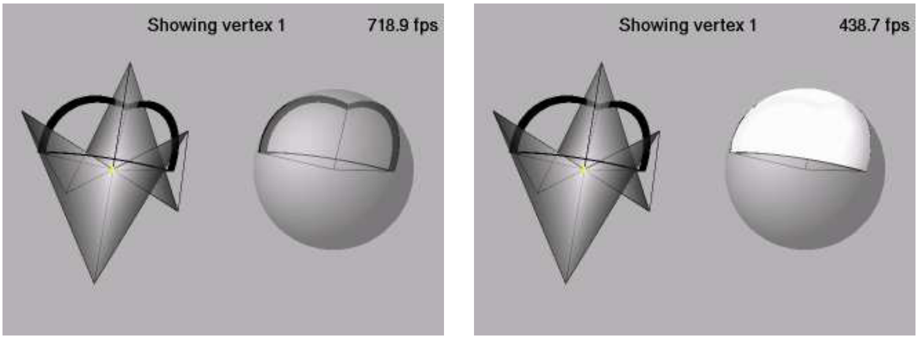


Fig. 7. Gauss images of the monkey saddle

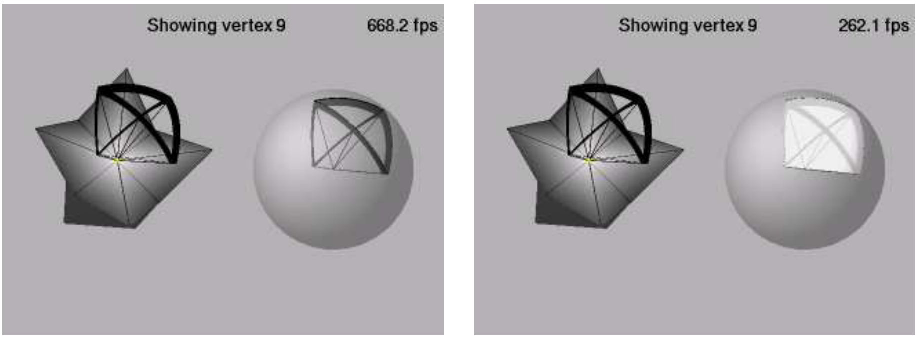


Fig. 8. Gauss images of the tulip vertex

In Fig. 8 Gauss image visualisations of the tulip vertex are presented. Its Gauss image consists of four equal spherical polygons, pairwise adjacent to each other, which are revealed by the spherical indicatrix. However the coloured Gauss image looks like one spherical polygon with four sides.

Gauss image visualisations of a generalised monkey saddle are presented in Fig. 9. Again, as in the previous pictures, in left picture only the spherical indicatrix is shown in order to provide a better understanding of the complex Gauss image.

The vertices, presented in Figs. 5–8, are embedded vertices, and their Gauss maps are in conformity with curvature characterisations for embedded polyhedral surfaces given in [3]. In the next few pictures we present Gauss map visualisations of vertices with self-intersections. This provides curvature characterisation of more complex surfaces, which are neither embedded nor immersed. Figs. 10 and 11 shows the Gauss map visualisations of two vertices, which we call a *pinch vertex* and a *reverse pinch vertex*. In order to understand the difference between a pinch vertex and a reverse pinch vertex, imagine a walk along the link of the star of a vertex ν. In the case of the pinch vertex the walk makes two full turns around the vertex, both turns having the same orientation (for example,

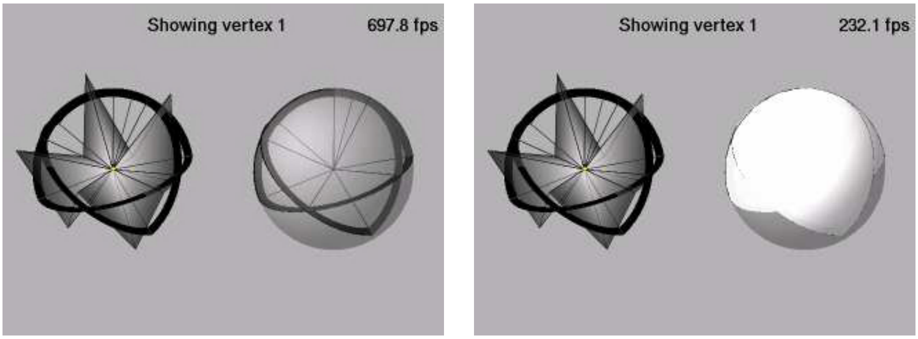


Fig. 9. Gauss images of a generalised monkey saddle

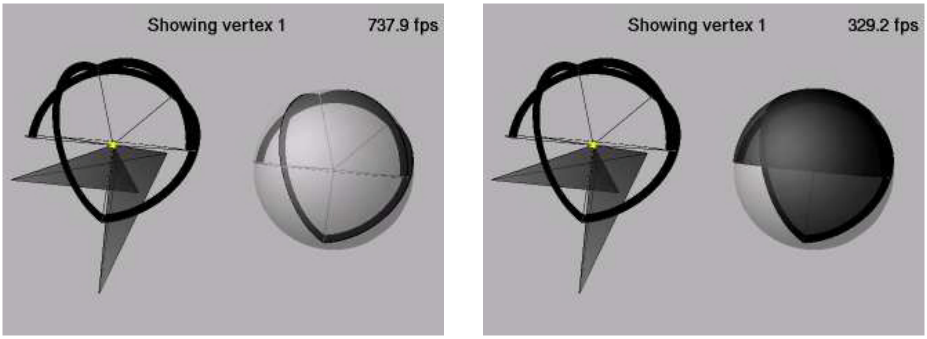


Fig. 10. Pinch vertex with the corresponding Gauss map

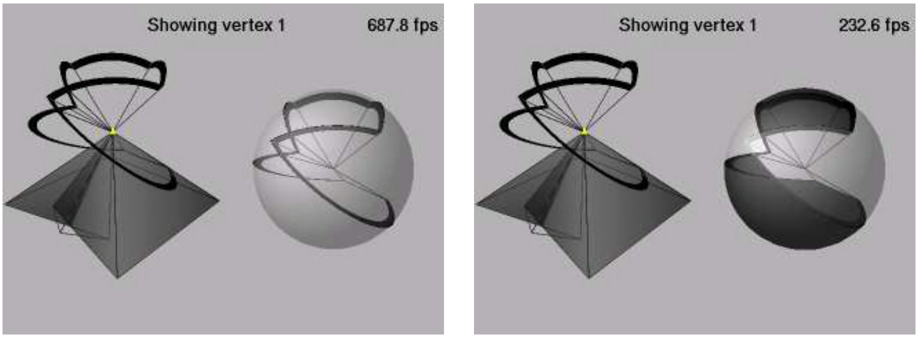


Fig. 11. Reverse pinch vertex with the corresponding Gauss map

counter clockwise). In the case of the reverse pinch point, the walk makes also two full turns, one, for example, in the counter clockwise direction, and the second one in the ‘reverse’ direction (i.e. clockwise). The Gauss map of the pinch vertex, presented in Fig. 10, has two overlapping areas, each of positive sign. One area is equal to the curvature of the convex star of the pinch vertex.

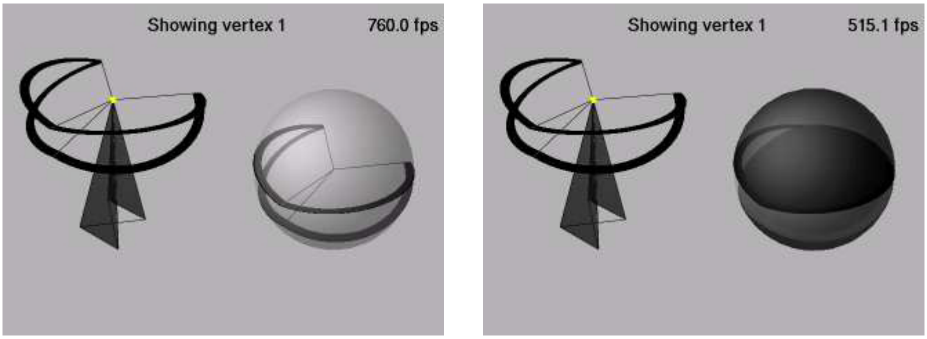


Fig. 12. Cone-eight with the corresponding Gauss map

The Gauss map of the reverse pinch vertex, presented in Fig. 11, has also two areas of positive curvature, but they are non-overlapping and separated by the area of negative curvature.

Finally, we show the Gauss map visualisation of the vertex whose link represents the figure ‘eight’, for simplicity referred to as the *cone-eight*. Such a vertex can also be considered as the ‘extreme’ case of the reverse pinch vertex, when the negative curvature part has disappeared. The two remaining positive parts are ’glued’ together and we have the third special case, described in the previous subsection. The Gauss image of the vertex, presented in Fig. 12, almost covers the whole sphere.

Gauss Images of Complex Surfaces. In this paragraph we present curvature visualisations of surfaces of several complex objects. In these visualisations we emphasise domains of a surface with respect to their incorporated curvatures. Each domain has vertices of the same curvature type. Therefore we single out the following domains: domains of positive curvature, whose vertices possess positive or zero curvature, and that are coloured in black in conformity with the colours of our Gauss map; domains of negative curvature, accordingly coloured in white, whose vertices are saddle-type vertices,; and domains of mixed curvature, coloured in grey, whose vertices are mixed. We visualise curvature domains by two methods: by the GMS method and by means of angle deficit.

In Fig. 13 visualisations of a torus are presented. Two views for each curvature visualisation are given. The left picture in each figure is the curvature visualisation of a torus by using the angle deficit, and the right picture - by means of the GMS method. Both visualisations are presented with the Gauss image of a selected vertex. One can clearly see that the curvature visualisation by means of the angle deficit does not reflect the ‘waviness’ of the torus. The vertex, determined as a vertex of positive curvature by means of the angle deficit, has eventually two parts of negative curvature, and is a mixed vertex, which is clearly seen in the curvature visualisation by means of the GMS method. The GMS visualisation indicates that the mesh may not be optimal.

By applying the optimisation method based on the minimsation of total absolute extrinsic curvature, introduced in [1] and later further developed in [8, 2],

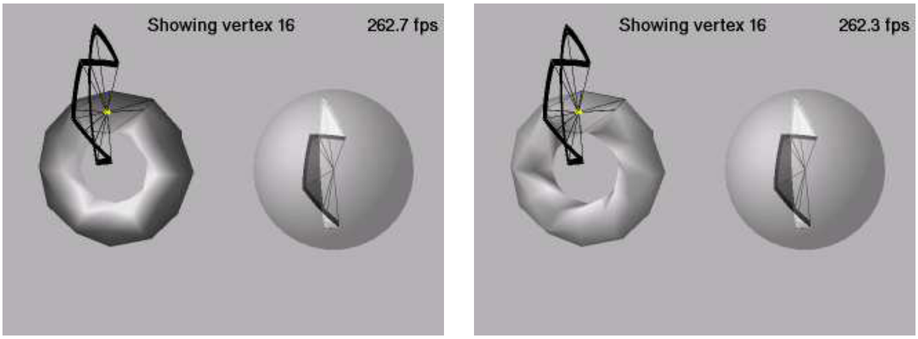


Fig. 13. Two visualisations of the initial torus with the Gauss image of a selected vertex

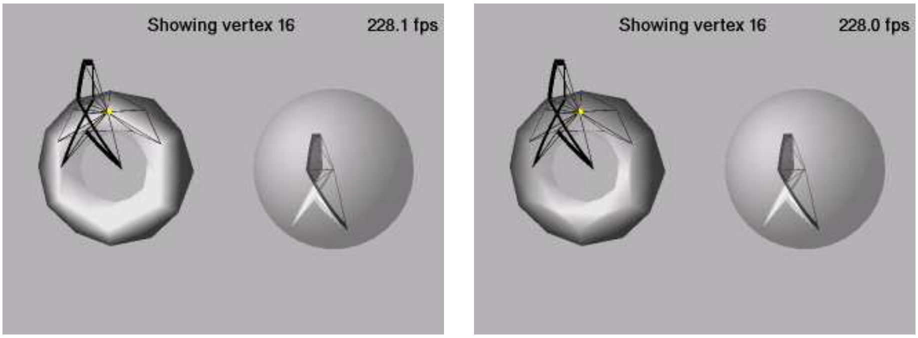


Fig. 14. Two visualisations of the optimised torus with the Gauss image of a selected vertex

we obtain an optimal mesh for this torus. The visualisations of the optimal mesh are given in Fig 14 (with the Gauss image of a selected vertex).

Again, as in Fig. 13, in the left picture of Fig. 14 the visualisation based on the angle deficit is presented. We can see that now the two curvature visualisations are almost identical. In the right view, mixed vertices form a thin circular ‘grey’ domain on the ‘top’ (and ‘bottom’) of the torus that separates the domains of positive and negative curvature. This domain is not present in the left view. The Gauss image of a vertex situated in this domain reflects the almost symmetrical distribution of negative and positive curvatures in this vertex.

In the last two figures two similar views of the curvature visualisation of a torso are presented. In Fig. 15 the visulisation is given with the full Gauss image and in Fig. 16 with the image of a selected vertex. The mesh can be considered as optimal in this case; however, the angle deficit method does not describe the curvature domains of the mesh correctly, as it cannot recognise mixed vertices. In this visualisation a vertex determined as a saddle by the angle deficit method, eventually describe the curvature domains of the mesh correctly.

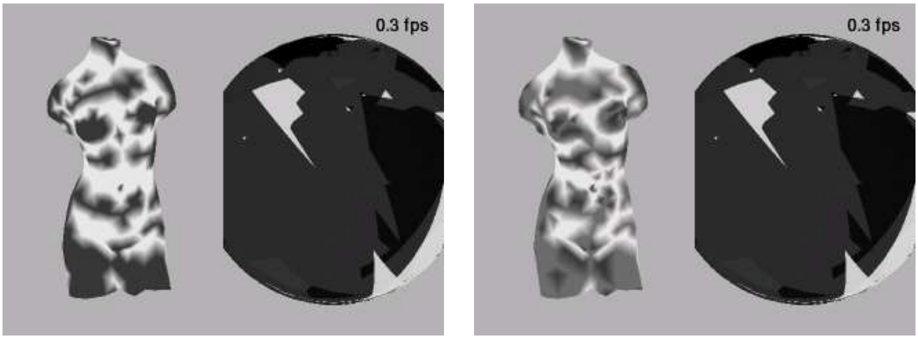


Fig. 15. Two visualisations of the torso with the full Gauss image

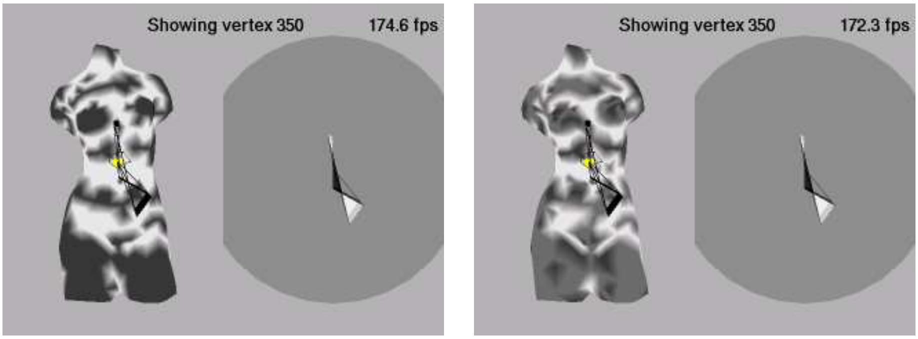


Fig. 16. Two visualisations of the torso with a selected vertex

5 Conclusions and Future Work

The conclusions which may be made from the results presented in the previous section can be summarised as follows:

- We develop a novel method to construct Gauss maps directly from a polyhedral surface. The method is theoretically valid and robust in implementation. It constructs Gauss images for a very large class of polyhedral surfaces, including non-manifold surfaces.
- Visualisation of Gauss images enables us to reveal all existing domains of positive and negative curvature of a surface. These domains, except for very special cases, cannot be determined by means of the angle deficit.
- The advantage of the discrete curvature computation by using the GMS with respect to discrete curvature computation presented in Section 2, is that it allows to split the curvature domains of the same sign into sub-domains. This gives us a more complete picture of geometric features of the surface and may be very useful for shape characterisation and description of an object.
- Curvature characterisation of a surface by means of the angle deficit is sufficient neither to capture the curvature information of a polyhedral surface nor

to estimate the Gaussian curvature of the underlying smooth surface from a mesh. However, it can serve as an indicator that a mesh should be optimised in case of a significant discrepancy in curvature visulisation of a surface between the GMS method and the angle deficit method. Indeed, for estimating the Gaussian curvature at a point ν of the underlying smooth surface the normalised angle deficit, i.e. $3\omega(\nu)/Area(\mathbf{Str}(\nu)$, where $Area(\mathbf{Str}(\nu)$ is the sum of the areas of triangles in $\mathbf{Str}(\nu)$, is widely used. But the triangles in $\mathbf{Str}(\nu)$ are determined by the connectivity of the mesh. Therefore, roughly speaking, the connectivity of the mesh influences estimation of the curvature, and therefore, obtained discrete estimates might differ considerably from curvatures of the underlying smooth surface.

- The GMS method might be useful to check the validity of the mesh in the cases when we presume that the mesh should be without self-intersections. All 'wrong' self-intersections in vertices will be revealed by the GMS.

We are currently working on the algorithm that will visualise 'hidden' domains not only in the star of a vertices, but also in a whole surface. It means that the grey areas in a surface that indicate the domains of mixed vertices, will be replaced by the domains of positive and negative curvatures. This will be very useful for evaluating the accuracy of curvature estimates of the underlying smooth surface directly from a mesh as well as for developing subdivision schemes that will guarantee the convergence of these estimates when the mesh is refined.

Acknowledgments. We would like to thank Dr. Alan Robinson and Dr. Jacques Penders for helpful comments, and Prof. Marcos Rodrigues for creating pleasant and productive research environment. We also want to thank Martin Kilian for his questions, which helped to instigate this research; and Prof. Wolfgang Kühnel for contributions to the theoretical background of this work; and the anonymous reviewers, whose comments improved the contents of the paper.

References

1. Alboul, L. and van Damme, R. Polyhedral metrics in surface reconstruction. In: Mullineux, G. (ed.), *The Mathematics of Surfaces VI*, pp. 171-200, Oxford, 1996.
2. Alboul, L., Optimising triangulated polyhedral surfaces with self-intersections. In: Wilson, M. J., Martin, R. R., *Mathematics of Surfaces*, LNCS 2768, pp. 48-72, 2003.
3. Banchoff, T.F., Critical points and curvature for embedded polyhedral surfaces. *Amer. Math. Monthly* 77, pp. 475-485, 1970.
4. Banchoff, T.F., and Kühnel, W., Tight submanifolds, smooth and polyhedral. In: Cecil T. E., and Chern S.-S. (eds.),*Tight and Tight Submanifolds*, MSRI Publications, v. 32, pp. 52-117, 1997.
5. Borelli, V., Cazals, F, and Morvan J.-M., On the angular defect of triangulations and the pointwise approximation of curvature. *Comp. Aided Geom. Design* 20, pp. 319-341, 2003.

6. Brehm, U., and Kühnel, W., Smooth approximation of polyhedral surfaces with respect to curvature measures. In: *Global differential geometry*, pp. 64-68, 1979.
7. Calladine, C. R., Gaussian Curvature and Shell Structures, In: J. A. Gregory (ed.), *The Mathematics of Surfaces*, pp. 179-196, University of Manchester, 1984.
8. Dyn N. et al., Optimising 3D triangulations using discrete curvature analysis. In: Lyche, T., and Schumaker, L. L. (eds.), *Mathematical Methods in CAGD*, pp. 1-12, Vanderbilt University Press, 2001.
9. Kilian M., Differentialgeometrische Konzepte für Dreiecksnetze, Master thesis, University of Karlsruche, 135 pp., 2004.
10. Kühnel, W., Differential geometry. Curves-Surfaces-Manifolds, *Amer Math Society*, 2002.
11. Little J. J., Extended Gaussian images, mixed volumes, shape reconstruction. In: Proc. of the first annual symposium on Comp. Geom., pp. 15-23, 1985.
12. Lowekamp, B., Rheingans, P., and Yoo, T.S., Exploring surface characteristics with interactive Gaussian images: a case study. *Proc. of the conference on Visualization '02*, pp. 553-556, 2002.
13. Maltret, J.-L., and Daniel, M., Discrete curvatures and applications: a survey. *Research report LSIS-02-004*, March 2002.
14. Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H., Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, H.-Ch., and Polthier, K. (eds.), *Visualization and Mathematics III*, pp. 35-59, Springer-Verlag, 2003.
15. Peng, J., Li, Q., Ja Kuo, C.-C., and Zhou, M., Estimating Gaussian Curvatures from 3D Meshes. In: Rogowitz, B.E., Pappas, Th. N. (eds.)
16. Sullivan, J., Curvature measures for discrete surfaces. http://torus.math.uiuc.edu/jms/Papers/dscrv.pdf

Manifold Embedding of Graphs Using the Heat Kernel

Xiao Bai, Richard C. Wilson, and Edwin R. Hancock

Department of Computer Science,
University of York, York Y010 5DD, UK
{baixiao, wilson, erh}@cs.york.ac.uk

Abstract. In this paper, we investigate the use of heat kernels as a means of embedding the individual nodes of a graph on a manifold in a vector space. The heat kernel of the graph is found by exponentiating the Laplacian eigen-system over time. We show how the spectral representation of the heat kernel can be used to compute both Euclidean and geodesic distances between nodes. We use the resulting pattern of distances to embed the nodes of the graph on a manifold using multi-dimensional scaling. The distribution of embedded points can be used to characterise the graph, and can be used for the purposes of graph clustering. Here for the sake of simplicity, we use spatial moments. We experiment with the resulting algorithms on the COIL database, and they are demonstrated to offer a useful margin of advantage over existing alternatives.

1 Introduction

One of the problems that arises in the manipulation of large amounts of graph data is that of embedding the nodes of individual graphs in a low dimensional space. This is a problem that arises in a number of areas including the clustering, matching and drawing of graphs. Stated succinctly, the problem is that of locating a mapping between the nodes of the graph and points in space. Once graph nodes have been mapped to points, then geometric methods can be used to characterise the original graph. Drawing the graph becomes the problem of joining the points, graph matching the problem of aligning them, and graph-clustering the problem of characterising the point distribution using moments or variance measurements. In other words, by using the mapping the discrete problems of graph manipulation can be re-posed as computationally simpler geometric operations.

In the mathematics literature, there is a considerable body of work aimed at understanding how graphs can be embedded on manifolds [1]. Broadly speaking there are three ways in which the problem has been addressed. First, the graph can be interpolated by a surface whose genus is determined by the number of nodes, edges and faces of the graph. Second, the graph can be interpolated by a hyperbolic surface which has the same pattern of geodesic (internode) distances

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 34–49, 2005.

as the graph [2, 3]. Third, a manifold can be constructed whose triangulation is the simplicial complex of the graph [4, 5].

In the pattern analysis community, on the other hand, there has recently been renewed interest in the use of embedding methods motivated by graph theory. One of the best known of these is ISOMAP [6]. Here a neighborhood ball is used to convert data-points into a graph, and Dijkstra's algorithm is used to compute the shortest(geodesic) distances between nodes. By applying multidimensional scaling (MDS) to the matrix of geodesic distances the manifold is reconstructed. The resulting algorithm has been demonstrated to locate well-formed manifolds for a number of complex data-sets. Related algorithms include locally linear embedding which is a variant of principal component analysis (PCA) that restricts the complexity of the input data using a nearest neighbor graph [7], and the Laplacian eigen-map that constructs an adjacency weight matrix for the data-points and projects the data onto the principal eigenvectors of the associated Laplacian matrix (the degree matrix minus the weight matrix) [8]. Collectively, these methods are sometimes referred to as manifold learning theory.

The spectrum of the Laplacian matrix has been widely studied in spectral graph theory [9] and has proved to be a versatile mathematical tool that can be put to many practical applications including routing [10], indexing [11], clustering [12] and graph-matching [13, 14]. One of the most important properties of the Laplacian spectrum is its close relationship with the heat equation. The heat equation can be used to specify the flow of information with time across a network or a manifold [15]. According to the heat-equation the time derivative of the kernel is determined by the graph Laplacian. The solution to the heat equation is obtained by exponentiating the Laplacian eigen-system over time. Because the heat kernel encapsulates the way in which information flows through the edges of the graph over time, it is closely related to the path length distribution on the graph. The graph can be viewed as residing on a manifold whose pattern of geodesic distances is characterised by the heat kernel. For short times the heat kernel is determined by the local connectivity or topology of the graph as captured by the Laplacian, while for long-times the solution gauges the global geometry of the manifold on which the graph resides.

For a Riemannian manifold, the heat kernel is determined by the pattern of geodesic distances, and can provide a means of analysing both the local and global differential geometry of the manifold. In particular, differential invariants can be computed from the heat kernel, and these in turn are related to the Laplacian eigen-system. This field of study is sometimes referred to as spectral geometry [16, 15]. One of the most interesting recent developments in this area has been to establish a link between graph-spectra and the geometry of the underlying manifold [17, 18, 19, 20]. Here considerable insight can be achieved through the analysis of the heat kernel of the graph [17, 19]. In the pattern analysis area Lebanon and Lafferty [18] have used the heat-kernel to construct statistical manifolds that can be used for inference and learning tasks.

There are a number of different invariants that can be computed from the heat-kernel. Asymptotically for small time, the trace of the heat kernel [9] (or the sum of the Laplacian eigenvalues exponentiated with time) can be expanded as a rational polynomial in time, and the co-efficients of the leading terms in the series are directly related to the geometry of the manifold. For instance, the leading co-efficient is the volume of the manifold, the second co-efficient is related to the Euler characteristic, and the third co-efficient to the Ricci curvature. The zeta-function (i.e. the sum of exponentials found by raising the eigenvalues to a non-integer power) for the Laplacian also contains geometric information. For instance its derivative at the origin is related to the torsion tensor for the manifold. Finally, Colin de Verdiere has shown how to compute geodesic invariants from the Laplacian spectrum [21].

The aim in this paper is to develop a means of embedding the nodes of graphs as points on a manifold in a vector space. In other words, we seek a mapping from the node-set of the graph to points in a vector-space. To do this we make use of multidimensional scaling. This is a technique that allows data specified in terms of relative distances rather than ordinal values to be embedded in a low-dimensional space in a manner that minimises the distortion (or stress) of the distance pattern. To apply this technique to graphs we require a means of assigning a distance measure to the graph-edges. Our distance function is furnished by an analysis of the heat-kernel. When the manifold on which the nodes of the graph reside is locally Euclidean, then the heat kernel may be approximated by a Gaussian function of the geodesic distance between nodes. By equating the spectral and Gaussian forms of the kernel, we can make estimates of the Euclidean distances (i.e. the shortest distance in the vector space) between the nodes of the graph under study. The geodesic distance (i.e the shortest distance on the manifold) is given by the floor of the path-length distribution, and this may be computed from the Laplacian spectrum. The difference between the geodesic and the Euclidean distances is related to the sectional curvature of the geodesics corresponding to edges of the embedded graph on the manifold. The manifold may be approximately reconstructed by using multi-dimensional scaling to locate a low-distortion embedding of the Euclidean distances.

Once embedded in this space, we can perform a number of graph-manipulation tasks by applying simple point-pattern analysis algorithms to the mapped node positions. Here we focus on the problem of graph clustering. Our approach is to extract feature-vectors that characterise the point distributions that result from the embedding the nodes of the graphs. Each graph has an associated feature-vector that characterises how its nodes are distributed in space. Sets of graphs can be projected into a pattern space by performing principal components analysis on the feature-vectors. We characterise the shape of point-distribution of the embedded nodes using affine invariant spatial moments.

The outline of this paper is as follows. In Section 2 we show how the analysis of the heat-kernel can lead to a measure of Euclidean distance between individual nodes of a graph. Section 3 discusses the spectral geometry of the manifold, and shows how the spectral estimates of geodesic and Euclidean distances derived in

Section 2 can be used to estimate the sectional curvature of the manifold. Section 4 describes the Euclidean multidimensional scaling method used to embed the graphs in low-dimensional space. Section 5 details the methods used to cluster the embedded graphs. In Section 6, we present experiments on real world synthetic data. Finally, Section 7 offers conclusions and suggests directions for future work.

2 Heat Kernels on Graphs

In this section, we develop a method for approximating the Euclidean distance between the nodes of a graph by exploiting the properties of the heat kernel. To commence, suppose that the graph under study is denoted by $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Since we wish to adopt a graph-spectral approach we introduce the adjacency matrix A for the graph where the elements are

$$A(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We also construct the diagonal degree matrix D, whose elements are given by $D(u,u) = \deg(u) = \sum_{v \in V} A(u,v)$. ¿From the degree matrix and the adjacency matrix we construct the Laplacian matrix $L = D - A$, i.e. the degree matrix minus the adjacency matrix. The normalised Laplacian is given by $\hat{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. The spectral decomposition of the normalised Laplacian matrix is

$$\hat{L} = \Phi \Lambda \Phi^T = \sum_{i=1}^{|V|} \lambda_i \phi_i \phi_i^T \tag{2}$$

where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is the diagonal matrix with the ordered eigenvalues $(0 = \lambda_1 < \lambda_2 \leq \lambda_3 ...)$ as elements and $\Phi = (\phi_1 | \phi_2 | .... | \phi_{|V|})$ is the matrix with the correspondingly ordered eigenvectors as columns. Since $\hat{L}$ is symmetric and positive semi-definite, the eigenvalues of the normalised Laplacian are all positive. The eigenvector ϕ_2 associated with the smallest non-zero eigenvalue λ_2 is referred to as the Fiedler-vector. We are interested in the heat equation associated with the Laplacian, i.e.

$$\frac{\partial h_t}{\partial t} = -\hat{L} h_t \tag{3}$$

where h_t is the heat kernel and t is time. The heat kernel can hence be viewed as describing the flow of information across the edges of the graph with time. The rate of flow is determined by the Laplacian of the graph. The solution to the heat equation is found by exponentiating the Laplacian eigen-spectrum, i.e.

$$h_t = \sum_{i=1}^{|V|} \exp[-\lambda_i t] \phi_i \phi_i^T = \Phi \exp[-t\Lambda] \Phi^T \tag{4}$$

The heat kernel is a $|V| \times |V|$ matrix, and for the nodes u and v of the graph G the resulting element is

$$h_t(u,v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t]\phi_i(u)\phi_i(v) \tag{5}$$

When t tends to zero, then $h_t \simeq I - \hat{L}t$, i.e. the kernel depends on the local connectivity structure or topology of the graph. If, on the other hand, t is large, then $h_t \simeq \exp[-t\lambda_2]\phi_2\phi_2^T$, where λ_2 is the smallest non-zero eigenvalue and ϕ_2 is the associated eigenvector, i.e. the Fiedler vector. Hence, the large time behavior is governed by the global structure of the graph.

2.1 Geodesic Distance

It is interesting to note that the heat kernel is also related to the path length distribution on the graph. To show this, consider the matrix $P = I - \hat{L}$, where I is the identity matrix. The heat kernel can be rewritten as $h_t = e^{-t(I-P)}$. We can perform a McLaurin expansion on the heat-kernel to re-express it as a polynomial in t. The result of this expansion is

$$\begin{aligned} h_t &= e^{-t}\left(I + tP + \frac{(tP)^2}{2!} + \frac{(tP)^3}{3!} + \cdots\right) && (6)\\ &= e^{-t}\sum_{k=0}^{\infty} P^k \frac{t^k}{k!} && (7) \end{aligned}$$

The matrix P has elements

$$P(u,v) = \begin{cases} 1 & \text{if } u = v \\ \frac{1}{\sqrt{\deg(u)\deg(v)}} & \text{if } u \neq v \text{ and } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

As a result, we have that

$$P^k(u,v) = \sum_{S_k} \prod_{i=1}^{k} \frac{1}{\sqrt{\deg(u_i)\deg(u_{i+1})}} \tag{9}$$

where the walk S_k is a sequence of vertices $u_0, \cdots, u_k$ of length k such that $u_i = u_{i+1}$ or $(u_i, u_{i+1}) \in E$. Hence, $P^k(u,v)$ is the sum of weights of all walks of length k joining nodes u and v. In terms of this quantity, the elements of the heat kernel are given by

$$h_t(u,v) = \exp[-t] \sum_{k=0}^{|V|^2} P^k(u,v)\frac{t^k}{k!} \tag{10}$$

Hence, the heat kernel takes the form of a sum of Poisson distributions over the path-length with time as parameter. The weights associated with the different components are determined by $P^k(u,v)$. As the path-length k becomes large, the Poisson distributions approach a Gaussian, with mean k and variance k.

A spectral expression for the matrix P^k can be obtained using the eigendecomposition of the normalised Laplacian. Writing $P^k = (I - \hat{L})^k = \Phi(I - \Lambda)^k \Phi^T$, the element associated with the nodes u and v is

$$P^k(u,v) = \sum_{i=1}^{|V|} (1 - \lambda_i)^k \phi_i(u)\phi_i(v) \tag{11}$$

The geodesic distance between nodes, i.e. the length of the walk on the graph with the smallest number of connecting edges, can be found by searching for the smallest value of k for which $P^k(u,v)$ is non zero, i.e. $d_G(u,v) = \text{floor}_k P^k(u,v)$.

2.2 Euclidean Distance

Here we are interested in using the heat-kernel to compute an approximate Euclidean distance between nodes. This is the shortest distance between nodes in the vector space in which the manifold resides. Knowledge of this distance is important if Euclidean multidimensional scaling is to be used to reconstruct the manifold, i.e. embed the individual node of the graph in a low-dimensional space. Asymptotically with small t, on a Riemannian manifold the heat kernel can be approximated by the so-called paramatrix [22]

$$h_t(u,v) = [4\pi t]^{-\frac{n}{2}} \exp[-\frac{1}{4t} d_G(u,v)^2] \sum_{m=0}^{\infty} b_m(u,v) t^m \tag{12}$$

where $d_G(u,v)$ is the geodesic distance between the nodes u and v on the manifold, n is the dimensionality of the space (taken to be 3 in our later experiments with MDS) and $b_m(u,v)$ real-valued co-efficients. When the manifold is locally Euclidean, then only the first term in the polynomial series is significant and the heat kernel is approximated by a Gaussian. Hence, to approximate the Euclidean distance between nodes in the embedding we can equate the spectral and Gaussian forms for the kernel. The result is

$$d_E^2(u,v) = -4t \ln\left\{ (4\pi t)^{\frac{n}{2}} \sum_{i=1}^{|V|} \exp[-\lambda_i t] \phi_i(u)\phi_i(v) \right\} \tag{13}$$

An alternative form for the above is

$$d_E^2(u,v) = -\ln\left\{ (4\pi t)^{2nt} \left[\sum_{i=1}^{|V|} \exp[-\lambda_i t] \phi_i(u)\phi_i(v) \right]^{4t} \right\} \tag{14}$$

and this leads us to the spectral form for the distance matrix

$$d_E^2 = -\ln[(4\pi t)^{2nt} \Phi \exp[-4\Lambda t^2] \Phi^T] \tag{15}$$

Hence, the distance matrix is just the negative logarithm of a matrix found by applying a transformation to the eigenvalue spectrum. Such a procedure was suggested by Smola and Kondor [20] as a means for regularising graphs. However, the function suggested above is more complex than the alternatives explored for regularisation.

Since $h_t = \exp[-\hat{L}t]$, we can write $d_e^2 = 4\hat{L}t^2 - 2nt\ln[4\pi t]$. As a result if u and v are connected by an edge, and $u \neq v$, then $d_E^2(u,v) = -\frac{4t^2}{\sqrt{\deg(u)\deg(v)}} - 2nt\ln[4\pi t]$. Hence, $d_E(u,v)$ tends to zero as t tends to zero.

3 Multidimensional Scaling

In order to explicitly reconstruct the manifold, we embed the pattern of Euclidean distances in a low dimensional space in a manner which minimises the distortion using multidimensional scaling(MDS). The pairwise Euclidean distances between nodes $d_E(u,v)$ are used as the elements of an $|V| \times |V|$ dissimilarity matrix S, whose elements are defined as follows

$$S(u,v) = \begin{cases} d_E^2(u,v) & \text{if } u \neq v \\ 0 & \text{if } u = v \end{cases} \tag{16}$$

The first step of MDS is to calculate a matrix T whose element with row r and column c is given by $T(r,c) = -\frac{1}{2}[S(r,c) - \hat{S}(r,.) - \hat{S}(.,c) + \hat{S}(.,.)]$, where $\hat{S}(r,.) = \frac{1}{|V|}\sum_{c=1}^{|V|} S(r,c)$ is the average dissimilarity value over the rth row, $\hat{S}(.,c)$ is the dissimilarly defined average value over the cth column and $\hat{S}(.,.) = \frac{1}{|V|^2}\sum_{r=1}^{|V|}\sum_{c=1}^{|V|} S(r,c)$ is the average dissimilarity value over all rows and columns of the dissimilarity matrix T.

We subject the matrix T to an eigenvector analysis to obtain a matrix of embedding co-ordinates X. If the rank of T is $k, k \leq |V|$, then we will have k non-zero eigenvalues. We arrange these k non-zero eigenvalues in descending order, i.e. $l_1 \geq l_2 \geq \cdots \geq l_k > 0$. The ith ordered eigenvector is denoted by $\boldsymbol{f}_i$. The matrix with the node embedding co-ordinates as rows has the ordered scaled eigenvectors as columns and is given by $X = [\sqrt{l_1}\boldsymbol{f}_1|\sqrt{l_2}\boldsymbol{f}_2|\ldots|\sqrt{l_s}\boldsymbol{f}_s]$. For the graph node indexed u, the embedded vector of co-ordinates is given by the uth row of the matrix, i.e. $\boldsymbol{x}_u = (X_{u,1}, X_{u,2}, ..., X_{u,s})^T$.

Of course this embedding procedure can be applied to the geodesic distances instead of the Euclidean distances. However, this is inconsistent with the Euclidean multidimensional scaling procedure that we have adopted. As we will demonstrate later, the use of geodesic distances leads to much poorer results.

4 Clustering

Once the nodes of a graph have been embedded, we can attempt to characterise the structure of the graph by summarising the distribution of points associated with the nodes. Here we explore the two alternatives of using statistical moments and a graph-spectral characterisation.

4.1 Statistical Moments

We use spatial moments to characterise the embedded point sets. The general moment is defined to be

$$\mu_{pq} = \sum_{u=1}^{|V|} \sum_{v=1}^{|V|} (X_{u,1} - \hat{X}_1)^p (X_{v,2} - \hat{X}_2)^q \tag{17}$$

where $\hat{X}_k = \frac{1}{|V|} \sum_{u=1}^{|V|} X_{u,k}$. ¿From the raw moment data, we compute the four affine invariants suggested by Flusser and Suk [23]:

$$I_1 = \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4} \tag{18}$$

$$I_2 = \frac{(\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} - 3\mu_{21}^2\mu_{12}^2)}{\mu_{00}^{10}} \tag{19}$$

$$I_3 = \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu 11(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7} \tag{20}$$

$$\begin{aligned} I_4 = (&\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} \\ &+ 9\mu_{20}^2\mu_{02}\mu_{12}^2 + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} \\ &- 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} - 8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} \\ &+ 9\mu_{20}\mu_{02}^2\mu_{21}^2 + 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} \\ &+ \mu_{02}^3\mu_{30}^2)/\mu_{00}^{11} \end{aligned} \tag{21}$$

The zero-order moment μ_{00} measures area. The second-order moment is akin to the moment of inertia of the object about its centre-of-gravity. The third-order measures the degree of symmetry. The four moment invariants are used to compute the graph feature vector $\boldsymbol{B} = (I_1, I_2, I_3, I_4)^T$.

4.2 Spectral Characterisation

To construct a spectral characterisation of the embedded points, we commence by computing a weighted proximity matrix W with elements

$$W_E(u,v) = \begin{cases} \exp[\frac{-\|\boldsymbol{x}_u - x_v\|_2^2}{2\sigma^2}] & \text{if } \|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2 < r \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where σ is a scale constant and r is the radius of a neigbourhood hypersphere in the embedding space. From the weighted proximity matrix we compute the Laplacian matrix $L_E = W_E - D_E$ where D_E is diagonal degree matrix with elements $DE(u,u) = \sum_{v \in V} W(u,v)$. The spectral decomposition of the Laplacian matrix is $L_E = \sum_{i=1}^n \lambda_i^E \boldsymbol{e}_i \boldsymbol{e}_i{}^T$, where λ_i^E is the ith eigenvalue and $\boldsymbol{e}_i$ is the corresponding eigenvector of the Laplacian matrix L_E. Our spectral characterisation of the graph is based on the vector of N leading Laplacian eigenvalues $\boldsymbol{B} = (\lambda_1^E, ..., \lambda_N^E)^T$.

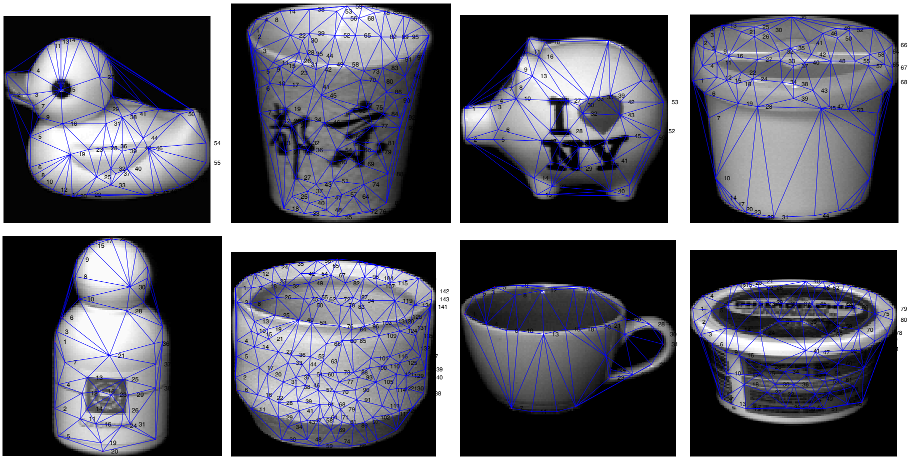


Fig. 1. Eight objects with their Delaunay graphs overlayed

4.3 Principal Components Analysis

Our aim is to construct a pattern-space for a set of graphs with pattern vectors $\boldsymbol{B}_k$, $k = 1, ..., M$, extracted using either spatial moments, heat-content polynomial co-efficents, sectional curvature histograms or Laplacian eigenvalues. There are a number of ways in which the graph pattern vectors can be analysed. Here, for the sake of simplicity, we use principal components analysis (PCA). We commence by constructing the matrix $\mathbf{S} = [\boldsymbol{B}_1|\boldsymbol{B}_2|\dots|\boldsymbol{B}_k|\dots|\boldsymbol{B}_M]$ with the graph feature vectors as columns. Next, we compute the covariance matrix for the elements of the feature vectors by taking the matrix product $\mathbf{C} = \mathbf{S}\mathbf{S}^T$. We extract the principal components directions by performing the eigendecomposition $\mathbf{C} = \sum_{i=1}^{M} l_i \boldsymbol{u}_i \boldsymbol{u}_i^T$ on the covariance matrix $\mathbf{C}$, where the l_i are the eigenvalues and the $\boldsymbol{u}_i$ are the eigenvectors. We use the first s leading eigenvectors (3 in practice for visualisation purposes) to represent the graphs extracted from the images. The co-ordinate system of the eigen-space is spanned by the s orthogonal vectors $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, .., \boldsymbol{u}_s)$. The individual graphs represented by the vectors $\mathbf{B}_k, k = 1, 2, \dots, M$ can be projected onto this eigenspace using the formula $\boldsymbol{\mathcal{B}}_k = \mathbf{U}^T \boldsymbol{B}_k$. Hence each graph G_k is represented by an s-component vector $\boldsymbol{\mathcal{B}}_k$ in the eigen-space.

5 Experiments

We have applied our embedding method to images from the COIL data-base. The data-base contains views of 3D objects under controlled viewer and lighting conditions. For each object in the data-base there are 72 equally spaced views, which are obtained as the camera circumscribes the object. We study the images from eight example objects. A sample view of each object is shown in Figure 1.

For each image of each object we extract feature points using the method of [24]. We have extracted graphs from the images by computing the Voronoi tessellations of the feature-points, and constructing the region adjacency graph, i.e. the Delaunay triangulation, of the Voronoi regions. Our embedding procedure has been applied to the resulting graph structures.

We commence by investigating the behavior of the moments extracted from the embedded points. In Figure 2 we plot the four moments as a function of the time parameter t.

For this experiment we have used a graph for the duck in Figure 1. The main effect to note here is that as the time parameter increases then so the four moments become small and indistinguishable. In Figure 3 we plot the four moments separately as a function of the view number for the images of the eight objects studied in the COIL data-base. From the top-left, and clockwise, the sequence shows the first, second, third and fourth moments respectively.

The individual moments appear relatively stable with view number. However, there are views where the moment jumps significantly, and this can be attributed to poor segmentation of the associated feature points from which the graphs are constructed. It is clear that although the individual moments could not be used to distinguish the objects since their traces with view number overlap, when combined they are sufficient to distinguish the objects.

Based on this study of the moments, it appears that they may provide the basis for a stable clustering of the graphs. We have therefore performed PCA on vectors whose components are the four moments. The data has been projected onto the space spanned by the leading three eigenvectors for the moment-vectors.

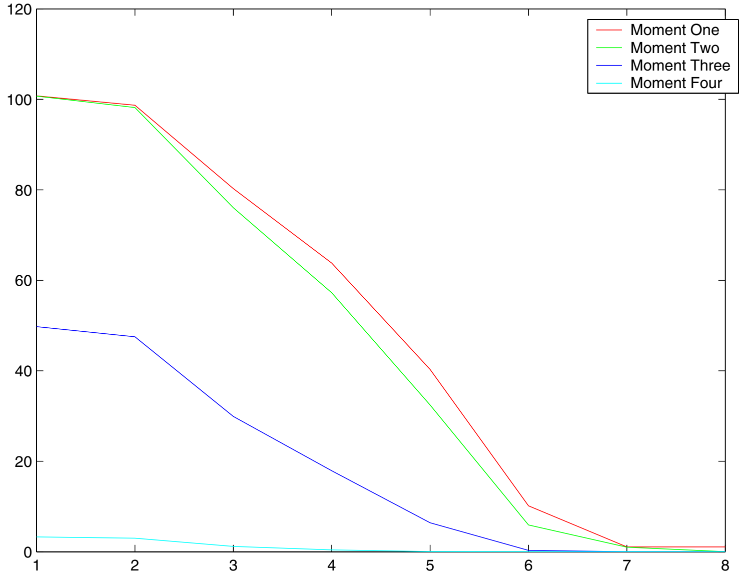


Fig. 2. Moments as a function of t for a graph from the COIL database

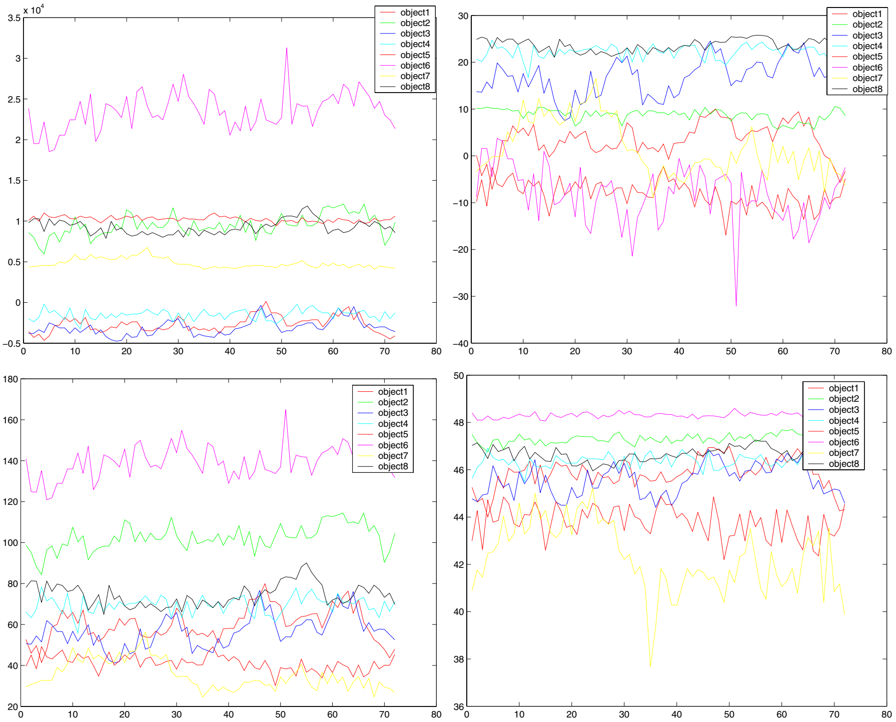


Fig. 3. Individual moments for graphs from the COIL database as a function of view number

We then investigate the effect of varying the time parameter t. In Figure 4 we show the effect on the graph embeddings when we vary t from 0.01π to 10000π. From left-to-right and top-to-bottom, we show the clustering results obtained when t equals 0.01π,1π, 4π, 10π, 100π and 10000π. In the figures the different views of the same object are displayed as differently colored symbols. In Figures 5 we show corresponding plots for the pairwise distances for the embedded graph nodes. The main feature to note from these plots is that as the value of t increases, then so the clusters corresponding to the different objects become overlapped. When the embedding procedure is applied to the geodesic distance d_G, found from the floor of the function $P^k(u, v)$, then the results shown in Figure 6 are obtained. Little of the cluster-structure emerges and the result is much poorer than that obtained with Euclidean distance at low values of t.

For comparison, Figure 7 shows the corresponding result when spectral clustering is used. Here we use the leading eigenvalues of the adjacency matrix as the components of a feature vector. The main qualitative feature is that the different views of the ten objects are more overlapped than when the heat-kernel method is used with a low value of t.

To investigate the behavior of the three methods in a more quantitative way, we have plotted the Rand index [25] for the different objects. The Rand index

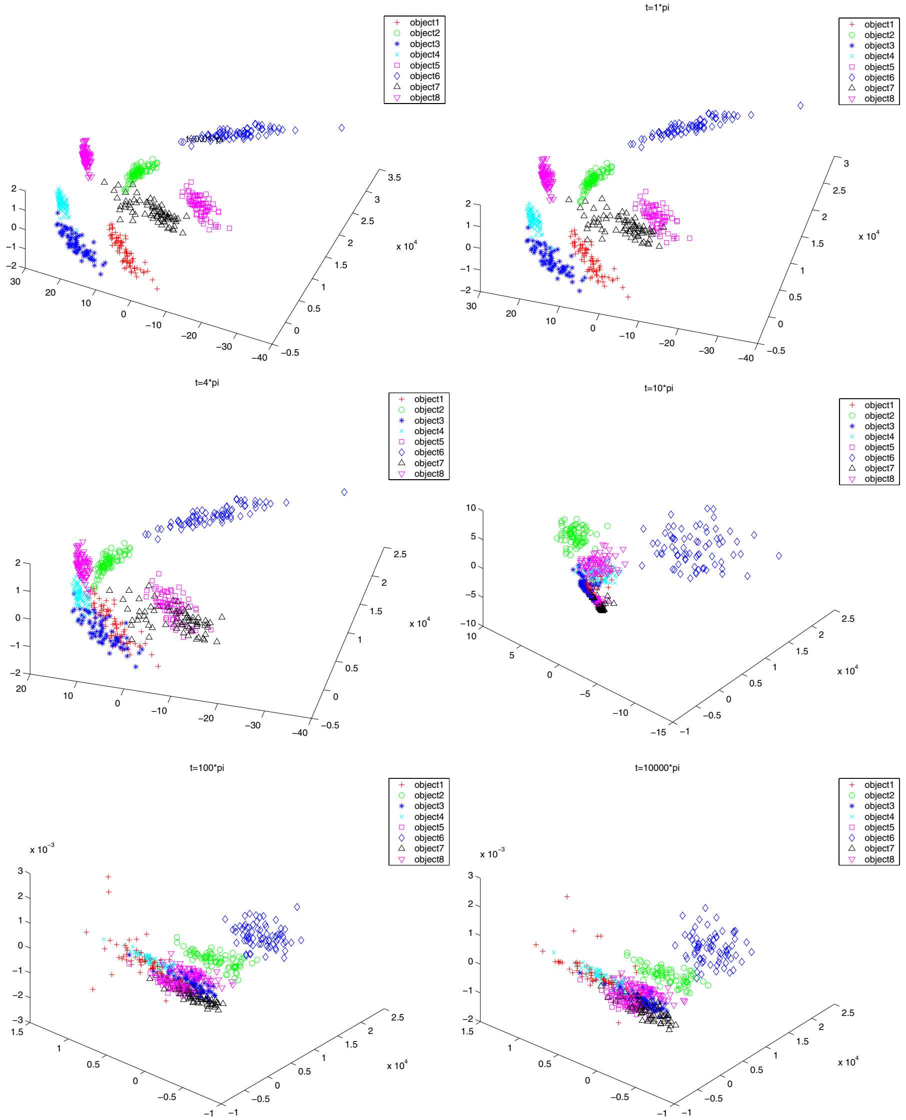
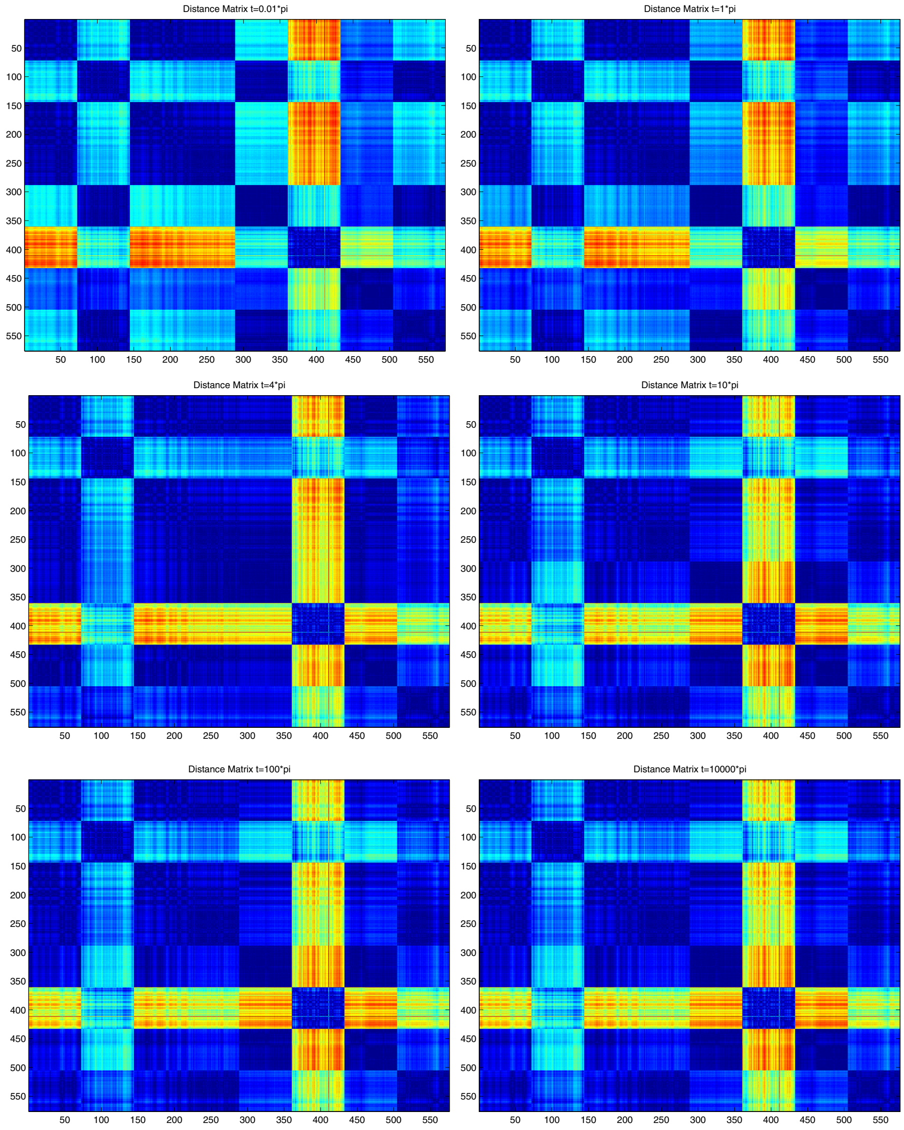


Fig. 4. Results of applying PCA to the point-moments of the embedded graphs with varying t

is defined as: $R_I = \frac{A}{A+D}$ where A is the number of "agreements" and D is the number of "disagreements" in cluster assignment. The index is hence the fraction of views of a particular class that are closer to an object of the same class than to one of another class. The curve marked with asterisks in the plot shows the Rand index as a function of t for the spatial moments. The performance of the method drops off rapidly once t exceeds 2π. The Rand index for the spectral


Distance Matrix t=0.01*pi

Distance Matrix t=1*pi

Distance Matrix t=4*pi

Distance Matrix t=10*pi

Distance Matrix t=100*pi

Distance Matrix t=10000*pi

Fig. 5. Distance matrices from the moments for the embedded graphs with varying t

clustering method is shown as a line marked with diamonds and that for the heat-content invariants as a black line. The performance of the heat-content invariants is slightly poorer and the spectral method significantly poorer than that for the heat-kernel method for small values of t.

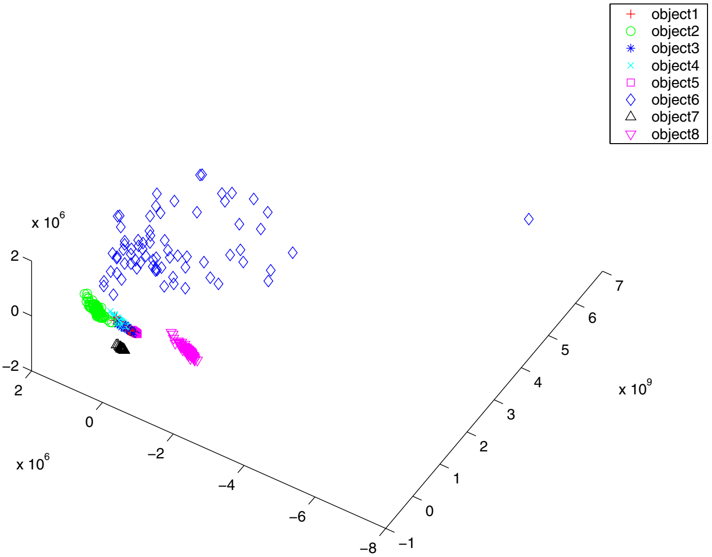


Fig. 6. Result of applying PCA to the affine moments obtained when the embedding is performed using geodesic distances

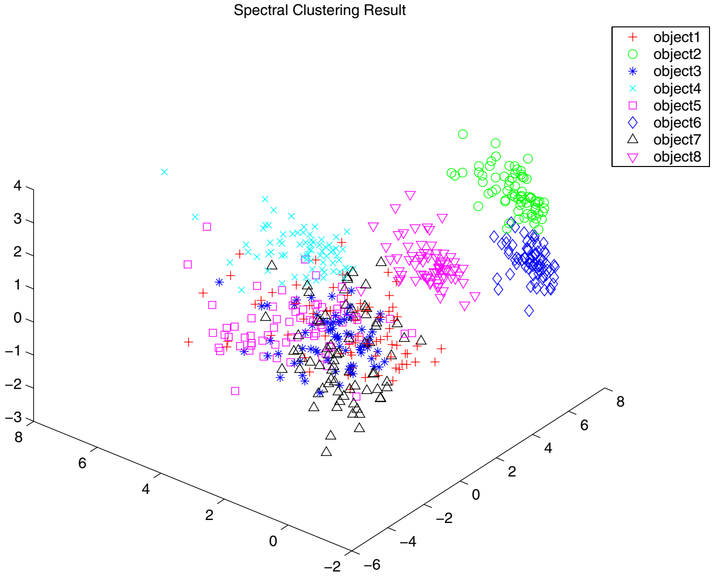


Fig. 7. Result of applying PCA to the leading Laplacian eigenvalues

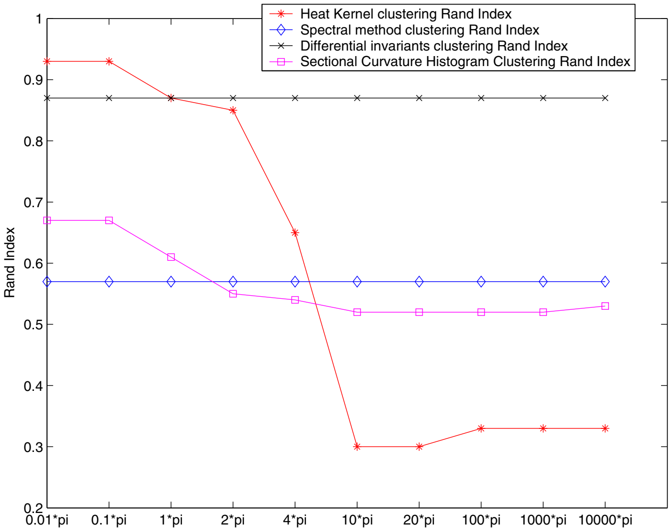


Fig. 8. Rand index for the different clustering methods

6 Conclusions and Future Work

In this paper we have explored how the use of heat kernels can lead to a measure of geodesic distance that can be used for the purposes of embedding graphs in low dimensional Euclidean spaces. The distance measure is found by equating the spectral and Gaussian forms of the heat kernel. We show how MDS can be used to embed the the distances.

Once the graphs are embedded in this space, then we can pose the problem of clustering as that of characterising the point-set distribution. We have shown how graph-clustering can be successfully effected using spatial moments.

There are clearly a number of ways in which the work reported in this paper can be extended. First, it would be interesting to explore the use of the sectional curvature as a means of directly embedding the nodes of the graphs on a manifold. One of the possibilities that exists here is the variant of MDS reported by Lindman and Caelli [26]. A second line of investigation would be the Euclidean distances or sectional curvature associated with the edges as attributes for the purposes of graph matching. Finally, it would be interesting to investigate if the distances and curvatures could be used to aid the process of visualising or drawing graphs.

References

1. N.Linial, E.London, Y.Rabinovich: The geometry of graphs and some its algorithmic application. Combinatorica **15** (1995) 215–245
2. A.D.Alexandrov, V.A.Zalgaller: Intrinsic geometry of surfaces. Transl. Math. Monographs **15** (1967)

3. H.Busemann: The geometry of geodesics. Academic Press (1955)
4. S.Weinberger: Review of algebraic l-theory and topological manifolds by a.ranicki. BAMS **33** (1996) 93–99
5. A.Ranicki: Algebraic l-theory and topological manifolds. Cambridge University Press (1992)
6. J.B.Tenenbaum, V.D.Silva, J.C.Langford: A global geometric framework for nonlinear dimensionality reduction. Science **290** (2000) 586–591
7. S.T.Roweis, L.K.Saul: Nonlinear dimensionality reduction by locally linear embedding. Science **290** (2000) 2323–2326
8. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. Neural Information Processing Systems **14** (2002) 634–640
9. F.R.K.Chung: Spectral graph theory. American Mathematical Society (1997)
10. Atkins, J.E., Bowman, E.G., Hendrickson, B.: A spectral algorithm for seriation and the consecutive ones problem. SIAM J. Comput. **28** (1998) 297310
11. A.Shokoufandeh, S.Dickinson, K.Siddiqi, S.Zucker: Indexing using a spectral encoding of topological structure. IEEE Conf. on Computer Vision and Pattern Recognition (1999) 491–497
12. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE PAMI **22** (2000) 888–905
13. S.Umeyama: An eigen decomposition approach to weighted graph matching problems. IEEE PAMI **10** (1988) 695–703
14. Luo, B., Hancock, E.: Structural graph matching using the em algorithm and singular value decomposition. IEEE PAMI **23** (2001) 1120–1136
15. S.T.Yau, R.M.Schoen: Differential geometry. Science Publication (1988)
16. Gilkey, P.B.: Invariance theory, the heat equation, and the atiyah-singer index theorem. Publish or Perish Inc. (1984)
17. A.Grigor'yan: Heat kernels on manifolds, graphs and fractals. European Congress of Mathematics: Barcelona **I** (2001) 393–406
18. Lafferty, J., Lebanon, G.: Diffusion kernels on statistical manifolds. Technical Report CMU-CS-04-101 (2004)
19. M.T.Barlow: Diffusions on fractals. Lectures on Probability Theory and Statistics **Lecture Notes Math. 1690** (1998) 1–121
20. Smola, A.J., Bartlett, P.L., Schlkopf, B., Schuurmans, D.: Advances in large margin classifiers. MIT Press, Cambridge, MA **354** (2000) 5111–5136
21. de Verdiere, C.: Spectra of graphs. Math of France **4** (1998)
22. S.Rosenberg: The laplacian on a Riemannian manifold. Cambridge University Press (2002)
23. J.Flusser, T.Suk: Pattern recognition by affine moment invariants. Pattern Recognition **26** (1993) 167–174
24. Harris, C., Stephens, M.: A combined corner and edge detector. Fourth Alvey Vision Conference (1994) 147–151
25. W.M.Rand: Objective criteria for the evaluation of clustering. Journal of the American Statistical Association **66** (1971) 846–850
26. H.Lindman, T.Caelli: Constant curvature Riemannian scaling. Journal of Mathematical Psychology **17** (1978) 89–109

Detection of Surface Creases in Range Data

Alexander Belyaev and Elena Anoshkina

Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
belyaev@mpi-sb.mpg.de

Abstract. We propose a fully automatic and view-independent computational procedure for detecting salient curvature extrema in range data. Our method consists of two major steps: (1) smoothing given range data by applying a nonlinear diffusion of normals with automatic thresholding; (2) using a Canny-like non-maximum suppression and hysteresis thresholding operations for detecting crease pixels. A delicate analysis of curvature extrema properties allows us to make those Canny-like image processing operations orientation-independent. The detected patterns of creases can be considered as 'shape fingerprints'. The proposed method can be potentially used for shape recognition, quality evaluation, and matching purposes.

1 Introduction

Shape features invariant under 3D rigid motions and scalings are important for many 3D shape processing operations including shape recognition, matching, and segmentation. Recent advances in 3D shape acquisition technology call for developing reliable methods for range data processing. In this paper, we propose a computational scheme for robust detection of view- and scale-invariant surface creases, ridges and ravines, in range data.

Surface creases defined via principal curvature extrema are view- and scale-independent and, therefore, have numerous applications in geometric modeling, image processing and other fields. They have been intensively studied in connection with research on the accommodation of the eye lens [1], structural geology [2], human perception [3], CAGD and computer graphics [4, 5, 6, 7, 8, 9, 10, 11, 12], face recognition [13], image analysis [14, 15, 16, 17], machine vision [18], geographical information systems [19], computational anatomy [20], and classical differential geometry and singularity theory [13, 21, 22, 23, 24, 25, 26, 27, 28]. See also references therein.

We define the ridges on a smooth surface as the locus of points where the maximal principal curvature takes a positive maximum along its curvature line and the ravines as the locus of points where the minimal principal curvature attains a negative minimum along its curvature line.

Fig. 1 shows the ridges and ravines detected in a range data with complex geometry.

The above definition of the ridges and ravines resembles a widely used definition for edges in image processing: the edges consists of pixels where the

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 50–61, 2005.

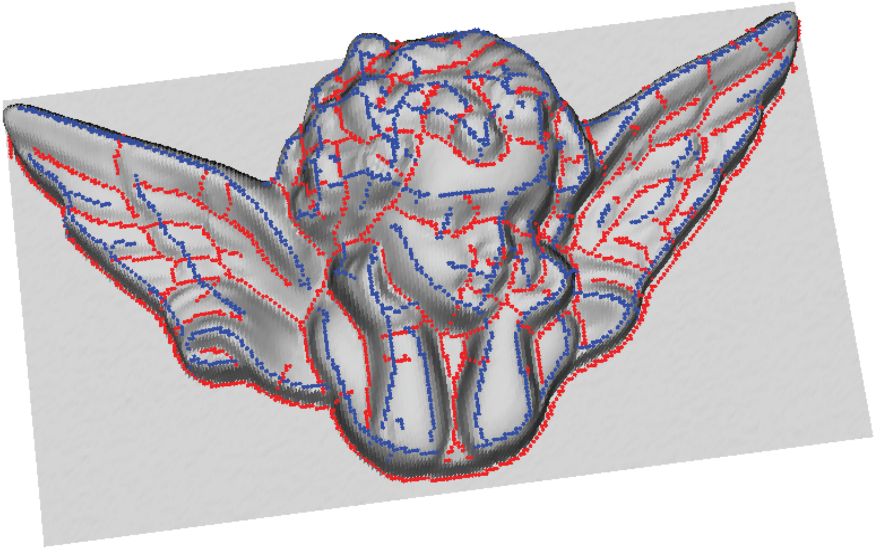


Fig. 1. Ridges (blue) and ravines (red) detected in 'angelo' range data

magnitude of the gradient of the image intensity has a local maximum in the direction of the gradient (see, for example, [29]).

The ridges and ravines defined as above are perceptually salient [3], are invariant under rotations, translations, and scalings, and correspond to the skeletal edges of the shape skeletons (defined locally) [13, 21, 26, 28]. Note also that the ridges and ravines are dual according to the above definition: changing the surface orientation turns the ridges into the ravines and vice versa.

Since computing of the ridges and ravines involves estimation of high-order surface derivatives, these surface features are very sensitive to noise and a sophisticated smoothing procedure is required in order to achieve robust detection of the ridges and ravines on shapes reconstructed from real world data.

Our method is based on a delicate analysis of curvature extrema (Section 2) and consists of of two steps: adaptive smoothing (Section 4) and detection (Section 3). The main ingredient of our range data smoothing scheme is a nonlinear diffusion of normals [30, 31, 32]. The main advantage of the scheme over these and similar techniques consists of using no user-specified parameters except the number of smoothing iterations.

We want to emphasize here that the ridges and ravines studied in this paper are not the best choice for range image segmentation purposes. Several range image segmentation algorithms proposed recently demonstrate superior performance in segmenting noisy range data consisted of a collection of geometrically simple objects [33, 34]. Moreover, as shown in [21], the ridges and ravines on a generic smooth surface do not deliver a surface segmentation. In spite of a complex smoothing procedure we use, our ridge and ravine detection scheme remains sensitive to noise and produces satisfactory results while processing relatively clean data like range images generated by laser scanner devices. Nevertheless we envision potential applications of our method to shape recognition, quality evaluation, and matching purposes.

2 Local Shape Analysis

Since the ridges and ravines turn into each other as the surface orientation is changed, without loss of generality we can consider only the ridges.

Consider a smooth surface and a non-umbilic point $\mathbf{p}$ on it. Let us choose local coordinates such that $\mathbf{p}$ is at the origin, the (x, y)-plane is the tangent plane to the surface at $\mathbf{p}$, the principal directions $\mathbf{t}_{\max}$ and $\mathbf{t}_{\min}$ coincide with x and y axes, respectively, and the normal $\mathbf{n}$ coincides with z-axis. Then in a small vicinity of $\mathbf{p}$ the surface is represented in the Monge form as the graph of a generic smooth function $z = F(x, y)$ such that

$$\begin{aligned} F(x,y) = \frac{1}{2}\left(\lambda x^2 + \mu y^2\right) + \frac{1}{6}\left(ax^3 + 3bx^2y + 3cxy^2 + dy^3\right) \\ + \frac{1}{24}\left(ex^4 + 4fx^3y + \dots\right) + O(x,y)^5 \end{aligned}$$

with $\lambda = k_{\max}(0,0)$, $\mu = k_{\min}(0,0)$, and $\lambda > \mu$. Assume that the surface orientation be chosen so that the maximal principal curvature is nonnegative at $\mathbf{p}$: $\lambda \geq 0$.

Since at $\mathbf{p}$ the principal directions $\mathbf{t}_{\max}$ and $\mathbf{t}_{\min}$ are given by vectors $(1, 0)$ and $(0, 1)$, respectively, then

$$\frac{\partial k_{\max}}{\partial \mathbf{t}_{\max}}(0,0) = a \quad \text{and} \quad \frac{\partial k_{\max}}{\partial \mathbf{t}_{\min}}(0,0) = b. \tag{1}$$

The curvature line associated with $k_{\max}$ is locally described by the problem

$$\frac{dy}{dx} = \frac{bx + cy}{\lambda - \mu} + O(x,y)^2, \qquad y(0) = 0.$$

Therefore, $y'(0) = 0$, $y''(0) = b/(\lambda - \mu)$ and in a neighborhood of the origin the curvature line is locally given by

$$y = \frac{bx^2}{2(\lambda - \mu)} + O(x^3).$$

It allows us to compute the Taylor series expansion of $k_{\max}$ on its curvature line

$$\lambda + ax + \left(-3\lambda^3 + e + \frac{3b^2}{\lambda - \mu}\right)\frac{x^2}{2} + O(x^3). \tag{2}$$

Thus $k_{\max}$ has a positive maximum along its curvature line if and only if

$$\lambda > 0, \quad a = 0, \quad A = -3\lambda^3 + e + \frac{3b^2}{\lambda - \mu} < 0. \tag{3}$$

Note that from (3) it follows that the ridges do not pass through generic umbilics [21]. Indeed, assume that a ridge goes through an isolated umbilical point. Passing to the limit $\lambda - \mu \to 0$ in (3) implies $a = 0 = b$ and, therefore, the umbilic is not generic.

Of course, surface curves consisting of some other types of extrema of the principal curvatures along their curvature lines can go through generic umbilics. Interesting relations between the curvature line patterns near a generic umbilic [35, 36] and extrema of the principal curvatures were analyzed in [13, 24, 27].

Consider the intersection curve between the surface and plane $\{y = \alpha z\}$, where α is a parameter. The curve is locally described by the equation

$$y = \alpha\lambda x^2/2 + \dots .$$

Let us consider the function

$$K(x,y) = k_{\max}(x,y) + \frac{C}{2}\left(x^2 + y^2 + F(x,y)^2\right) \tag{4}$$

obtained from the maximal principal curvature by adding a function proportional to the squared distance from the origin. The Taylor expansion of K on the intersection curve has the form

$$\lambda + ax + \left(-3\lambda^3 + e + \frac{2b^2}{\lambda - \mu} + \alpha b\lambda + C\right)\frac{x^2}{2} + O(x^3). \tag{5}$$

Now let us set

$$C = -\alpha b\lambda + b^2/(\lambda - \mu). \tag{6}$$

It follows from (2) and (5) that the ridge points can be detected as the positive maxima of $K(x,y)$ at $\mathbf{p}$ along the intersection curve between the surface and plane $\{y = \alpha z\}$.

Theorem 1. *Let $K(x,y)$ be defined by (4) and (6). Then $\mathbf{p}$ is a ridge point if and only if K has a positive maximum at $\mathbf{p}$ along the intersection curve between the surface and plane $\{y = \alpha z\}$.*

3 Practical Detection of Ridges and Ravines

Consider a range data (depth field) defined by a height function $z = R(x,y)$. Let $k_{\max}$ and $k_{\min}$ ($k_{\max} > k_{\min}$) be the principal curvatures of the surface $z = R(x,y)$, $\mathbf{n}(x,y)$ be the downward unit normal, and $\mathbf{v}_{\max}$ and $\mathbf{v}_{\min}$ be the projections of the associated principal directions $\mathbf{t}_{\max}$ and $\mathbf{t}_{\min}$ onto the xy-plane.

In order to decide whether $k_{\max}$ takes a maximum along the intersection curve between the surface and vertical plane $\langle \mathbf{v}_{\max}, \mathbf{n} \rangle$ at a grid vertex $\mathbf{p}$ we can adopt a standard technique used for edge detection [29], see Fig. 2. The principal curvature $k_{\max}$ at $\mathbf{q}_1$ and $\mathbf{q}_2$ can be estimated by bilinear interpolations between the values of $k_{\max}$ at $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$, $\mathbf{p}_4$, respectively.

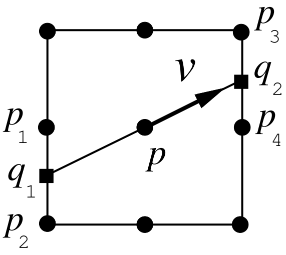


Fig. 2. The principal curvatures at $\mathbf{q}_1$ and $\mathbf{q}_2$ are estimated by bilinear interpolations between the curvature values at $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$, $\mathbf{p}_4$, respectively

A simple approach taken in [37] consists of comparing $k_{\max}(\mathbf{p})$ with $k_{\max}(\mathbf{q}_1)$ and $k_{\max}(\mathbf{q}_2)$ and determines the ridges' vertices as the points where $k_{\max}$ attains a positive maximum in the direction of $\mathbf{v}_{\max}$. However our goal here is to test at $\mathbf{p}$ whether $k_{\max}$ attains a positive maximum on the curvature line corresponding to $\mathbf{t}_{\max}$.

Let $\mathbf{e}_z = (0, 0, 1)$ be the unit vector defining vertical direction z. For a surface point $\mathbf{p}$ let us define α such that

$$\mathbf{n} \cdot \mathbf{e}_z = \frac{1}{\sqrt{1+\alpha^2}}.$$

As shown in the previous section, in order to determine whether $k_{\max}$ takes a maximum on the curvature line corresponding to $\mathbf{t}_{\max}$ we have to compare $K(\mathbf{p}) = k_{\max}(\mathbf{p})$ with $K(\mathbf{q}_1)$ and $K(\mathbf{q}_2)$, where

$$K(\mathbf{q}) = k_{\max}(\mathbf{q}) + \frac{C}{2}\left(|\mathbf{p}-\mathbf{q}|^2 + |R(\mathbf{p}) - R(\mathbf{q})|^2\right). \tag{7}$$

Here

$$C = -\alpha\, k_{\max}(\mathbf{p}) \frac{\partial k_{\max}}{\partial \mathbf{t}_{\min}}(\mathbf{p}) + \frac{1}{k_{\max}(\mathbf{p}) - k_{\min}(\mathbf{p})} \left[\frac{\partial k_{\max}}{\partial \mathbf{t}_{\min}}(\mathbf{p})\right]^2.$$

To estimate the derivative of $k_{\max}$ along $\mathbf{t}_{\min}$ at $\mathbf{p}$ we set $\mathbf{v} = \mathbf{v}_{\min}$ in Fig. 2, employ the bilinear interpolation to estimate $k_{\max}(\mathbf{q}_1)$ and $k_{\max}(\mathbf{q}_2)$, and then use a standard finite difference scheme with three points $\mathbf{q}_1$, $\mathbf{p}$, and $\mathbf{q}_2$.

Now let us set $\mathbf{v} = \mathbf{v}_{\max}$ in Fig. 2. We mark grid vertex $\mathbf{p}$ as a ridge vertex if

$$K(\mathbf{p}) > K(\mathbf{q}_1), \quad K(\mathbf{p}) > K(\mathbf{q}_2), \quad \text{and} \quad k_{\max}(\mathbf{p}) > T,$$

where T is a positive threshold.

We use hysteresis thresholding [38] to remove unessential ridge and ravine vertices and keep the ridge and ravine vertices connected as much as possible. Consider the set $\mathcal{R}$ of grid vertices where $k_{\max} > 0$. Let us call the value of $k_{\max}$ at a vertex from $\mathcal{R}$ the ridge-strength of that vertex. We choose two thresholds T_{lo} and T_{hi}, $T_{\text{hi}} > T_{\text{lo}} > 0$, at the 30th and 60th percentiles of the ridge-strength

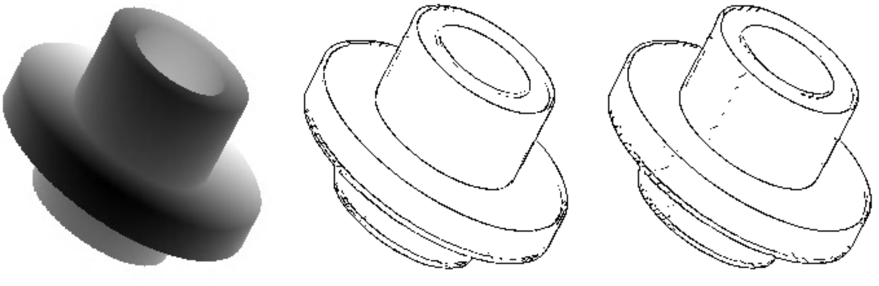


Fig. 3. Left: a synthetic range image representing a surface with sharp edges. Middle: sharp edges (ridges and ravines) are detected using $K(\mathbf{q})$. Right: $k_{\max}(\mathbf{q})$ is used instead of $K(\mathbf{q})$. For both the results the same thresholds and number of smoothing iterations were used

data for $\mathcal{R}$, i.e., the 30th percentile value is chosen so that for 30 percent of the vertices from $\mathcal{R}$, $k_{\max}$ is below that value. We keep a chain of connected ridge vertices with $k_{\max} > T_{\text{lo}}$ if $k_{\max} > T_{\text{hi}}$ for at least one grid vertex in the chain.

Surprisingly, although computing $K(\mathbf{q})$ according to (7) involves an estimation of third-order surface derivatives, using $K(\mathbf{q})$ instead of $k_{\max}(\mathbf{q})$ leads not only to a coordinate-independent computational procedure but also often provides with a better detection of sharp features, as seen in Fig. 3.

Let us point out again that our goal here does not consist of processing geometrically simple shapes with sharp edges like those shown in Fig. 3. Segmenting such shapes can be effectively done by methods discussed in [33, 34], see also references therein.

4 Smoothing Range Data

As we noted before, a robust detection of the ridges and ravines requires a sophisticated smoothing procedure. In this section, we develop an automatic smoothing procedure based on a nonlinear diffusion of the range data normals. Fig. 4 demonstrates how important preliminary smoothing is.

Smoothing by repeated local averaging (diffusion) is a popular image processing technique. Consider a grey-scale image given by its intensity function $z = R(x, y)$. Image smoothing by repeated local averaging has the form

$$I(x, y, n+1) = \frac{\sum I(x+i, y+j, n) w(x+i, y+j, n)}{\sum w(x+i, y+j, n)}, \tag{8}$$

where $I(x, y, 0) = I(x, y)$, $\{w(x, y, n)\}$ are positive weights which may vary from pixel to pixel and change during the iterative averaging procedure (8), and the summation is taken over a local neighborhood of (x, y) pixel.

If all weights $\{w\}$ in (8) are equal, the image is blurred and image edges, sets of pixels where the image gradient is large, are destroyed. If the weights in (8)

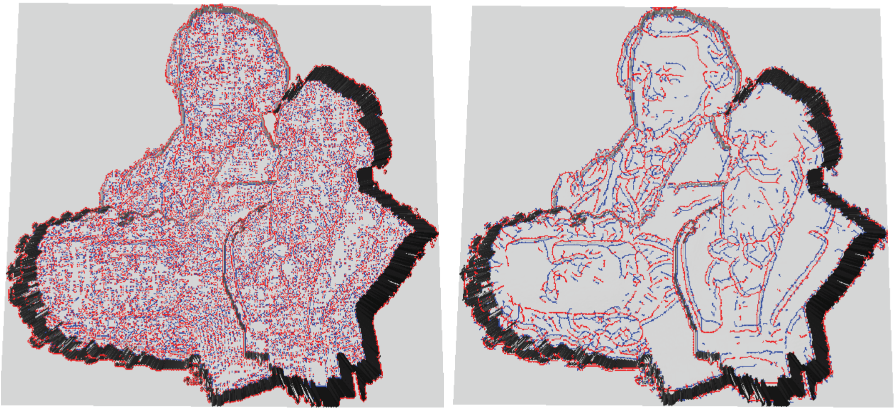


Fig. 4. Ridges (blue) and ravines (red) detected on a complex range image. Left: no preliminary smoothing was done. Right: nonlinear diffusion of normals was used for preliminary smoothing

are chosen adaptively: they are small at those pixels where the image gradient is large, then (8) takes into account only pixels with nearly the same intensity values and edges are not destroyed.

This idea of adaptive image averaging was developed by Saint-Marc and Medioni [39] and can be considered as a simplification of the Perona and Malik nonlinear diffusion approach [40]. Saint-Marc and Medioni [39] proposed also to apply adaptive smoothing to the image derivatives in order to achieve a robust detection of curvature features in range images. Unfortunately Saint-Marc and Medioni filtering schemes are not invariant under 3D rotations and, therefore, are not appropriate for the detection of rotation-invariant features.

The smoothing procedure presented below adapts the Saint-Marc and Medioni approach to smoothing the field of surface normals. In contrast to similar schemes [41] our procedure involves no user-specified parameters except the number of smoothing iterations.

The downward unit normal of $z = R(x, y)$ is given by

$$\mathbf{n}(x, y) = [n_1, n_2, n_3] = \frac{[R_x(x, y), R_y(x, y), -1]}{\sqrt{1 + R_x(x, y)^2 + R_y(x, y)^2}}.$$

Notice that the partial derivatives of range data $z = R(x, y)$ are recovered from the field of normals

$$R_x(x, y) = -\frac{n_1(x, y)}{n_3(x, y)} \quad \text{and} \quad R_y(x, y) = -\frac{n_2(x, y)}{n_3(x, y)}.$$

Let $P(x, y, n)$ and $Q(x, y, n)$ approximate the range data derivatives $R_x(x, y)$, $R_y(x, y)$ after n smoothing iterations. The Gaussian and mean curvatures can be approximated by

$$K_G = \frac{P_x Q_y - P_y Q_x}{(1+P^2+Q^2)^2}, \tag{9}$$

$$K_m = \frac{(1+P^2)P_x - PQ(P_y+Q_x) + (1+Q^2)Q_y}{2(1+P^2+Q^2)^{3/2}}. \tag{10}$$

The smoothed principal curvatures are now given by

$$k_{\max} = K_m + \sqrt{K_m^2 - K_G} \quad \text{and} \quad k_{\min} = K_m - \sqrt{K_m^2 - K_G}.$$

For every grid vertex (x, y) let us define its weight by

$$w(x, y, n) = \exp\{-\frac{\sigma}{2}\left[k_{\max}^2(x, y, n) + k_{\min}^2(x, y, n)\right]\}.$$

Here $\sigma > 0$ controls which ridges are preserved and which are blurred.

In contrast to standard general nonlinear diffusion schemes [40, 39] we allow σ to vary from vertex to vertex: $\sigma = \sigma(x, y)$. To determine $\sigma(x, y)$ automatically for a grid vertex (x, y) we estimate the directional curvatures k_{ij} from (x, y) to its eight nearest neighbors $\{(x+i, y+j) : -1 \le i, j \le 1\}$. Let φ_{ij} be the angle between $\mathbf{n}(x+i, y+j)$ and $\mathbf{n}(x, y)$, we set

$$k_{ij} = \varphi_{ij}/d_{ij}, \quad d_{ij} = \sqrt{i^2 + j^2 + [R(x+i, y+j) - R(x, y)]^2}.$$

Here d_{ij} is the distance between to surface points $P_0 = [x, y, R(x, y)]$ and $P_{ij} = [x+i, y+j, R(x+i, y+j)]$ and k_{ij} estimates the directional curvature in the direction $P_0 P_{ij}$. We choose $\sigma(x, y)$ such that 20% of $\{|k_{ij}|\}$ are below $\sigma(x, y)$ and 80% are above. According to our experiments, the 20% threshold works well.

Consider the following discrete vector-valued diffusion process

$$\mathbf{m}(x, y, n+1) = \frac{\sum \mathbf{m}(x+i, y+j, n) w(x+i, y+j, n)}{\sum w(x+i, y+j, n)},$$
$$\mathbf{m}(x, y, 0) = \mathbf{n}(x, y),$$

where the summations are taken over a 3 by 3 square neighborhood of (x, y) grid vertex. The next approximations of the image derivatives are now computed by

$$P(x, y, n+1) = -m_1(x, y, n+1)/m_3(x, y, n+1),$$
$$Q(x, y, n+1) = -m_2(x, y, n+1)/m_3(x, y, n+1).$$

After a number of smoothing iterations, we estimate the Gaussian and mean curvatures via (9) and (10), respectively. Then the principal curvatures are computed. The coefficients of the first and second fundamental forms are estimated by

$$E = 1 + P^2, \qquad F = PQ \qquad G = 1 + Q^2,$$

$$L = \frac{P_x}{\sqrt{1+P^2+Q^2}}, \quad M = \frac{P_y + Q_x}{2\sqrt{1+P^2+Q^2}}, \quad N = \frac{Q_y}{\sqrt{1+P^2+Q^2}},$$

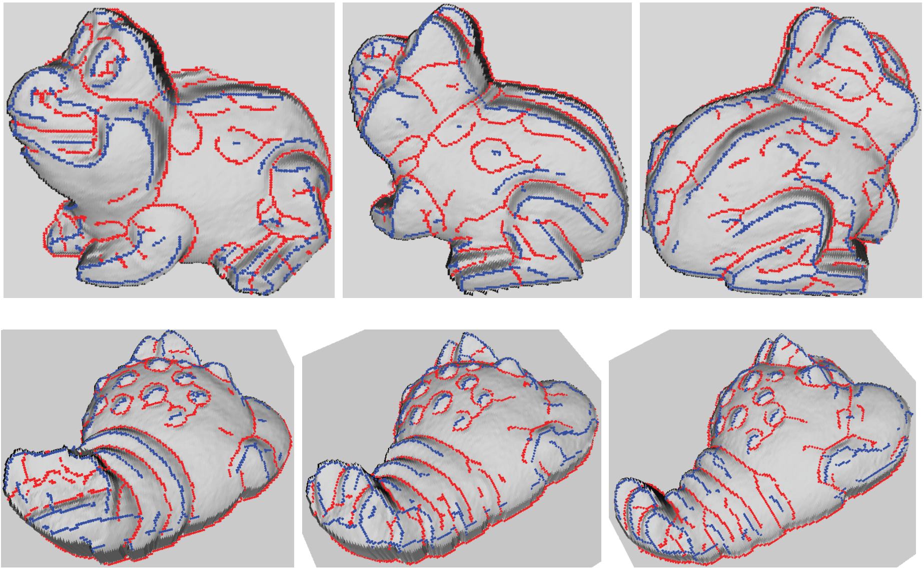


Fig. 5. Ridges and ravines detected on range images of a frog (top) and lobster (bottom) models. For each model, the images were made from three different positions

respectively, and computation of the projections $\mathbf{v}_{\max}$ and $\mathbf{v}_{\min}$ of the principal directions onto the image plane is straightforward.

According to our numerical experiments, five iterations produce a good smoothing effect. For all range images presented in this paper we use five iterations of the above smoothing procedure.

5 Discussion and Conclusion

In the paper, we have combined methods of classical differential geometry with modern image processing techniques and developed a coordinate-independent computational procedure for detecting salient extrema of the principal curvatures, the ridges and ravines, in range data.

Fig. 5 demonstrates the ridges and ravines detected on range data generated by generated by a laser scanner. Geometrically and perceptually important features of two complex 3D models are well detected. Also one can find small differences in the ridge and ravine patterns in range images made from different positions, we believe that the developed technique can be used for shape matching, quality evaluation, and recognition purposes.

Our method does not include post-processing filtering. The post-processing filtering schemes proposed in [7, 12] for improving connectivity of curvature extrema networks can be easily combined with the approach developed in this paper.

Similar to many other iterative image filtering schemes, our nonlinear image diffusion process may change positions of image features slightly. We think that after a proper modification, bilateral filtering [42, 43] and similar non-iterative adaptive image denoising techniques are capable to deliver better filtering results.

Acknowledgments. The range data used in this paper are courtesy of the Ohio State University and University of Bologna. We would like to thank the anonymous reviewers of this paper for their valuable and constructive comments. Thanks to Yutaka Ohtake for fruitful discussions. The research of the first author is supported in part by AIM@SHAPE, a Network of Excellence project (506766) within EU's Sixth Framework Programme.

References

1. Gullstrand, A.: Zur Kenntnis der Kreispunkte. Acta Mathematica **29** (1904) 59–100
2. Ramsay, J.G.: Folding and Fracturing of Rocks. McGraw Hill (1967)
3. Hoffman, D.D., Richards, W.A.: Parts of recognition. Cognition **18** (1985) 65–96
4. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. ACM Transactions on Graphics **22** (2003) 848–855 Proceedings of ACM SIGGRAPH 2003.
5. Hartmann, E.: On the curvature of curves and surfaces defined by normalforms. Comput. Aided Geom. Design **16** (1999) 355–376
6. Hosaka, M.: Modeling of Curves and Surfaces in CAD/CAM. Springer, Berlin (1992)
7. Ohtake, Y., Belyaev, A., Seidel, H.P.: Ridge-valley lines on meshes via implicit surface fitting. ACM Transactions on Graphics **23** (2004) 609–612 Proceedings of ACM SIGGRAPH 2004.
8. Patrikalakis, N.M., Maekawa, T.: Shape Interrogation for Computer Aided Design and Manufacturing. Springer, Heidelberg (2002)
9. Stylianou, G., Farin, G.: Crest lines extraction from 3D triangulated meshes. In Farin, G., Hamann, B., Hagen, H., eds.: Hierarchical and Geometrical Methods in Scientific Visualization, Springer (2003) 269–281
10. Stylianou, G., Farin, G.: Crest lines for surface segmentation and flattening. IEEE Transactions on Visualization and Computer Graphics **10** (2004) 536–544
11. Watanabe, K., Belyaev, A.G.: Detection of salient curvature features on polygonal surfaces. Computer Graphics Forum **20** (2001) 385–392 Eurographics 2001 issue.
12. Yoshizawa, S., Belyaev, A.G., Seidel, H.P.: Fast and robust detection of crest lines on meshes. In: ACM Symposium on Solid and Physical Modeling 2005, MIT, Cambridge, MA, USA (2005) To appear.
13. Hallinan, P.L., Gordon, G.G., Yuille, A.L., Giblin, P., Mumford, D.: Two- and Tree-Dimensional Patterns of the Face. A K Peters (1999)
14. Eberly, D., Gardner, R., Morse, D., Pizer, S., Scharlach, C.: Ridges for image analysis. J. Mathematical Imaging and Vision **4** (1994) 353–373
15. Monga, O., Armande, N., Montesinos, P.: Thin nets and crest lines: Application to satellite data and medical images. Computer Vision and Image Understanding: CVIU **67** (1997) 285–295

16. Monga, O., Ayache, N., Sander, P.T.: From voxel to intrinsic surface features. Image and Vision Computing **10** (1992) 403–417
17. Yuille, A.L.: Zero crossings on lines of curvature. Graphical Models and Image Processing **45** (1989) 68–87
18. Kent, J.T., Mardia, K.V., West, J.: Ridge curves and shape analysis. In: The British Machine Vision Conference 1996. (1996) 43–52
19. Little, J.J., Shi, P.: Structural lines, TINs and DEMs. Algorithmica, **30** (2001) 243–263
20. Pennec, X., Ayache, N., Thirion, J.P.: Landmark-based registration using features identified through differential geometry. In Bankman, I.N., ed.: Handbook of Medical Imaging. Academic Press (2000)
21. Belyaev, A.G., Anoshkina, E.V., Kunii, T.L.: Ridges, ravines, and singularities. In: *A. T. Fomenko, and T. L. Kunii*, Topological Modeling for Visualization. *Chapter 18*, Springer (1997) 375–383
22. Bruce, J.W., Giblin, P.J., Tari, F.: Ridges, crests and sub-parabolic lines of evolving surfaces. Int. J. Computer Vision **18** (1996) 195–210
23. Bruce, J.W., Giblin, P.J., Tari, F.: Families of surfaces: focal sets, ridges and umbilics. Math. Proc. Cambridge Philos. Soc. **125** (1999) 243–268
24. Cazals, F., Pouget, M.: Differential topology and geometry of smooth embedded surfaces: selected topics. Computational Geometry and Applications (2005) To appear.
25. Cazals, F., Pouget, M.: Topology driven algorithms for ridge extraction on meshes. Rapport de Recherche RR-5526, INRIA (2005)
26. Koenderink, J.J.: Solid Shape. MIT Press (1990)
27. Porteous, I.R.: Geometric Differentiation for the Intelligence of Curves and Surfaces. Cambridge University Press, Cambridge (2nd Edition, 2001)
28. Yuille, A.L., Leyton, M.: 3D symmetry-curvature duality theorems. Graphical Models and Image Processing **52** (1990) 124–140
29. Faugeras, O.: Three-Dimensional Computer Vision, *Ch. 4:* Edge Detection. MIT Press (1993)
30. Belyaev, A.G., Ohtake, Y., Abe, K.: Detection of ridges and ravines on range images and triangular meshes. In: Vision Geometry IX, Proc. SPIE 4117. (2000) 146–154
31. Ohtake, Y., Belyaev, A.G., Bogaevski, I.A.: Mesh regularization and adaptive smoothing. Computer-Aided Design **33** (2001) 789–800
32. Tasdizen, T., Whitaker, R., Burchard, P., Osher, S.: Geometric surface smoothing via anisotropic diffusion of normals. In: Proceedings of IEEE Visualization '02. (2002) 125–132
33. Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P., Bunke, H., Goldgof, D., Bowyer, K., Eggert, D., Fitzgibbon, A., Fisher, R.: An experimental comparison of range image segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (1996) 673–689
34. Lee, K.M., Meer, P., Park, R.H.: Robust adaptive segmentation of range images. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 200–205
35. Berry, M.V., Hannay, J.H.: Umbilic points on gaussian random surfaces. J. Phys. A **10** (1977) 1809–1821
36. Maekawa, T., Wolter, F.E., Patrikalakis, N.M.: Umbilics and lines of curvature for shape interrogation. Computer Aided Geometric Design **13** (1996) 133–161
37. Gordon, G.G.: Face recognition from depth maps and surface curvature. In: Geometric Methods in Computer Vision, Proc. SPIE 1570. (1991) 234–247

38. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986) 679–698
39. Saint-Marc, P., Chen, J.S., Medioni, G.: Adaptive smooting: A general tool for early vision. IEEE Transactions on Pattern Analysis and Machine Intelligence **13** (1991) 514–529
40. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 629–638
41. Tasdizen, T., Whitaker, R.: Anisotropic diffusion of surface normals for feature preserving surface reconstruction. In: Fourth International Conference on 3-D Digital Imaging and Modeling. (2003) 353–360
42. Smith, S.M., Brady, J.M.: SUSAN – a new approach to low level image processing. International Journal of Computer Vision **23** (1997) 45–78
43. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: ICCV '98: Proceedings of the Sixth International Conference on Computer Vision. (1998) 839–846

Efficient Linear System Solvers for Mesh Processing

Mario Botsch, David Bommes, and Leif Kobbelt

Computer Graphics Group,
RWTH Aachen Technical University, Germany
{botsch, bommes, kobbelt}@cs.rwth-aachen.de

Abstract. The use of polygonal mesh representations for freeform geometry enables the formulation of many important geometry processing tasks as the solution of one or several linear systems. As a consequence, the key ingredient for efficient algorithms is a fast procedure to solve linear systems. A large class of standard problems can further be shown to lead more specifically to sparse, symmetric, and positive definite systems, that allow for a numerically robust and efficient solution.

In this paper we discuss and evaluate the use of *sparse direct solvers* for such kind of systems in geometry processing applications, since in our experiments they turned out to be superior even to highly optimized multigrid methods, but at the same time were considerably easier to use and implement. Although the methods we present are well known in the field of high performance computing, we observed that they are in practice surprisingly rarely applied to geometry processing problems.

1 Introduction

In the field of geometry processing suitable data structures that enable the implementation of efficient algorithms are getting more and more important [1], especially since the complexity of the geometric models to be processed is growing much faster than the steadily increasing computational power and available memory of today's PC systems. Typical examples are higher order spline surfaces $\mathbf{f}(u,v) = \sum_i \mathbf{c}_i \Phi_i(u,v)$, represented as a weighted average of control points $\mathbf{c}_i$, or piecewise linear triangle meshes $\mathcal{M}$ obtained from sampling a surface at the mesh vertices $\mathbf{x}_i = \mathbf{f}(u_i, v_i)$.

Using finite differences or finite elements, many standard geometry processing problems, like for instance PDEs on or of surfaces, can be formulated as a set of (linear or non-linear) equations in either the control points $\mathbf{c}_i$ of a spline surface or the vertex positions $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T \in \mathbb{R}^{n\times 3}$ of an approximating triangle mesh.

A common technique to efficiently handle non-linear problems is their decomposition into a sequence of linear ones, like, e.g., the (semi-)implicit integration of non-linear geometric flows by solving a linear equation in each time step [1] or the Levenberg-Marquardt method for non-linear optimization [2]. Similarly, continuous energy functionals $E(\mathbf{f}) = \int_\Omega e(\mathbf{f}, \mathbf{x})\, d\mathbf{x}$ are approximated by quadratic

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 62–83, 2005.

forms $E(X) = X^T Q X$, such that their minimizer surfaces can efficiently be derived by solving the linear systems $QX = B$, assuming proper boundary constraints B [3]. These examples motivate why a large class of geometric problems comes down to the solution of one or several linear systems. As a consequence, high performance linear system solvers are of major importance for the development of efficient algorithms.

Since differential surface properties are defined locally, the discretization of PDEs typically leads to *sparse* linear systems, in which the ith row contains non-zeros only in those entries corresponding to the geodesic or topological neighborhood of vertex $\mathbf{x}_i$. We are therefore interested in solvers that exploit this sparsity in order to minimize both memory consumption and computation times.

Within the class of sparse linear systems, we will further concentrate on symmetric positive definite (so-called *spd*) matrices, since exploiting their special structure allows for the most efficient and most robust implementations. Such systems frequently occur when minimizing energy functionals of the form $E(X) = X^T Q X$ with an spd matrix Q. A very popular source of spd systems is the discrete Laplace-Beltrami operator $\Delta_{\mathcal{S}}$ [4], which is closely related to frequencies of scalar fields defined on a two-manifold surface $\mathcal{S}$ [5]. This operator has various applications in surface smoothing [1, 5], surface parameterization [6, 7], variational surface modeling [8, 9, 10, 11], mesh morphing [12, 13, 14], and shape analysis [15]. Besides from surfaces, the standard Laplace operator is for instance also used in image editing [16] and fluid simulation [17]. Finally, all linear problems $A\mathbf{x} = \mathbf{b}$ that cannot be solved exactly and hence are approximated in the least squares sense by using the normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$ also result in spd linear systems [18]. This large but still incomplete list of applications involving spd systems legitimates focusing on this special class of problems.

Another important point to be considered is whether the linear systems are solved just once or several times, e.g., for different right-hand sides. Since most geometric problems are separable w.r.t. the coordinate components, they can be solved component-wise for x, y, and z using the same system matrix. Multiple right-hand side problems also naturally occur in applications where the user interactively changes boundary constraints, e.g., in surface editing.

There is another situation for solving a sequence of similar systems: when decomposing a non-linear problem into a sequence of linear systems, the values of the matrix entries usually change in each iteration, but its structure, i.e., the pattern of non-zero elements $\{(i,j) \mid A_{ij} \neq 0\}$, stays the same, because it usually depends on the mesh connectivity, only which does not change. In both cases — solving for multiple right-hand sides or matrices of identical structure — this additional information should be exploited as much as possible, e.g., by investing pre-computation time in some kind of factorization or preconditioning.

In this paper we propose the use of direct solvers for the sparse spd systems, as they arise from typical computer graphics and geometry processing problems. We mainly focus on Laplacian or bi-Laplacian systems for triangle meshes, however, analogous results hold for systems of similar structure. The size of the linear systems corresponds to the number of vertices in the mesh, which, in our context, usually is of the order of 10^4 or 10^5. Due to the local definition of the Laplace operator, the resulting matrices are highly sparse and in the average exhibit 7 or 19 non-zero entries per row for Laplacian or bi-Laplacian systems, respectively. Since in many applications these systems have to be solved for multiple right-hand sides, the sparse factorizations of direct solvers allow for highly efficient implementations. After reviewing the commonly known and widely used direct and iterative solvers, we introduce sparse direct solvers and point out their advantageous properties in Sect. 2. After comparing the different solvers in Sect. 3 we finally present a list of applications that greatly benefit from sparse direct solvers in Sect. 4.

2 Linear System Solvers

We describe and compare the following classes of solvers: dense direct solvers, iterative solvers, multigrid solvers, and finally sparse direct solvers. For the following discussion we restrict to sparse spd problems $A\mathbf{x} = \mathbf{b}$, with $A = A^T \in \mathbb{R}^{n \times n}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$, and denote by $\mathbf{x}^*$ the exact solution $A^{-1}\mathbf{b}$. For completeness, the general case of a non-symmetric indefinite system is outlined in Sect. 2.5. More elaborate surveys on how to efficiently solve general large linear systems can be found in the books [19, 20].

2.1 Dense Direct Solvers

Direct linear system solvers are based on a factorization of the matrix A into matrices of simpler structure, e.g., triangular, diagonal, or orthogonal matrices. This structure allows for an efficient solution of the factorized system. As a consequence, once the factorization is computed, it can be used to solve the linear system for several different right-hand sides.

The most commonly used examples for *general* matrices A are, in the order of increasing numerical robustness and computational effort, the LU factorization, QR factorization, or the singular value decomposition. However, in the special case of a spd matrix the Cholesky factorization $A = LL^T$, with L denoting a lower triangular matrix, should be employed, since it exploits the symmetry of the matrix and can additionally be shown to be numerically very robust due to the positive definiteness of the matrix A [21].

On the downside, the asymptotic time complexity of all dense direct methods is $O(n^3)$ for the factorization and $O(n^2)$ for solving the system based on the pre-computed factorization. Since for the problems we are targeting at, n can be of the order of 10^5, the total cubic complexity of dense direct methods is prohibitive. Even if the matrix A is highly sparse, the naïve direct methods enumerated here are not designed to exploit this structure, hence the factors are dense matrices in general (cf. Fig. 2, top row, on page 70).

2.2 Iterative Solvers

In contrast to dense direct solvers, iterative methods are able to exploit the sparsity of the matrix A. Since they additionally allow for a simple implementation

[22], iterative solvers are the de-facto standard method for solving sparse linear systems in the context of geometric problems. A detailed overview of iterative methods with precious implementation hints can be found in [23, 24].

Iterative methods compute a converging sequence $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(i)}$ of approximations to the solution $\mathbf{x}^*$ of the linear system, i.e., $\lim_{i\to\infty} \mathbf{x}^{(i)} = \mathbf{x}^*$. In practice, however, one has to find a suitable criterion to stop the iteration as soon as the current solution $\mathbf{x}^{(i)}$ is accurate enough, i.e., if the norm of the error $\mathbf{e}^{(i)} := \mathbf{x}^* - \mathbf{x}^{(i)}$ is less than some ε. Since the solution $\mathbf{x}^*$ is not known beforehand, the error has to be estimated by considering the residual $\mathbf{r}^{(i)} := A\mathbf{x}^{(i)} - \mathbf{b}$. These two are related by the *residual equations* $A\mathbf{e}^{(i)} = \mathbf{r}^{(i)}$, leading to an upper bound $\left\|\mathbf{e}^{(i)}\right\| \leq \left\|A^{-1}\right\| \cdot \left\|\mathbf{r}^{(i)}\right\|$, i.e., the norm of the inverse matrix has to be estimated or approximated in some way (see [23]).

The simplest examples for iterative solvers are the Jacobi and Gauss-Seidel methods. They belong to the class of static iterative methods, whose update steps can be written as $\mathbf{x}^{(i+1)} = M\mathbf{x}^{(i)} + \mathbf{c}$ with constant M and $\mathbf{c}$, such that the solution $\mathbf{x}^*$ is the fixed point of this iteration. An analysis of the eigenstructure of the update matrices M reveals that both methods rapidly remove the high frequencies of the error, but the iteration stalls if the error is a smooth function. By consequence, the convergence to the exact solution $\mathbf{x}^*$ is usually too slow in practice. As an additional drawback these methods only converge for a restricted set of matrices, e.g., for diagonally dominant ones.

Non-stationary iterative solvers are more powerful, and for spd matrices the method of conjugate gradients (CG) [25, 21] is suited best, since it provides guaranteed convergence with monotonically decreasing error. For a spd matrix A the solution of $A\mathbf{x} = \mathbf{b}$ is equivalent to the minimization of the quadratic form

$$\phi\left(\mathbf{x}\right) := \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}.$$

The CG method successively minimizes this functional along a set of linearly independent search directions $\mathbf{p}^{(i)}$, such that

$$\mathbf{x}^{(i)} = \operatorname{argmin}\left\{\phi\left(\mathbf{x}\right) \,\middle|\, \mathbf{x} \in \mathbf{x}_0 + \operatorname{span}\left\{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(i)}\right\}\right\}.$$

Due to the nestedness of these spaces the error decreases monotonically, and the exact solution $\mathbf{x}^* \in \mathbb{R}^n$ is found after at most n steps (neglecting rounding errors). Minimizing ϕ by gradient descent results in inefficient zigzag paths in steep valleys of ϕ, which correspond to strongly differing eigenvalues of A. In order to cancel out the effect of A's eigenvalues on the search directions $\mathbf{p}_i$, those are chosen to be *A-conjugate*, i.e., orthogonal w.r.t. the scalar product induced by A: $\mathbf{p}_j^T A\mathbf{p}_i = 0$ for $i \neq j$ [26]. The computation of and minimization along these optimal search directions can be performed efficiently and with a constant memory consumption.

The complexity of each CG iteration is mainly determined by the matrix-vector product $A\mathbf{x}$, which is of the order $O(n)$ if the matrix is sparse. Given the maximum number of n iterations, the total complexity is $O(n^2)$ in the worst case, but it is usually better in practice.

As the convergence rate mainly depends on the spectral properties of the matrix A, a proper pre-conditioning scheme should be used to increase the efficiency and robustness of the iterative scheme. This means that a slightly different system $\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ is solved instead, with $\tilde{A} = PAP^T$, $\tilde{\mathbf{x}} = P^{-T}\mathbf{x}$, $\tilde{\mathbf{b}} = P\mathbf{b}$, where the regular pre-conditioning matrix P is chosen such that $\tilde{A}$ is well conditioned [21, 23]. However, the matrix P is restricted to have a simple structure, since an additional linear system $P\mathbf{z} = \mathbf{r}$ has to be solved in each iteration of the solver.

The iterative conjugate gradients method manages to decrease the computational complexity from $O(n^3)$ to $O(n^2)$ for sparse matrices, but this is still too slow to compute exact (or sufficiently accurate) solutions of large linear systems, in particular if the systems are numerically ill-conditioned, like for instance the higher order Laplacian systems used in variational surface modeling [8, 9].

2.3 Multigrid Iterative Solvers

As mentioned in the last section, one characteristic problem of most iterative solvers is that they are *smoothers*: they attenuate the high frequencies of the error $\mathbf{e}^{(i)}$ very fast, but their convergence stalls if the error is a smooth function. This fact is exploited by multigrid methods, that build a fine-to-coarse hierarchy $\{\mathcal{M} = \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k\}$ of the computation domain $\mathcal{M}$ and solve the linear system hierarchically from coarse to fine [27, 28].

After a few (pre-)smoothing iterations on the finest level $\mathcal{M}_0$ the high frequencies of the error are removed and the solver becomes inefficient. However, the remaining low frequency error $\mathbf{e}_0 = \mathbf{x}^* - \mathbf{x}_0$ on $\mathcal{M}_0$ corresponds to higher frequencies when restricted to the coarser level $\mathcal{M}_1$ and therefore can be removed efficiently on $\mathcal{M}_1$. Hence the error is solved for using the residual equations $A\mathbf{e}_1 = \mathbf{r}_1$ on $\mathcal{M}_1$, where $\mathbf{r}_1 = R_{0\to1}\mathbf{r}_0$ is the residual on $\mathcal{M}_0$ transfered to $\mathcal{M}_1$ by a restriction operator $R_{0\to1}$. The result is prolongated back to $\mathcal{M}_0$ by $\mathbf{e}_0 \leftarrow P_{1\to0}\mathbf{e}_1$ and used to correct the current approximation: $\mathbf{x}_0 \leftarrow \mathbf{x}_0 + \mathbf{e}_0$. Small high-frequency errors due to the prolongation are finally removed by a few post-smoothing steps on $\mathcal{M}_0$. The recursive application of this two-level approach to the whole hierarchy can be written as

$$\Phi_i \;=\; S_\mu \, P_{i+1\to i} \, \Phi_{i+1} \, R_{i\to i+1} \, S_\lambda ,$$

with λ and μ pre- and post-smoothing iterations, respectively. One recursive run is known as a *V-cycle* iteration.

Another concept is the method of *nested iterations*, that exploits the fact that iterative solvers are very efficient if the starting value is sufficiently close to the actual solution. One starts by computing the exact solution on the coarsest level $\mathcal{M}_k$, which can be done efficiently since the system $A_k\mathbf{x}_k = \mathbf{b}_k$ corresponding to the restriction to $\mathcal{M}_k$ is small. The prolongated solution $P_{k\to k-1}\mathbf{x}_k^*$ is then used as starting value for iterations on $\mathcal{M}_{k-1}$, and this process is repeated until the finest level $\mathcal{M}_0$ is reached and the solution $\mathbf{x}_0^* = \mathbf{x}^*$ is computed.

The remaining question is how to iteratively solve on each level. The standard method is to use one or two V-cycle iterations, leading to the so-called *full multigrid* method. However, one can also use an iterative smoothing solver (e.g.,

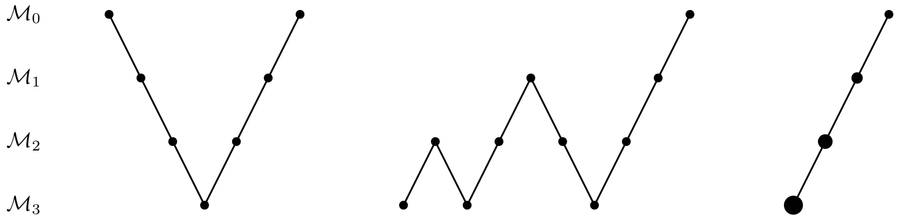


Fig. 1. A schematic comparison in terms of visited multigrid levels for V-cycle (*left*), full multigrid with one V-cycle per level (*center*), and cascading multigrid (*right*). The size of the dots represents the number of iterations on the respective level

Jacobi or CG) on each level and completely avoid V-cycles. In the latter case the number of iterations m_i on level i must not be constant, but instead has to be chosen as $m_i = m\,\gamma^i$ to decrease exponentially from coarse to fine [29]. Besides the easier implementation, the advantage of this *cascading multigrid* method is that once a level is computed, it is not involved in further computations and can be discarded. A comparison of the three methods in terms of visited multigrid levels is given in Fig. 1.

Due to the logarithmic number of hierarchy levels $k = O(\log n)$ the full multigrid method and the cascading multigrid method can both be shown to have linear asymptotic complexity, as opposed to quadratic for non-hierarchical iterative methods. However, they cannot exploit synergy for multiple right-hand sides, which is why factorization-based approaches are clearly preferable in such situations, as we will show in the next section.

Since in our case the discrete computational domain $\mathcal{M}$ is an irregular triangle mesh instead of a regular 2D or 3D grid, the coarsening operator for building the hierarchy is based on mesh decimation techniques [30, 31]. The shape of the resulting triangles is important for numerical robustness, and the edge lengths on the different levels should mimic the case of regular grids. Therefore the decimation usually removes edges in the order of increasing lengths, such that the hierarchy levels have uniform edge lengths and triangles of bounded aspect ratio.

The simplification from one hierarchy level $\mathcal{M}_i$ to the next coarser one $\mathcal{M}_{i+1}$ should additionally be restricted to remove a *maximally independent set* of vertices, i.e., no two removed vertices $v_j, v_l \in \mathcal{M}_i \setminus \mathcal{M}_{i+1}$ are connected by an edge $e_{jl} \in \mathcal{M}_i$. In [32] some more efficient alternatives to this standard Dobkin-Kirkpatric hierarchy are discussed. In order to achieve higher performance, we do not change the simple way the hierarchy is constructed, but instead solve the linear system on every second or third level only, and use the prolongation operator alone on all in-between levels.

The linear complexity of multi-grid methods allows for the highly efficient solution even of very complex systems. However, the main problem of these solvers is their quite involved implementation, since special care has to be taken for the hierarchy building, for special multigrid pre-conditioners, and for the inter-level conversion by restriction and prolongation operators. A detailed overview of these techniques is given in [32].

Additionally, the number of iterations per hierarchy level have to be chosen: This includes the number of V-cycles and pre- and post-smoothing iterations per V-cycle for the full multigrid method, or m and γ for the cascading multigrid approach. These numbers have to be chosen either by heuristic or experience, since they not only depend on the problem (structure of A), but also on its specific instance (values of A). Nevertheless, if iterative solvers are to be used, multigrid methods are the only way to achieve acceptable computing times for solving large systems, as has been shown in [8, 33, 32].

2.4 Sparse Direct Solvers

The use of direct solvers for large sparse linear systems is often neglected, since naïve direct methods have complexity $O(n^3)$, as described above. The problem is that even when the matrix A is sparse, the factorization will not preserve this sparsity, such that the resulting Cholesky factor is a dense lower triangular matrix (cf. Fig. 2, top row).

However, an analysis of the factorization process shows that a *band-limitation* of the matrix A will be preserved. Following [34], we define the bandwidth $\beta(A)$ in terms of the bandwidth of its ith row

$$\beta(A) \;:=\; \max_{1\leq i\leq n}\{\beta_i(A)\} \quad \text{with} \quad \beta_i(A) \;:=\; i - \min_{1\leq j\leq i}\{j \mid A_{ij}\neq 0\}\,.$$

If the matrix A has bandwidth $\beta(A)$ then so has its factor L. An even stricter bound is that also the so-called *envelope* or *profile*

$$\mathrm{Env}(A) \;:=\; \{(i,j) \mid 0 < i-j \leq \beta_i(A)\}$$

is preserved, i.e., no additional non-zeros (so-called *fill-in* elements) are generated outside the envelope.

This additional structure can be exploited in both the factorization and the solution process, such that their complexities reduce from $O(n^3)$ and $O(n^2)$ to linear complexity in the number of non-zeros $\mathrm{nz}(A)$ of A [34]. Since usually $\mathrm{nz}(A) = O(n)$, this is the same linear complexity as for multigrid solvers. However, in the graphics-related examples we will show in the following sections, sparse direct method turned out to be more efficient compared to multigrid methods, in particular for multiple right-hand side problems.

Since we assume the matrices to be sparse, but not band-limited or profile-optimized, the first step is to minimize the matrix envelope, which can be achieved by symmetric row and column permutations $A \mapsto P^T A P$ using a permutation matrix P, i.e., a re-ordering of the mesh vertices. Although this re-ordering problem is NP complete, several good heuristics exist, of which we will present the most commonly used in the following. All of these methods work on the undirected *adjacency graph* $\mathrm{Adj}(A)$ corresponding to the non-zeros of A, i.e., two nodes $i, j \in \{1, \ldots, n\}$ are connected by an edge if and only if $A_{ij} \neq 0$.

The standard method for envelope minimization is the *Cuthill-McKee* algorithm [35], that picks a start node and renumbers all its neighbors by traversing the adjacency graph in a breadth-first manner, using a greedy selection in the order of increasing valence. It has further been proven in [36] that reverting this permutation leads to better re-orderings, such that usually the *reverse Cuthill-McKee* method (RCMK) is employed. The result P^TAP of this matrix re-ordering is depicted in the second row of Fig. 2.

Since no special pivoting is required for the Cholesky factorization, the non-zero structure of its matrix factor L can symbolically be derived from the non-zero structure of the matrix A alone, or, equivalently, from its adjacency graph. The graph interpretation of the Cholesky factorization is to successively eliminate the node with the lowest index from the graph and connect all its immediate neighbors mutually to each other. The additional edges e_{ij} generated in this so-called *elimination graph* correspond to the fill-in elements $L_{ij} \neq 0 = A_{ij}$.

In order to minimize fill-in the *minimum degree* algorithm (MD) and its variants [37, 38] remove the nodes with smallest valence first from the elimination graph, since this causes the least number of additional pairwise connections. Many efficiency optimizations of this basic method exist, the most prominent of which is the *super-nodal* approach: instead of removing eliminated nodes from the graph, neighboring eliminated nodes are clustered to so-called super-nodes, allowing for more efficient graph updates. The resulting minimum degree re-orderings do not lead to some kind of a band-structure (which implicitly limits fill-in), but instead directly minimize the fill-in of L (cf. Fig. 2, third row).

The last class of re-ordering approaches is based on graph partitioning. Consider a matrix A whose adjacency graph has m separate connected components. Such a matrix can be restructured to a block-diagonal matrix of m blocks, such that the factorization can be performed on each block individually. If the adjacency graph is connected, a small subset S of nodes, whose elimination would separate the graph into two components of roughly equal size, is found by one of several heuristics [39]. This graph-partitioning results in a matrix consisting of two large diagonal blocks (two connected components) and $|S|$ rows representing their connection (separator S). Recursively repeating this process leads to the method of *nested dissection* (ND), leading to matrices of the typical block structure shown in the bottom row of Fig. 2. Besides the obvious fill-in reduction, these systems also allow for easy parallelization of both the factorization and the solution.

For the comparison of the different matrix re-ordering strategies a rather small matrix was used in Fig. 2 to allow for clearer visualization. On an analogous $5k \times 5k$ matrix the number of non-zeros $\mathrm{nz}(L)$ decreases from 2.3M to 451k, 106k, and 104k by applying the RCMK, MD, and ND method, respectively. The timings to obtain those re-orderings are 17ms, 12ms, and 38ms. It can further be observed that for larger systems the nested dissection method [39] generally leads to the best results.

One important advantage of the Cholesky factorization is that the non-zero structure of the factor L can be determined from $\mathrm{Adj}(A)$ without any numerical

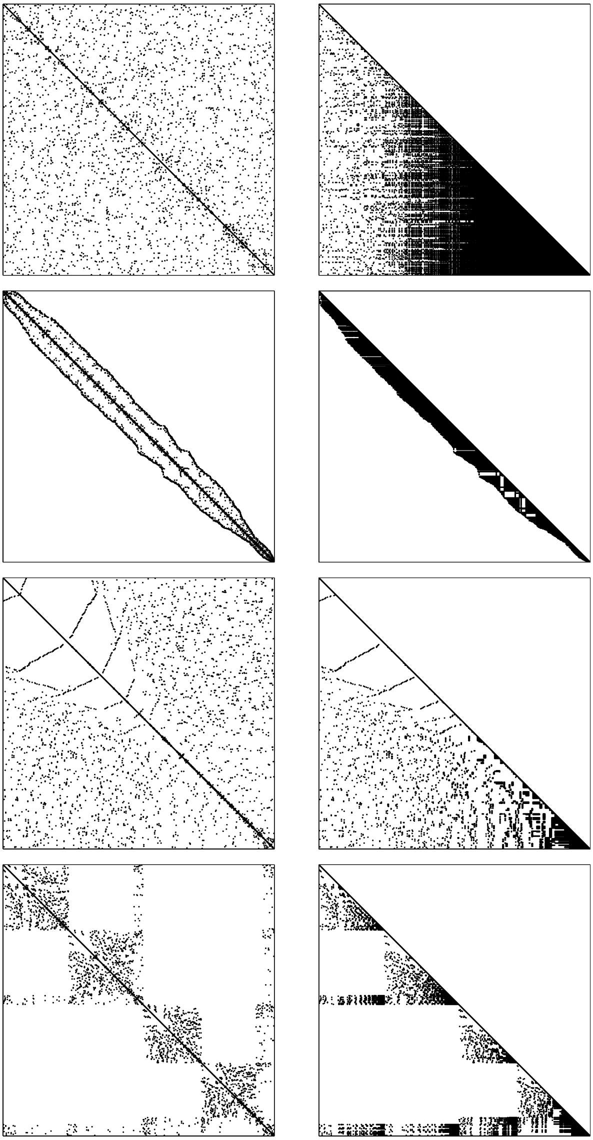


Fig. 2. The top row shows the non-zero pattern of a typical 500 × 500 matrix A and its Cholesky factor L, corresponding to a Laplacian system on a triangle mesh. Although A is highly sparse (3502 non-zeros), the factor L is dense (36k non-zeros). The reverse Cuthill-McKee algorithm minimizes the envelope of the matrix, resulting in 14k non-zeros of L (*2nd row*). The minimum degree ordering avoids fill-in during the factorization, which decreases the number of non-zeros to 6203 (*3rd row*). The last row shows the result of a nested dissection method (7142 non-zeros), that allows for parallelization due to its block structure

computations. This allows us to setup an efficient *static* data structure for L *before* the actual *numerical factorization*, which is therefore called *symbolic factorization*. Since suitable data structures and proper memory layout are crucial for efficient numerical computations, this two-step factorization process allows for significant optimizations.

Analogously to the dense direct solvers, the factorization can be exploited to solve for different right-hand sides in a very efficient manner. In addition to this, whenever the matrix A is changed, such that its non-zero structure $\mathrm{Adj}(A)$ is preserved, then the matrix re-ordering as well as the symbolic factorization can obviously be re-used. Solving the modified system therefore only requires to re-compute the numerical factorization and performing the back-substitution, which typically saves about 50% of the total computation time for solving the modified system. As we will show in Sect. 4, this allows for an efficient implementation of a large class of algorithms that decompose a non-linear problem into a sequence of similar linear ones, like for instance the implicit fairing approach [1] or the Levenberg-Marquardt optimization for non-linear problems [22, 2].

Another advantage of sparse direct methods is that no additional parameters have to be chosen in a problem-dependent manner, as for instance the different numbers of iterations for the multigrid solvers. The only degree of freedom is the matrix re-ordering, but this only depends on the symbolic structure of the problem and therefore can be chosen quite easily. For more details and implementation notes the reader is referred to the book of George and Liu [34]; a highly efficient implementation is publicly available in the TAUCS library [40].

2.5 Non-symmetric Indefinite Systems

When the assumptions about the symmetry and positive definiteness of the matrix A are not satisfied, optimal methods like the Cholesky factorization or conjugate gradients cannot be used. In this section we shortly outline which techniques are applicable instead.

From within the class of iterative solvers, the bi-conjugate gradients algorithm (BiCG) is typically used as a replacement for the conjugate gradients method [22]. Although working well in most cases, BiCG does not provide any theoretical convergence guarantees and has a very irregular non-monotonically decreasing residual error for ill-conditioned systems. On the other hand, the GMRES method converges monotonically with guarantees, but its computational cost and memory consumption increase in each iteration [21]. As a good trade-off, the stabilized Bi-CGSTAB [23] represents a mixture between the efficient BiCG and the smoothly converging GMRES; it provides a much smoother convergence and is reasonably efficient and easy to implement.

When considering dense direct solvers, the Cholesky factorization cannot be used for general matrices. Therefore the LU factorization is typically employed (instead of QR or SVD), since it is similarly efficient and also extends well to sparse direct methods. However, (partial) row and column pivoting is essential for the numerical robustness of the LU factorization, since this avoids zeros on the diagonal during the factorization process.

Similarly to the Cholesky factorization, it can be shown that the LU factorization also preserves the band-width and envelope of the matrix A. Techniques like the minimum degree algorithm generalize to non-symmetric matrices as well. But as for dense matrices, the banded LU factorization relies on partial pivoting in order to guarantee numerical stability. In this case, two competing types of permutations are involved: symbolic permutations for matrix re-ordering and pivoting permutations ensuring numerical robustness. As these permutations cannot be handled separately, a trade-off between stability and fill-in minimization has to be found, resulting in a significantly more complex factorization process.

As a consequence, the re-ordering depends on the numerical values of the matrix entries, such that an exact symbolic factorization like in the Cholesky case is not possible. In order to nevertheless be able to setup a static data structure, a more conservative envelope is typically used, such that pivoting within this structure is still possible. A highly efficient implementation of a sparse LU factorization is provided by the SuperLU library [41].

3 Laplace Systems

Most of the example applications shown in Sect. 4 require the solution of linear Laplacian systems, therefore we analyze these matrices and compare different solvers for their solution. Although we focus on Laplacian systems, we will see in Sect. 4 that analogous results hold for matrices of similar structure, like for instance sparse least squares systems.

The discrete Laplace-Beltrami operator $\Delta_{\mathcal{S}} f$ of a scalar-valued function f on the manifold $\mathcal{S}$ [1, 4, 6] is defined for a center vertex v_i as a linear combination with its one-ring neighbors $v_j \in N_1(v_i)$:

$$\Delta_{\mathcal{S}} f(v_i) \;=\; \frac{2}{A(v_i)} \sum_{v_j \in N_1(v_i)} \left(\cot\alpha_{ij} + \cot\beta_{ij}\right)\left(f(v_j) - f(v_i)\right),$$

where $\alpha_{ij} = \angle(\mathbf{x}_i, \mathbf{x}_{j-1}, \mathbf{x}_j)$, $\beta_{ij} = \angle(\mathbf{x}_i, \mathbf{x}_{j+1}, \mathbf{x}_j)$, and $\mathbf{x}_i$ represents the 3D position of the mesh vertex v_i. The normalization factor $A(v_i)$ denotes the Voronoi area around the vertex v_i [4]. In matrix notation the vector of the Laplacians of $f(v_i)$ can be written as

$$\begin{pmatrix} \vdots \\ \Delta_{\mathcal{S}} f(v_i) \\ \vdots \end{pmatrix} \;=\; D \cdot M \cdot \begin{pmatrix} \vdots \\ f(v_i) \\ \vdots \end{pmatrix},$$

where D is a diagonal matrix with normalization factors $D_{ii} = 2/A(v_i)$, and M is a symmetric matrix of cotangent weights with

$$M_{ij} \;=\; \begin{cases} 0 & i \neq j\,,\ v_j \ /\!\!\!\in N_1(v_i) \\ \cot\alpha_{ij} + \cot\beta_{ij}, & i \neq j\,,\ v_j \in N_1(v_i) \\ -\sum_{v_j \in N_1(v_i)} \left(\cot\alpha_{ij} + \cot\beta_{ij}\right) & i = j \end{cases}.$$

Since the Laplacian of a vertex v_i is defined *locally* in terms of its one-ring neighbors, the matrix M is highly sparse and has non-zeros in the ith row only on the diagonal and in those columns corresponding to v_i's one-ring neighbors $N_1\left(v_i\right)$. Due to the Euler characteristic for triangle meshes, this results in about 7 non-zeros per row in average. Analogously, higher order Laplacian matrices $\Delta_{\mathcal{S}}^k$ have non-zeros for the k-ring neighbors $N_k\left(v_i\right)$, which are, e.g., about 19 for bi-Laplacian systems ($k = 2$).

For a closed mesh without boundaries, Laplacian systems $\Delta_{\mathcal{S}}^k \mathbf{x} = \mathbf{b}$ of any order k can be turned into symmetric ones by moving the first diagonal matrix factor D to the right-hand side:

$$M\left(DM\right)^{k-1}\mathbf{x} \;=\; D^{-1}\mathbf{b}.$$

Boundary constraints are typically employed by restricting the positions of certain vertices, which corresponds to eliminating their respective rows and columns of the left-hand side and hence keeps the matrix symmetric. The case of meshes with boundaries is equivalent to a patch bounded by constrained vertices and therefore also results in a symmetric matrix. Pinkall and Polthier [6] additionally showed that this system is positive definite, such that we can apply the efficient solvers presented in the last section.

In the following we compare the different kinds of linear system solvers for Laplacian as well as for bi-Laplacian systems. All timings we report in this and the next section were taken on a 3.0GHz Pentium4 running Linux. The iterative solver from the `gmm++` library [42] is based on the conjugate gradients method and uses an incomplete LDL^T factorization as preconditioner. Our cascading multigrid solver performs preconditioned conjugate gradient iterations on each hierarchy level and additionally exploits SSE instructions in order to solve for up to four right-hand sides simultaneously. The direct solver of the TAUCS library [40] employs nested dissection re-ordering and a sparse complete Cholesky factorization. Although our linear systems are symmetric, we also compare to the popular SuperLU solver [41], which is based on a sparse LU factorization, for the sake of completeness.

Iterative solvers have the advantage over direct ones that the computation can be stopped as soon as a sufficiently small error is reached, which — in typical computer graphics applications — does not have to be the highest possible precision. In contrast, direct methods always compute the exact solution up to numerical round-off errors, which in our application examples actually was more precise than required. The stopping criteria of the iterative methods have therefore been chosen to yield sufficient results, such that their quality is comparable to that achieved by direct solvers. The resulting residual errors were allowed to be about one order of magnitude larger than those of the direct solvers. While the latter achieved an average residual error of 10^{-7} and 10^{-5} for Laplacian and bi-Laplacian systems, respectively, the iterative solvers were stopped at an error of 10^{-6} and 10^{-4}.

Table 1 shows timings for the different solvers on Laplacian systems $\Delta_{\mathcal{S}} X = B$ of 10k to 50k and 100k to 500k unknowns, i.e., free vertices X. For each solver three columns of timings are given:

Setup: Computing the cotangent weights for the Laplace discretization and building the matrix structure (done per-level for the multigrid solver).

Precomputation: Preconditioning (*iterative*), building the hierarchy by mesh decimation (*multigrid*), matrix re-ordering and sparse factorization (*direct*).

Solution: Solving the linear system for three different right-hand sides corresponding to the x, y, and z components of the free vertices X.

Due to its effective preconditioner, which computes a sparse incomplete factorization, the iterative solver scales almost linearly with the system complexity. However, for large and thus ill-conditioned systems it breaks down. Notice that without preconditioning the solver would not converge for the larger systems.

The experiments clearly verify the linear complexity of multigrid and sparse direct solvers. Once their sparse factorizations are pre-computed, the computational costs for actually solving the system are about the same for the LU and Cholesky solver. However, they differ significantly in the factorization performance, because the numerically more robust Cholesky factorization allows for more optimizations, whereas pivoting is required for the LU factorization to guarantee robustness. This is the reason for the break-down of the LU solver, such that the multigrid solver is more efficient in terms of total computation time for the larger systems.

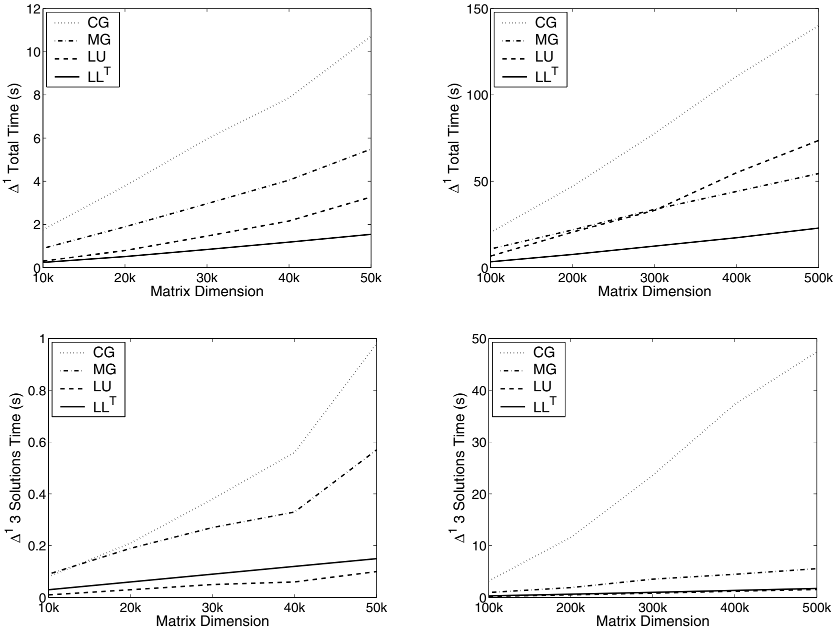
Interactive applications often require to solve the same linear system for several right-hand sides (e.g. once per frame), which typically reflects the change of boundary constraints due to user interaction. For such problems the solution times, i.e., the third columns of the timings, are more relevant, as they correspond to the per-frame computational costs. Here the precomputation of a sparse factorization pays off and the direct solvers are clearly superior to the multigrid method.

Table 2 shows the same experiments for bi-Laplacian systems $\Delta_{\mathcal{S}}^2 X = B$ of the same complexity. In this case, the matrix setup is more complex, the matrix condition number is squared, and the sparsity decreases from 7 to 19 non-zeros per row.

Due to the higher condition number the iterative solver takes much longer and even fails to converge on large systems. In contrast, the multigrid solver converges robustly without numerical problems; notice that constructing the multigrid hierarchy is almost the same as for the Laplacian system (up to one more ring of boundary constraints). The computational costs required for the sparse factorization are proportional to the increased number of non-zeros per row. The LU factorization additionally has to incorporate pivoting for numerical stability and failed for larger systems. In contrast, the Cholesky factorization worked robustly in all our experiments.

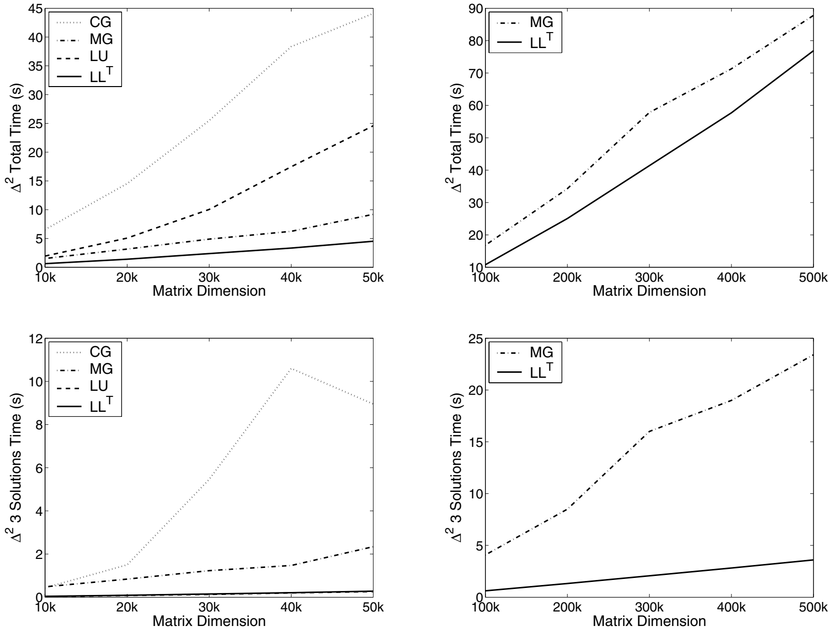
If we focus on the solution times for the bi-Laplacian systems and compare them to the Laplacian systems, we observe that the direct solver scales with the sparsity of the matrix, while the number of iterations required for the multigrid solver depends on the (squared) matrix condition. In our experiments it turned out that the performance gap between multigrid and direct methods is even larger for bi-Laplacian systems.

Table 1. Comparison of different solvers for Laplacian systems $\Delta_{\mathcal{S}} X = B$ of 10k to 50k and 100k to 500k free vertices X. The three timings for each solver represent matrix setup, pre-computation, and three solutions for the x, y, and z components of X. The graphs in the upper row show the total computation times (sum of all three columns). The center row depicts the solution times only (3rd column), as those typically determine the per-frame cost in interactive applications



Size	Iterative	Multigrid	LU	Cholesky
10k	0.11/1.56/0.08	0.15/0.65/0.09	0.07/0.22/0.01	0.07/0.14/0.03
20k	0.21/3.36/0.21	0.32/1.38/0.19	0.14/0.62/0.03	0.14/0.31/0.06
30k	0.32/5.26/0.38	0.49/2.20/0.27	0.22/1.19/0.05	0.22/0.53/0.09
40k	0.44/6.86/0.56	0.65/3.07/0.33	0.30/1.80/0.06	0.31/0.75/0.12
50k	0.56/9.18/0.98	0.92/4.00/0.57	0.38/2.79/0.10	0.39/1.00/0.15
100k	1.15/16.0/3.19	1.73/8.10/0.96	0.79/5.66/0.21	0.80/2.26/0.31
200k	2.27/33.2/11.6	3.50/16.4/1.91	1.56/18.5/0.52	1.59/5.38/0.65
300k	3.36/50.7/23.6	5.60/24.6/3.54	2.29/30.0/0.83	2.35/9.10/1.00
400k	4.35/69.1/37.3	7.13/32.5/4.48	2.97/50.8/1.21	3.02/12.9/1.37
500k	5.42/87.3/47.4	8.70/40.2/5.57	3.69/68.4/1.54	3.74/17.4/1.74

Table 2. Comparison of different solvers for bi-Laplacian systems $\Delta_{\mathcal{S}}^2 X = B$ of 10k to 50k and 100k to 500k free vertices X. The three timings for each solver represent matrix setup, pre-computation, and three solutions for the x, y, and z components of X. The graphs in the upper row show the total computation times (sum of all three columns). The center row depicts the solution times only (3rd column), as those typically determine the per-frame cost in interactive applications. For the larger systems, both the iterative solver and the sparse LU factorization fail to compute a solution



Size	Iterative	Multigrid	LU	Cholesky
10k	0.33/5.78/0.44	0.40/0.65/0.48	0.24/1.68/0.03	0.24/0.35/0.04
20k	0.64/12.4/1.50	0.96/1.37/0.84	0.49/4.50/0.08	0.49/0.82/0.09
30k	1.04/19.0/5.46	1.40/2.26/1.23	0.77/9.15/0.13	0.78/1.45/0.15
40k	1.43/26.3/10.6	1.69/3.08/1.47	1.07/16.2/0.20	1.08/2.05/0.21
50k	1.84/33.3/8.95	2.82/4.05/2.34	1.42/22.9/0.26	1.42/2.82/0.28
100k	—	4.60/8.13/4.08	2.86/92.8/0.73	2.88/7.29/0.62
200k	—	9.19/16.6/8.50	—	5.54/18.2/1.32
300k	—	17.0/24.8/16.0	—	8.13/31.2/2.07
400k	—	19.7/32.6/19.0	—	10.4/44.5/2.82
500k	—	24.1/40.3/23.4	—	12.9/60.4/3.60

We also analyzed the memory consumption of the multigrid method and the sparse Cholesky solver, although both methods were optimized more for performance than for memory requirements. The memory consumption of the multigrid method is mainly determined by the meshes representing the different hierarchy levels. In contrast, the memory required for the Cholesky factorization depends significantly on the sparsity of the matrix, too. On the 500k example the multigrid method and the direct solver need about 1GB and 600MB for the Laplacian system, and about 1.1GB and 1.2GB for the bi-Laplacian system. Hence, the direct solver would not be capable of factorizing Laplacian systems of higher order on current PCs, while the multigrid method would succeed.

These comparisons show that direct solvers are a valuable and efficient alternative to multigrid methods even if the linear systems are highly complex. In all our experiments the sparse Cholesky solver was faster than the multigrid method, and if the system has to be solved for multiple right-hand sides, the precomputation of a sparse factorization is even more beneficial.

4 Applications

In this section we finally show several typical computer graphics and geometry processing applications that benefit from the use of sparse direct solvers. Most applications are based on solving Laplacian or bi-Laplacian systems, thus their characteristic behavior for different complexities or different solvers can be transferred from the experiments of the last section. Notice that it is difficult to compare to timings published in original papers on these approaches, since the computational costs depend on hardware factors (e.g., CPU, memory bandwidth), software factors (operating system, compiler), and on the datasets used. Although we tried to pick similar machines, these comparisons should be considered as a rough performance indication only.

Surface Modeling. The first application is freeform modeling or multiresolution modeling [8, 9], which requires to compute (the change of) a smooth base surface by solving bi-Laplacian systems $\Delta_S^2 X = B$ for the x, y, and z coordi-

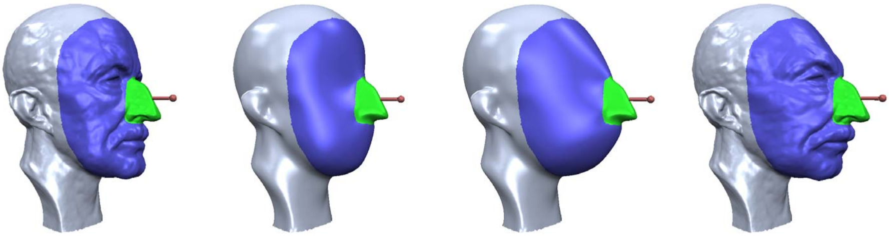


Fig. 3. Multiresolution modeling allows a low-frequency change of the global shape based on the change of a smooth base surface, that is computed by solving a bi-Laplacian system $\Delta_S^2 X = B$

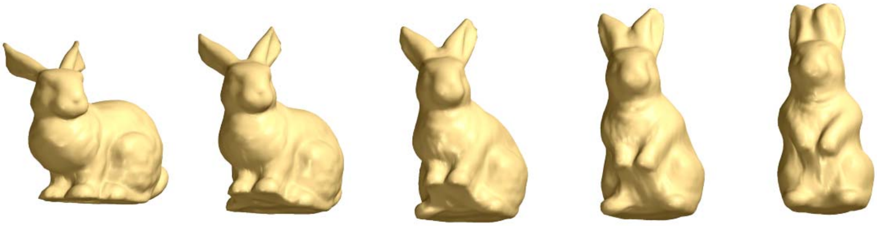


Fig. 4. Mesh morphing of two bunny models based on Poisson shape interpolation. Instead of absolute vertex positions, gradient fields (or Laplace coordinates) are interpolated as $D_t = (1-t)\,D_0 + t\,D_1$, and the vertex positions are derived by solving the Poisson system $\Delta X_t = D_t$ (Image courtesy of Xu et al.)

nates of the unconstrained (*dark/blue*) vertices X (cf. Fig. 3). Each time the designer drags some points on the surface, the boundary constraints change and the linear system has to be solved for another right-hand side in order to compute the deformed surface. As a consequence, these approaches greatly benefit from the sparse factorization solvers. The precomputation of basis functions for the deformation [9] also requires to solve the linear system for several right-hand sides, such that this precomputation gets more efficient, too.

Mesh Morphing. Given two meshes of identical connectivity, morphing between them corresponds to some linear interpolation of their geometry. But instead of using absolute vertex coordinates $\mathbf{x}_i$ for this task, Alexa [13] proposed to represent the meshes by *differential* Laplace coordinates $\mathbf{d}_i := \Delta\mathbf{x}_i$ and to linearly interpolate those instead. In a recent approach, Xu et al. [14] propose a non-linear interpolation of gradient fields, which avoids shrinkage of in-between models. In both cases each morphing step leads to a new set of Laplace vectors $D = (\mathbf{d}_0, \ldots, \mathbf{d}_n)$, from which the vertex positions can be derived by solving $\Delta X = D$. The resulting Laplacian multiple right-hand side problems can again be solved efficiently by sparse Cholesky factorizations.

Implicit Smoothing. In the implicit fairing approach [1] meshes are smoothed by an integration of the PDE $\partial\,\mathbf{x}_i/\partial t = \lambda\Delta_{\mathcal{S}}\mathbf{x}_i$, leading to the so-called *mean curvature flow*. Using semi-implicit integration, this non-linear problem is decomposed into a sequence of linear ones, such that in each time-step the Laplace discretization $\Delta_{X^{(i)}}$ is updated and the Laplacian system $(I - \lambda\Delta_{X^{(i)}})\,X^{(i+1)} = X^{(i)}$ is solved. In this case the matrix re-ordering and the symbolic factorization can be kept and just the numerical factorization and the solution have to be computed. In our experiments this saved 40%-60% of the solver time per iteration.

Conformal Parameterization. Computing a conformal parameterization [6, 7] with fixed boundary vertices requires the solution of a Laplacian system $\Delta_{\mathcal{S}}X = B$ for x and y (cf. Fig. 5, left). In [32] a highly elaborate multigrid

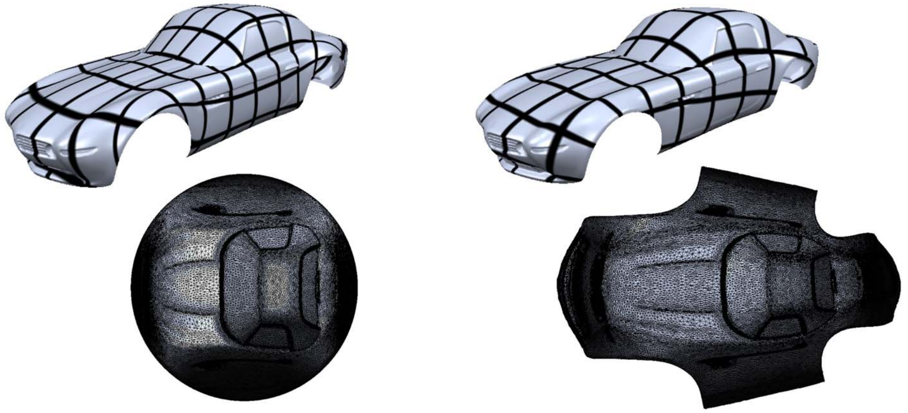


Fig. 5. Two different parameterizations of a car model: discrete conformal parameterization with fixed boundary (*left*), least squares conformal map with free boundary (*right*). Both parameterizations are computed by solving a sparse spd system for the free 2D parameter values associated to the mesh vertices

solver has been derived by evaluating different kinds of multigrid hierarchies and preconditioning strategies. This solver was then used for the parameterization of large meshes, where it takes only 37s for 580k DoFs on a 2.8GHz Pentium4. This time includes loading the system from disk, building the hierarchy, and solving the system for the x coordinate [43]. Our implementation based on the sparse Cholesky solver takes (on a 3.0GHz Pentium4) 28s for for the parameterization of 600k vertices, including matrix setup, re-ordering, factorization, and two solutions.

Least Squares Conformal Maps. In the approach of [18] a conformal parameterization is not computed by minimizing the discrete Dirichlet energy, but instead by solving a system of Cauchy-Riemann equations for each face (cf. Fig. 5, right). Since the number of faces F is about twice the number of vertices V, this system is overdetermined and hence solved in the least squares sense using the normal equations, leading to a spd matrix of dimension $2V \times 2V$, which is similar in structure to a Laplacian matrix. Since the iterative solver used in the original paper [18] was not capable of parameterizing large meshes, the use of multigrid methods was proposed in [33]. On an 1.2GHz Pentium4 their hierarchical approach takes 18s, 31s, and 704s for meshes of 18k, 36k, and 560k vertices, respectively. On a comparable machine (Athlon 1.2GHz) the direct sparse solver is about 4–5 times faster; on the 3.0GHz machine these parameterizations can be computed in 1.4s, 3.2s, and 95s, respectively.

Fluid Dynamics. In Stam's stable fluid approach [17] the Navier-Stokes equations are solved by a four-step procedure in each time step: after updating external forces and advecting the velocity field, a diffusion process considers the

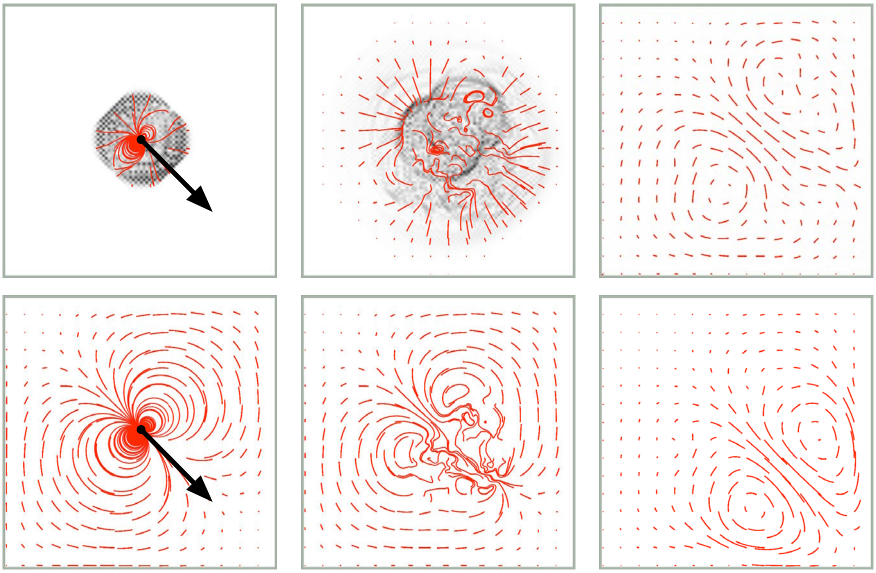


Fig. 6. This example shows a fluid's reaction to a high external force after 1, 3, and 20 time-steps (from left to right) on a 100 × 100 grid. The line segments visualize the velocity field, the background color shows the amount of divergence. A constant number of CG iterations per frame fails to sufficiently propagate the forces and to keep the field free of divergence (*top row*). The sparse Cholesky solver requires a constant time per frame, is significantly faster, and yields correct results independent from the external forces (*bottom row*)

viscosity and a final projection yields a divergence-free velocity field. The last two steps both involve solving a Laplacian system. Since the field is assumed not to change too much from one time-step to the next, the current state yields good starting values, such that in most implementations a fixed small number of CG iterations is used for solving both systems. The break-down of this method in case of high external forces is shown in the top row of Fig. 6. In contrast, the sparse direct solver is twice as fast in this example and yields correct results also for rapidly changing fields (cf. Fig. 6, bottom row).

Poisson Matting. Laplacian systems are also used in image manipulation, like for instance the Poisson matting approach of [16]. A given image I is considered as a composition of a foreground object F and a background B using the *matting equation* $I = \alpha F + (1-\alpha)B$, which is to be solved for the matte $\alpha\,(x,y)$ (cf. Fig. 7). We use a variant of the original approach, where taking the divergence of an approximate gradient of the matting equation leads to the Poisson system $\Delta\alpha = \operatorname{div}(\operatorname{sign}(F-B)\,\nabla I)$. Hence, the computation of the α-matte amounts to solving a spd Laplacian system and therefore benefits from the sparse direct solvers like the other examples.

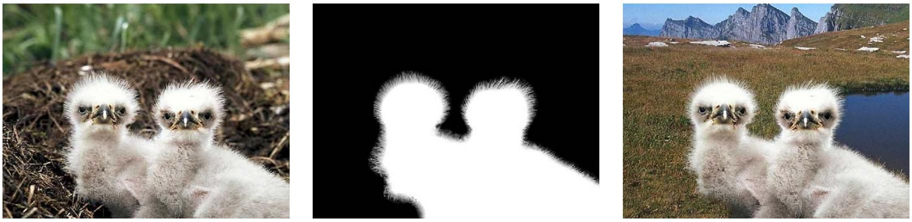


Fig. 7. In order to separate an image I into foreground F and background B, the Poisson matting approach derives an α-matte by solving a Poisson equation $\Delta\alpha = b$

5 Conclusion

In this paper we discussed and compared different classes of linear system solvers for large sparse symmetric positive matrices, and pointed out that sparse direct solvers are a valuable alternative to the usually employed multigrid methods, since they turned out to be more efficient and easier to use in all our experiments.

Although the class of sparse spd matrices seems to be quite restricted, many frequently encountered geometry processing problems over polygonal meshes lead to exactly this kind of systems or can easily be reformulated in this form. As we demonstrated in our experiments, all these applications could benefit considerably from the use of sparse direct solvers.

References

1. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. of ACM SIGGRAPH 99. (1999) 317–324
2. Gill, P.R., Murray, W., Wright, M.: Practical Optimization. Academic Press (1981)
3. Kobbelt, L.: Discrete fairing. In: Proc. on 7th IMA Conference on the Mathematics of Surfaces. (1997) 101–131
4. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In Hege, H.C., Polthier, K., eds.: Visualization and Mathematics III. Springer-Verlag, Heidelberg (2003) 35–57
5. Taubin, G.: A signal processing approach to fair surface design. In: Proc. of ACM SIGGRAPH 95. (1995) 351–358
6. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics **2** (1993) 15–36
7. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic parameterizations of surface meshes. In: Proc. of Eurographics 02. (2002) 209–218
8. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P.: Interactive multi-resolution modeling on arbitrary meshes. In: Proc. of ACM SIGGRAPH 98. (1998) 105–114
9. Botsch, M., Kobbelt, L.: An intuitive framework for real-time freeform modeling. In: Proc. of ACM SIGGRAPH 04. (2004) 630–634
10. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proc. of Eurographics symposium on Geometry Processing 04. (2004) 179–188

11. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with Poisson-based gradient field manipulation. In: Proc. of ACM SIGGRAPH 04. (2004) 644–651
12. Alexa, M.: Local control for mesh morphing. In: Proc. of Shape Modeling International 01. (2001) 209–215
13. Alexa, M.: Differential coordinates for local mesh morphing and deformation. The Visual Computer **19** (2003) 105–114
14. Xu, D., Zhang, H., Wang, Q., Bao, H.: Poisson shape interpolation. In: Proc. of ACM symposium on Solid and Physical Modeling 05. (2005)
15. Ni, X., Garland, M., Hart, J.C.: Fair morse functions for extracting the topological structure of a surface mesh. In: Proc. of ACM SIGGRAPH 04. (2004) 613–622
16. Sun, J., Jia, J., Tang, C.K., Shum, H.Y.: Poisson matting. In: Proc. of ACM SIGGRAPH 04. (2004) 315–321
17. Stam, J.: Stable fluids. In: Proc. of ACM SIGGRAPH 99. (1999) 121–128
18. Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: Proc. of ACM SIGGRAPH 02. (2002) 362–371
19. Demmel, J.W.: Applied numerical linear algebra. SIAM (1997)
20. Meurant, G.A.: Computer solution of large linear systems. Elsevier (1999)
21. Golub, G.H., Loan, C.F.V.: Matrix Computations. Johns Hopkins University Press, Baltimore (1989)
22. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes: The Art of Scientific Computing. 2nd edn. Cambridge University Press (1992)
23. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., der Vorst, H.V.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition. SIAM, Philadelphia, PA (1994)
24. Saad, Y., van der Vorst, H.A.: Iterative solution of linear systems in the 20th century. J. Comput. Appl. Math. **123** (2000) 1–33
25. Hestenes, M., Stiefel, E.: Method of conjugate gradients for solving linear systems. J. Res. Nat. Bur. Stand. **49** (1952) 409–436
26. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University (1994)
27. Hackbusch, W.: Multi-Grid Methods and Applications. Springer Verlag (1986)
28. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial. 2nd edn. SIAM (2000)
29. Bornemann, F.A., Deuflhard, P.: The cascading multigrid method for elliptic problems. Num. Math. **75** (1996) 135–152
30. Kobbelt, L., Campagna, S., Seidel, H.P.: A general framework for mesh decimation. In: Proc. of Graphics Interface 98. (1998) 43–50
31. Garland, M.: Multiresolution modeling: Survey & future opportunities. In: Eurographics State of the Art Report 99. (1999)
32. Aksoylu, B., Khodakovsky, A., Schröder, P.: Multilevel Solvers for Unstructured Surface Meshes. SIAM Journal on Scientific Computing **26** (2005) 1146–1165
33. Ray, N., Levy, B.: Hierarchical Least Squares Conformal Map. In: Proc. of Pacific Graphics 03. (2003) 263–270
34. George, A., Liu, J.W.H.: Computer Solution of Large Sparse Positive Definite Matrices. Prentice Hall (1981)
35. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proc. of the 24th National Conference ACM. (1969) 157–172

36. Liu, J.W.H., Sherman, A.H.: Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. SIAM J. Numerical Analysis **2** (1976) 198–213
37. George, A., Liu, J.W.H.: The evolution of the minimum degree ordering algorithm. SIAM Review **31** (1989) 1–19
38. Liu, J.W.H.: Modification of the minimum-degree algorithm by multiple elimination. ACM Trans. Math. Softw. **11** (1985) 141–153
39. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal of Sci. Comput. **20** (1998) 359–392
40. Toledo, S., Chen, D., Rotkin, V.: Taucs: A library of sparse linear solvers. (http://www.tau.ac.il/∼stoledo/taucs)
41. Demmel, J.W., Eisenstat, S.C., Gilbert, J.R., Li, X.S., Liu, J.W.H.: A supernodal approach to sparse partial pivoting. SIAM Journal on Matrix Analysis and Applications **20** (1999) 720–755
42. Renard, Y., Pommier, J.: Gmm++: a generic template matrix C++ library. (http://www-gmm.insa-toulouse.fr/getfem/gmm_intro)
43. Aksoylu, B.: (personal communication)

Smoothing of Time-Optimal Feedrates for Cartesian CNC Machines

Casey L. Boyadjieff, Rida T. Farouki, and Sebastian D. Timar

Department of Mechanical and Aeronautical Engineering,
University of California, Davis, CA 95616, USA
{clboyadjieff, farouki, sdtimar}@ucdavis.edu

Abstract. Minimum-time traversal of curved paths by Cartesian CNC machines, subject to prescribed bounds on the magnitude of acceleration along each axis, usually involves a "bang-bang" control strategy in which the acceleration bound is realized by one or another of the machine axes at each instant during the motion. For a path specified by a polynomial parametric curve and prescribed acceleration bounds, the time-optimal feedrate may be expressed in terms of a C^0 piecewise-rational function of the curve parameter. This function entails sudden changes in either the identity of the limiting axis, or the sign of acceleration on a single limiting axis, incurring demands for instantaneous changes of motor torque that may not be physically realizable. A scheme is proposed herein to generate smoothed C^1 (slightly sub-optimal) feedrate functions, that incur only finite rates of change of motor torque and remain consistent with the axis acceleration bounds. An implementation on a 3-axis CNC mill driven by an open-architecture software controller is used to illustrate this scheme.

1 Introduction

When a system of bounded motive force is commanded to execute a given spatial path, it is natural to ask: what control strategy, consistent with the motive-force constraints, yields the minimum traversal time? Suppose the path is a straight line, to be traversed by a rocket capable of equal maximum forward and reverse thrust, that starts and ends at rest and is free of external forces. The obvious answer in this case is to accelerate with maximum forward thrust to the midpoint and then to apply maximum reverse thrust for the second half of the traversal. This type of solution is known as "bang-bang" control, since it involves operation at the limits of the motive-force constraints throughout the entire motion, and is characteristic of most time-optimal control schemes [9, 12].

Time-optimal feedrates are of interest in *high-speed machining* [11, 16, 19] where the traversal of strongly-curved paths at high feedrates can cause inertial effects to dominate cutting forces, friction, etc. However, time-optimal feedrates for Cartesian CNC machines subject to prescribed axis acceleration bounds are in general continuous, but non-differentiable at certain *switching points* or *break points* along the curved path. The resulting discontinuities in feed acceleration (rate of change of feedrate) along the curved path imply instantaneous changes

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 84–101, 2005.

in the output torque of the motors that drive the machine axes, which are not physically realizable. The goal of this study is to develop a simple automatic means of smoothing time-optimal feedrates so as to remove the feed acceleration discontinuities, without significantly increasing the overall path traversal time or violating the original axis acceleration constraints.

This paper is organized as follows. Section 2 provides a brief summary of the time-optimal feedrate problem for acceleration-limited Cartesian CNC machines executing paths specified by polynomial parametric curves [18]. The procedure for smoothing the C^0 break points and switching points of these feedrates is then developed in Section 3, together with a method for ensuring that the smoothed feedrate does not violate the original acceleration constraints. In Section 4 we describe the real-time CNC interpolator algorithm for the smoothed feedrate. Finally, Section 5 provides experimental validation of the smoothing process through implementation on a 3-axis open-architecture CNC mill, and Section 6 summarizes our present results and makes some concluding remarks.

2 Time-Optimal Feedrates

The time-optimal feedrate problem was first studied in the context of robotics [2, 13, 14, 15] – typically for systems with revolute joints. In a previous study [18] we have applied these methods to Cartesian CNC machines. This context is attractive, because it admits computation of a closed-form expression for the time-optimal feedrate when the path is specified as a polynomial parametric curve. The differential equation that governs the extremal acceleration phases can be solved analytically, and the break points and switching points delineating such phases can be computed using only a univariate polynomial root solver.

Since it is rather involved, we refer the reader to [18] for complete details[1] of the time-optimal feedrate algorithm. Our focus in this paper is on developing a post-processing step, employed to smooth the tangent-discontinuous points that arise generically in the time-optimal feedrate. The smoothed feedrate, although slightly sub-optimal, is less taxing on the axis drive motors since it does not demand instantaneous changes in the axis motor output torque.

2.1 Path Geometry and Kinematics

Consider a path specified [4] by a degree-n Bézier curve

$$\mathbf{r}(\xi) = \sum_{k=0}^{n} \mathbf{p}_k \binom{n}{k} (1-\xi)^{n-k}\xi^k \,, \qquad \xi \in [0,1]$$

[1] See also [17] for an extension of the algorithm to accommodate finite axis velocity bounds as well as acceleration bounds.

with control points $\mathbf{p}_k = (x_k, y_k, z_k)$ for $k = 0, \ldots, n$. If s denotes cumulative arc length along the curve, the *parametric speed* $\sigma(\xi)$ of this curve is defined by

$$\sigma(\xi) = \frac{\mathrm{d}s}{\mathrm{d}\xi} = |\mathbf{r}'(\xi)|$$

and the unit tangent and normal vectors and curvature are defined by

$$\mathbf{t} = \frac{\mathbf{r}'}{\sigma}, \qquad \mathbf{n} = \frac{\mathbf{r}' \times \mathbf{r}''}{|\mathbf{r}' \times \mathbf{r}''|} \times \mathbf{t}, \qquad \kappa = \frac{|\mathbf{r}' \times \mathbf{r}''|}{\sigma^3}. \tag{1}$$

The feedrate v is the derivative $\mathrm{d}s/\mathrm{d}t$ of arc length with respect to time, while $a = \mathrm{d}v/\mathrm{d}t$ is called the *feed acceleration.* Derivatives with respect to time t and the curve parameter ξ, denoted by dots and primes respectively, are related by

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\mathrm{d}s}{\mathrm{d}t}\frac{\mathrm{d}\xi}{\mathrm{d}s}\frac{\mathrm{d}}{\mathrm{d}\xi} = \frac{v}{\sigma}\frac{\mathrm{d}}{\mathrm{d}\xi}. \tag{2}$$

In terms of v and a, the velocity and acceleration vectors along the curve may be written as

$$\mathbf{v} = \dot{\mathbf{r}} = v\,\mathbf{t}, \qquad \mathbf{a} = \ddot{\mathbf{r}} = a\,\mathbf{t} + \kappa v^2\,\mathbf{n}.$$

Given a feedrate function $v(\xi)$ along the path $\mathbf{r}(\xi)$, the function of the real-time CNC interpolator algorithm within the machine control software is to compute a *reference point* in each sampling interval of the digital controller. The difference between the reference point and the actual machine position, as measured by encoders on the machine axes, constitutes the error signal for the control loop.

2.2 Construction of the Time-optimal Feedrate

Using relations (1) and (2) and noting that $\mathbf{r}' \cdot \mathbf{r}'' = \sigma\sigma'$ allows the acceleration vector to be written as

$$\mathbf{a} = \frac{vv'}{\sigma^2}\,\mathbf{r}' + \frac{v^2}{\sigma^3}\,(\sigma\mathbf{r}'' - \sigma'\mathbf{r}').$$

It is convenient to work with the *square* of the feedrate, which we denote by $q = v^2$. In terms of q and its derivative $q' = 2vv'$ the components of the acceleration along each axis may be expressed as

$$\begin{aligned} a_x &= \tfrac{q'}{2\sigma^2}\,x' + \tfrac{q}{\sigma^3}\,(\sigma x'' - \sigma' x'), \\ a_y &= \tfrac{q'}{2\sigma^2}\,y' + \tfrac{q}{\sigma^3}\,(\sigma y'' - \sigma' y'), \\ a_z &= \tfrac{q'}{2\sigma^2}\,z' + \tfrac{q}{\sigma^3}\,(\sigma z'' - \sigma' z'). \end{aligned} \tag{3}$$

For a given path $\mathbf{r}(\xi)$, the time-optimal feedrate problem consists of determining the function $v(\xi)$ that will minimize the integral

$$T = \int_0^1 \frac{\sigma(\xi)}{v(\xi)}\,\mathrm{d}\xi$$

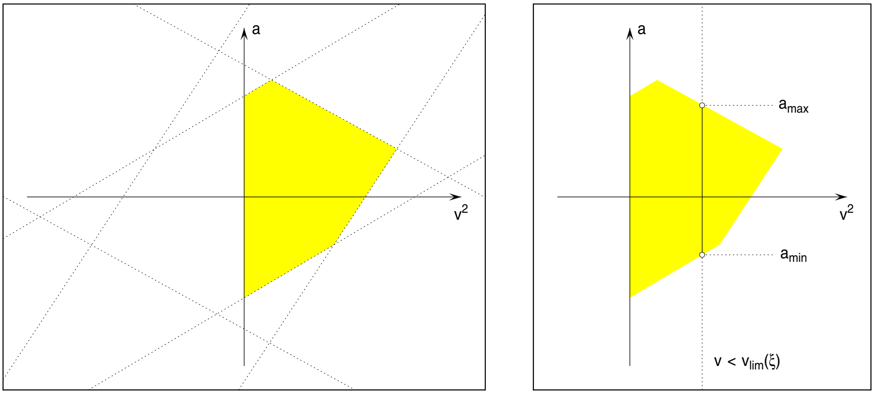


Fig. 1. Left: The constraints (4) define the set of feasible states as the portion of a six-sided parallelogram in the right half of the (v^2, a) plane. Right: Any feedrate lower than the velocity limit $v_{\mathrm{lim}}(\xi)$ yields a range $a_{\min} \le a \le a_{\max}$ of feasible feed accelerations

subject for all $\xi \in [\,0,1\,]$ to the constraints

$$-A_x \le a_x \le +A_x\,, \quad -A_y \le a_y \le +A_y\,, \quad -A_z \le a_x \le +A_z\,, \tag{4}$$

where A_x, A_y, A_z are prescribed acceleration bounds for the machine axes.

Complete details on the solution of this problem were presented in [18]. We content ourselves here with summarizing some key points that are pertinent to the feedrate smoothing problem. Observing that $q' = 2\sigma a$, the three inequalities (4) are linear in q and a, and each defines a strip of feasible states in the (q, a) plane. The intersection of these three strips yields a six-sided parallelogram of feasible (v^2, a) combinations, and of course only the portion in the right half of the plane $(v^2 > 0)$ is physically meaningful (see Figure 1).

At each point of the path $\mathbf{r}(\xi)$, the right-most vertex of the parallelogram shown in Figure 1 defines the maximum possible feedrate $v_{\mathrm{lim}}(\xi)$ consistent with the constraints (4). The graph of $v_{\mathrm{lim}}(\xi)$ in the (ξ, v) plane is the *velocity limit curve* (VLC). This graph is continuous, but not differentiable everywhere – it exhibits slope discontinuities at *critical points*, where there is a change in the identity of the constraints defining the right-most parallelogram vertex.

The construction of the time-optimal feedrate function is performed in the (ξ, q) plane, below the “forbidden region” bounded by the VLC. At points below the VLC, there is a range $[\,a_{\min}, a_{\max}\,]$ of feasible feed accelerations, and in general the optimal feedrate $q(\xi)$ is a piecewise-analytic function whose segments correspond alternately to integrating the differential equations $a = a_{\max}(\xi, v)$ and $a = a_{\min}(\xi, v)$. Transitions between consecutive $a_{\max}$ and $a_{\min}$ phases are called *switching points*, since they signal a change in the identity of the limiting acceleration constraint. In general, some switching points lie on the VLC and others are situated below it. Besides switching points, the time-optimal feedrate generically exhibits other types of break points, corresponding to turning points, inflections, equi-orientation points, and transition points – see [18] for further details and a complete algorithm description.

Consider the form of the time-optimal feedrate function during an extremal acceleration phase. If x is the limiting axis, taking (4) with equality yields

$$\frac{q'}{2\sigma^2}\,x' \,+\, \frac{q}{\sigma^3}\,(\sigma x'' - \sigma' x') \,=\, \alpha_x A_x$$

or

$$q' \,+\, 2\left(\frac{x''}{x'} - \frac{\sigma'}{\sigma}\right) q \,=\, \alpha_x\,\frac{2A_x\sigma^2}{x'}\,, \tag{5}$$

where the quantity $\alpha_x = \pm 1$ identifies $a_{\max}/a_{\min}$ phases. The linear differential equation (5) admits the closed-form solution

$$q \,=\, \left(\frac{\sigma}{x'}\right)^2 (C + 2\alpha_x A_x x)\,, \tag{6}$$

with the integration constant C determined from a known point (ξ_*, q_*) of the trajectory. Similar expressions hold if y or z, rather than x, is the limiting axis.

2.3 Illustrative Example

Figure 2 shows a typical example[2] of the time-optimal feedrate computation for a planar Bézier curve with axis acceleration bounds $A_x = A_y = 10^4$ in/min^2.

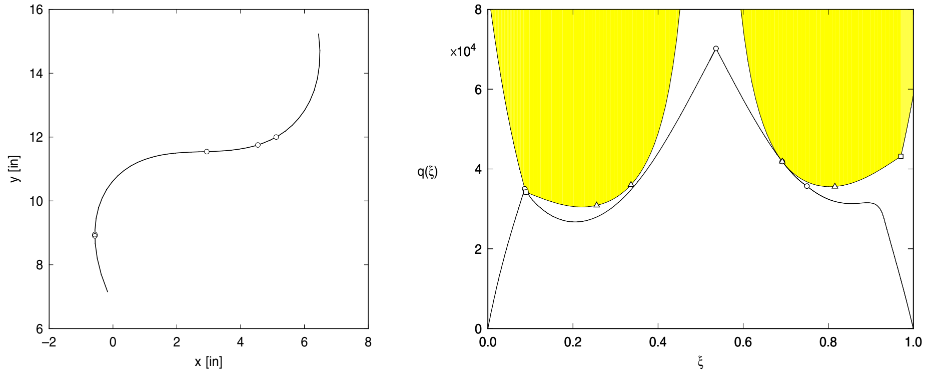


Fig. 2. Test curve (left) and construction of time-optimal feedrate along it (right). The region of infeasible states above the VLC is shown shaded. The time-optimal squared feedrate $q(\xi)$ is a piecewise-rational function with three switching points below the VLC (circles) and two switching points on the VLC (squares denote VLC critical points, and triangles denote tangency points of $a_{\max}$ or $a_{\min}$ trajectories with the VLC)

[2] In keeping with manufacturing practice in the USA, velocities and accelerations are quoted in terms of inches and minutes. For ease of reference, note that 100 in/min = 42.3 mm/sec and $10,000$ in/min^2 = 70.6 mm/sec^2.

The time-optimal feedrate $q(\xi)$ can be represented exactly as a piecewise-rational function with five switching points: one is a critical point on the VLC, one is a tangency of an $a_{\min}/a_{\max}$ trajectory with the VLC, and the remaining three are intersections of $a_{\min}$ and $a_{\max}$ trajectories below the VLC.

3 Smoothing of Time-Optimal Feedrates

To maintain a near-time-optimal form for the smoothed feedrate, we adopt a strategy that inserts a short smoothing segment centered on each C^0 break point or switching point of the exact time-optimal feedrate. With this approach, most of the time-optimal feedrate is left unmodified, and the smoothing segments are designed to meet the unaltered segments with C^1 or C^2 continuity.

To illustrate the need for feedrate smoothing, we show in Figure 4 the time-optimal feedrate for the Bézier curve in Figure 3. This feedrate exhibits two

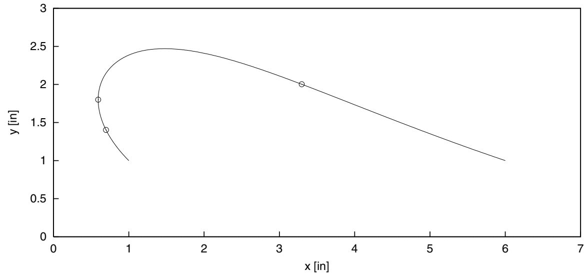


Fig. 3. Quintic Bézier curve with two switching points and an intermediate break point indicated, delineating the three segments of the time-optimal feedrate function

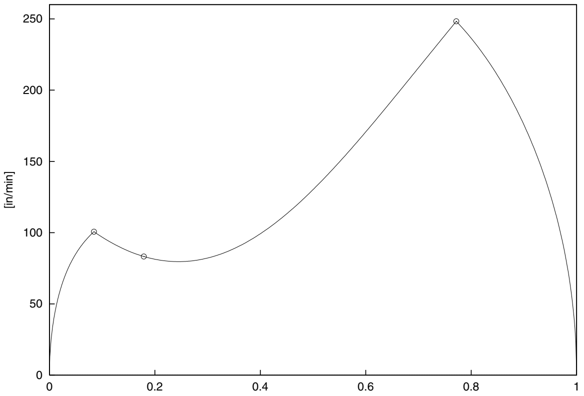


Fig. 4. Time-optimal feedrate v versus curve parameter ξ for the Bézier curve shown in Figure 3, with switching points between feedrate segments denoted by circles

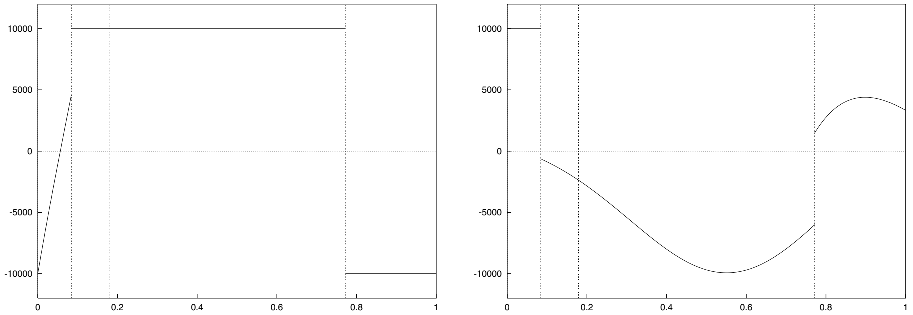


Fig. 5. Accelerations of the x axis (left) and y axis (right) in in/min^2 versus the curve parameter ξ for time-optimal traversal of the curve shown in Figure 3. On the first curve segment, the acceleration is saturated at the upper bound on the y axis, while on the second and third segments it is saturated at the upper and lower bounds on the x axis. Note the acceleration discontinuities in both the x and y axes at the switching points, which demand instantaneous changes in output torque of the axis drive motors

switching points that incur feed acceleration discontinuities, and an intermediate break point defined by a path turning point. Figure 5 indicates the individual x and y axis accelerations resulting from application of the time-optimal feedrate to the specified path. The bang-bang nature of the motion is clearly apparent, with either the x or y axis exhibiting saturation at the prescribed acceleration limits $\pm 10,000$ in/min^2 throughout the entire traversal (for simplicity we employ planar tool paths as examples: the extension to spatial paths is straightforward).

3.1 Form of Feedrate-Smoothing Segments

The first step in the smoothing process is to identify all C^0 break points and switching points ξ_k in the exact time-optimal feedrate, and to define a smoothing interval $[\,\xi_l, \xi_r\,]$ of width $\Delta\xi_k = \xi_r - \xi_l$ centered on each. On this interval, the function $q(\xi)$ giving the square of the exact feedrate is replaced by the form

$$q_s(\xi) \,=\, \frac{\sigma^2(\xi)}{w^2(\xi)} \qquad \text{for } \xi \in [\,\xi_l, \xi_r\,]\,. \tag{7}$$

Here the parametric speed $\sigma(\xi)$ is determined by the curve $\mathbf{r}(\xi)$, while $w(\xi)$ is a polynomial whose coefficients will be used to match the endpoint values and derivatives of the smoothing feedrate segment with the unmodified time-optimal segments at the interval end-points ξ_l, ξ_r. Each C^0 point ξ_k of the time-optimal feedrate has an individually-determined polynomial $w(\xi)$.

Specifying the smoothing function $w(\xi)$ as a quintic allows the modified feed segment to match the unmodified feedrate segments with C^2 continuity (or C^1 continuity with two residual degrees of freedom available for shape adjustment).

This polynomial is defined in Bernstein form using a normalized parameter by

$$w(u) \;=\; \sum_{k=0}^{5} w_k \binom{5}{k} (1-u)^{5-k} u^k \,, \qquad \text{where } u \;=\; \frac{\xi - \xi_l}{\xi_r - \xi_l}\,.$$

The coefficients w_0, w_1 and w_4, w_5 allow for interpolation of end-point values and first derivatives of the time-optimal feed segments, while w_2 and w_3 yield the additional degrees of freedom. These degrees of freedom may be used to ensure that the modified feedrate segment will be traversed in an integer number of time steps of the digital controller.

The specific form (7) of the smoothing element is motivated by the desire to have a closed-form reduction of the integral arising in the real-time interpolator algorithm (described in Section 4 below). In terms of the normalized parameter u, the elapsed time along this segment with the feedrate (7) is

$$t(u) \;=\; \int_0^u \frac{\sigma}{v}\,\mathrm{d}u \;=\; \int_0^u w\,\mathrm{d}u$$

and hence $t(u)$ is simply a polynomial, since w is a polynomial in u.

3.2 Matching End-Point Values and Derivatives

For each switching point ξ_k with smoothing interval $[\,\xi_l, \xi_r\,]$ a unique quintic smoothing function may be obtained by matching the values and first and second derivatives of the smoothing segment to those of the time-optimal feedrate at the interval endpoints ξ_l, ξ_r. The first and second derivatives of the smoothing segment (7) with respect to ξ are

$$\begin{aligned} q' &= \tfrac{2\sigma(\sigma' w - \sigma w')}{w^3} \\ q'' &= \tfrac{2(\sigma\sigma'' + \sigma'^2)}{w^2} \;-\; \tfrac{2(\sigma^2 w'' + 4\sigma\sigma' w')}{w^3} \;+\; \tfrac{6\sigma^2 w'^2}{w^4}\,. \end{aligned} \tag{8}$$

Re-arranging (7) and (8) to write w, w', w'' on the left then gives

$$\begin{aligned} w &= \tfrac{\sigma}{\sqrt{q}}\,, \\ w' &= \tfrac{2\sigma\sigma' w - q' w^3}{2\sigma^2}\,, \\ w'' &= \tfrac{3w'^2}{w} \;-\; \tfrac{4\sigma' w'}{\sigma} \;+\; \tfrac{2w(\sigma\sigma'' + \sigma'^2) - q'' w^3}{2\sigma^2}\,. \end{aligned} \tag{9}$$

Taking first and second derivatives of $w(u)$ and relating $u \in [\,0,1\,]$ to $\xi \in [\,\xi_l, \xi_r\,]$ then yields the following expressions for the coefficients of $w(u)$:

$$\begin{aligned} w_0 &= w(\xi_l)\,, \quad w_1 = w_0 + \tfrac{\Delta\xi_k\, w'(\xi_l)}{5}\,, \quad w_2 = 2w_1 - w_0 + \tfrac{(\Delta\xi_k)^2\, w''(\xi_l)}{20}\,, \\ w_5 &= w(\xi_r)\,, \quad w_4 = w_5 - \tfrac{\Delta\xi_k\, w'(\xi_r)}{5}\,, \quad w_3 = 2w_4 - w_5 + \tfrac{(\Delta\xi_k)^2\, w''(\xi_r)}{20}\,. \end{aligned}$$

The factors of $\Delta\xi_k$ and $(\Delta\xi_k)^2$ arise from the fact that

$$\frac{\mathrm{d}w}{\mathrm{d}\xi} \;=\; \frac{1}{\Delta\xi_k}\,\frac{\mathrm{d}w}{\mathrm{d}u}\,.$$

3.3 Adjustment for Integer Number of Time Steps

Because CNC machine controllers employ digital time sampling, it is desirable to adjust the smoothing function slightly to ensure that the smoothing segment may be traversed in an integer number of time steps Δt. This can be accomplished while maintaining C^1 continuity between the smoothed and unmodified feedrate segments by adjusting only the middle two coefficients w_2, w_3 of the smoothing function. The total traversal time T for the smoothed feedrate segment is

$$T \,=\, \Delta\xi_k \int_0^1 w \,\mathrm{d}u \,=\, \frac{\Delta\xi_k}{6} \sum_{k=0}^{5} w_k \,. \tag{10}$$

The total time is divided by the sampling interval Δt to give the nominal number of steps needed for the smoothing segment. This number is then rounded up to the nearest integer value N, and a new traversal time T_{new} corresponding to an integer number of time steps is defined by $T_{\text{new}} = N\Delta t$.

Multiplying the middle two control points of (10) by a scaling factor k and equating with T_{new} gives

$$T_{\text{new}} \,=\, \frac{\Delta\xi_k}{6} \left[\, w_0 + w_1 + k(w_2 + w_3) + w_4 + w_5 \,\right],$$

and hence the appropriate value of the scaling factor is

$$k \,=\, \frac{6\,T_{\text{new}}/\Delta\xi_k - (w_0 + w_1 + w_4 + w_5)}{w_2 + w_3} \,. \tag{11}$$

Once the coefficients w_2 and w_3 have been multiplied by k, the resulting time function will correspond to an integer number of time steps for traversal of the smoothed feedrate segment.

3.4 Compatibility with Acceleration Constraints

Using a smoothed feedrate segment of the form (7) with end-point values and derivatives matched to those of the unmodified time-optimal feedrate does not automatically guarantee that the smoothed segment will be compatible with the axis acceleration constraints (4). Requiring the smoothed feedrate to be less than or equal to the original time-optimal feedrate does not *per se* ensure the satisfaction of these constraints. Thus, an additional step is required to check that the smoothed feedrate does not violate these constraints.

For the smoothing feedrate segment (7) the axis accelerations (3) are

$$a_x \,=\, \frac{wx'' - w'x'}{w^3} \,, \qquad a_y \,=\, \frac{wy'' - w'y'}{w^3} \,. \tag{12}$$

Applying the smoothing algorithm described above to a variety of time-optimal feedrates and evaluating (12) indicates that in most cases the modified feedrate segments do satisfy the acceleration constraints, but in some cases the size of the smoothing interval $\Delta\xi_k$ may require adjustment.

Starting with a nominal smoothing interval of $\Delta\xi_k = 0.08$, and incrementally reducing this interval if violations of the acceleration constraints occur, appears to be a satisfactory approach to satisfying the constraints in all cases. A lower bound on $\Delta\xi_k$ (and the corresponding number of time steps) is imposed by the need for an integer number of time steps. Rounding up for an integer number of time steps has little effect when the total number of time steps is large, but rounding a segment up from a low number of steps can cause an oscillatory smoothed feedrate, incurring acceleration constraint violations. To preserve a near-time-optimal form for the modified feedrate, an upper bound on $\Delta\xi_k$ must also be imposed so that the smoothing segments account for a relatively small fraction of the overall feedrate profile. Assuming equal, symmetric acceleration constraints ($A_x = A_y = A,$ say) gives

$$-A \le \frac{wx'' - w'x'}{w^3} \le +A \qquad \text{and} \qquad -A \le \frac{wy'' - w'y'}{w^3} \le +A\,,$$

or

$$\begin{aligned} wx'' - w'x' - w^3 A &\le 0\,, \qquad & wy'' - w'y' - w^3 A &\le 0\,, \\ wx'' - w'x' + w^3 A &\ge 0\,, \qquad & wy'' - w'y' + w^3 A &\ge 0\,, \end{aligned} \tag{13}$$

for the axis acceleration bounds on a smoothed feedrate segment.

All four of the inequalities (13) must be satisfied to ensure that there are no acceleration constraint violations. To verify this, the coefficients of each of the four polynomials on the left are examined. If all the coefficients are of the same sign, and are greater than or equal to zero or less than or equal to zero, as appropriate, the constraints are satisfied. However, coefficients of mixed sign do not necessarily signal a constraint violation over the smoothing interval. In the case of mixed-sign coefficients, a Sturm sequence [8, 21] may be used to verify if the bounding function has any roots within the smoothing interval. If no roots are indicated, the bound is satisfied. Otherwise, a shorter smoothing interval must be used to avoid violation of an acceleration constraint.

3.5 Smoothing Example

To illustrate the smoothing process, a step-by-step smoothing of the time-optimal feedrate shown in Figure 7, for the path in Figure 6, is described below. In this example, there are four switching points that require feedrate smoothing and an additional three break points (corresponding to path turning points) that do not require smoothing. For simplicity, a single smoothing interval $\Delta\xi = 0.06$ will be used, centered about each of the four switching points ξ_k. Each of these points is smoothed individually as follows.

1. Starting with the left-most switching point to be smoothed and referring to it as ξ_1, the left and right smoothing interval endpoints ξ_l and ξ_r are simply $\xi_l = \xi_1 - \frac{1}{2}\Delta\xi$ and $\xi_r = \xi_1 + \frac{1}{2}\Delta\xi$.
2. The squared time-optimal feedrate $q(\xi)$ along with its derivatives $q'(\xi)$, $q''(\xi)$ and the parametric speed $\sigma(\xi)$ and its derivatives $\sigma'(\xi)$, $\sigma''(\xi)$ are evaluated at ξ_l and ξ_r for use in (9) to determine values for $w(\xi)$, $w'(\xi)$, $w''(\xi)$ at ξ_l and

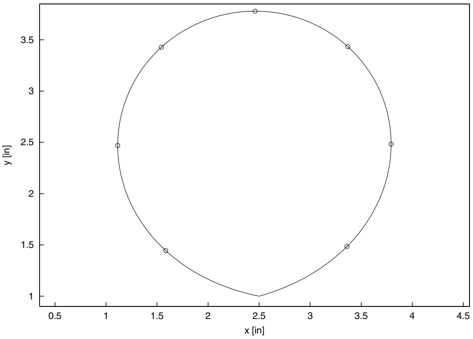


Fig. 6. Quintic Bézier curve with break points and switching points between feedrate segments indicated by circles. The curve starts and ends at the point $(x, y) = (2.5, 1.0)$

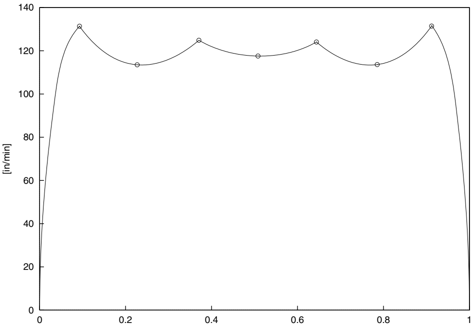


Fig. 7. The time-optimal feedrate v (for acceleration bounds $A = \pm 10,000$ in/min^2 on both axes) versus the parameter ξ along the curve in Figure 6, with break points and switching points indicated by circles

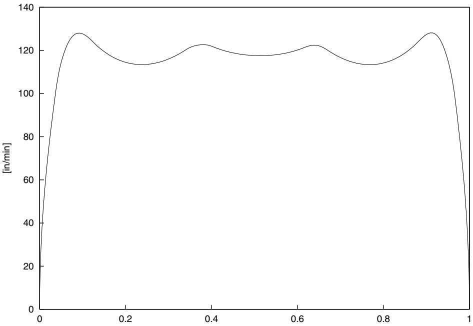


Fig. 8. Smoothed time-optimal feedrate v versus curve parameter ξ for the curve shown in Figure 6 with smoothing intervals $\Delta\xi = 0.06$

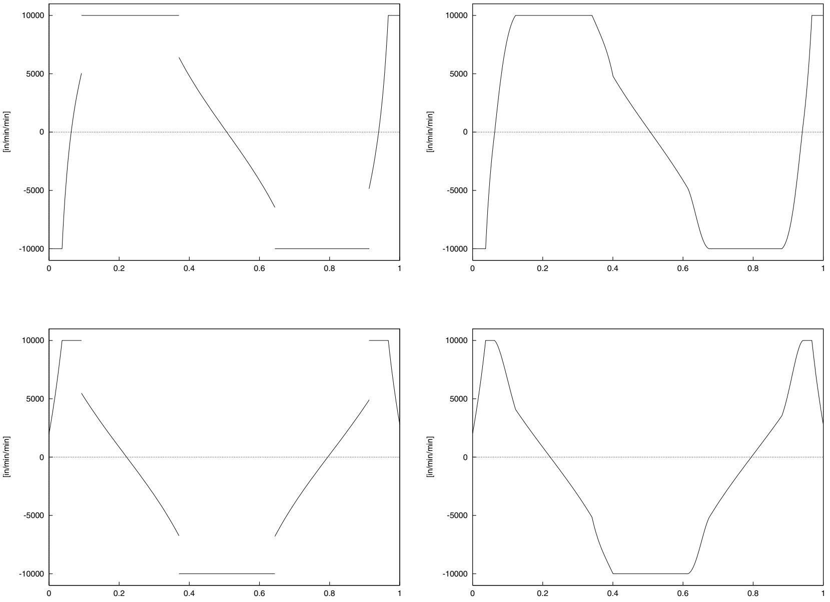


Fig. 9. Comparison of individual axis accelerations for the x (upper) and y (lower) axes versus the curve parameter ξ, corresponding to the original time-optimal feedrate shown in Figure 7 (left) and the smoothed feedrate shown in Figure 8 (right). The acceleration discontinuities incurred by the original feedrate have been eliminated with the smoothed feedrate, but the acceleration bounds $\pm 10,000$ in/min^2 are still satisfied

ξ_r. With the desired values of w and its derivatives known at the smoothing interval endpoints, coefficients for the smoothing function $w(u)$, defined on $u \in [\,0,1\,]$, are determined.

3. The function $w(u)$ is integrated over $[\,0,1\,]$ to obtain the traversal time T for the smoothing segment, and a new traversal time $T_{\rm new}$ corresponding to an integer number of sampling intervals $\varDelta t$ is obtained from $T_{\rm new} = \lceil T/\varDelta t\rceil \, \varDelta t$. The time factor k is given by (11) and once w_2 and w_3 have been multiplied by k, we are assured that the smoothed segment will be completed in a whole number of time steps of the digital controller.
4. The smoothed feedrate segment is checked for violations of the acceleration constraint, as discussed in the preceding section, and adjusted as necessary if the constraints are not immediately satisfied.

Figure 8 shows the resulting feedrate after all the derivative discontinuities have been removed. Figure 9 compares the accelerations of the x and y axes corresponding to the original time-optimal feedrate (Figure 7) and the smoothed feedrate (Figure 8). The acceleration discontinuities apparent in the former have clearly been eliminated in the latter, while maintaining consistency with the prescribed acceleration bounds.

4 Real-Time CNC Interpolator

The role of the real-time intepolator in a CNC system is to compute, at each sampling time $t_j = j\Delta t$ of the digital controller, a reference point $\mathbf{r}(\xi_j)$ along the curve in accordance with the specified feedrate variation. The actual machine position at time t_j (as measured by encoders on the machine axes) is compared with this reference point in order to generate a control signal for the axis drive motors. The elapsed time t is related to the curve parameter ξ through the interpolation integral, defined by

$$t(\xi_j) \,=\, j\Delta t \,=\, \int_0^{\xi_j} \frac{\sigma}{v}\,\mathrm{d}\xi\,. \tag{14}$$

Note that the unknown ξ_j in equation (14) is the *upper limit of integration.* In order to have an efficient and accurate real-time interpolator, capable of accommodating a variable feedrate v, it is desirable that the above integral have a simple closed-form reduction. For further background on real-time CNC interpolator algorithms, see [3, 5, 6, 7, 10, 20, 22]

If the function $t(\xi)$ has a simple closed-form expression, we can solve the equation $t(\xi_j) = j\Delta t$ by a few Newton-Raphson iterations, using the preceding reference-point parameter value ξ_{j-1} as a starting approximation. Note that $t(\xi)$, being the integral of a positive function, is monotone-increasing. Fortunately, a closed-form reduction of the interpolation integral is possible for both the unmodified time-optimal feedrate segments and the smoothing segments.

For the form (6) of the squared time-optimal feedrate (assuming that x is the acceleration-limited axis), substituting into (14) gives

$$t(\xi_j) \,=\, \int_0^{\xi_j} \frac{x'}{\sqrt{C + 2\alpha_x A_x x}}\,\mathrm{d}\xi \,=\, \frac{\sqrt{C + 2\alpha_x A_x x(\xi_j)}}{A_x} + K\,,$$

where the integration constant K is determined by the condition $\xi = 0$ at $t = 0$. Similarly, for a smoothing feedrate segment of the form (7) we obtain

$$t(\xi_j) \,=\, \int_0^{\xi_j} w\,\mathrm{d}\xi\,,$$

and since w is a polynomial, t is a polynomial of degree one higher. It is defined on the normalized interval $u \in [\,0, 1\,]$ in Bernstein form

$$t(u) \,=\, \sum_{k=0}^{6} t_k \binom{n}{k} (1-u)^{6-k} u^k.$$

by the coefficients $t_0 = 0$ and

$$t_k \,=\, t_{k-1} + \frac{w_{k-1}}{6} \qquad \text{for } k = 1, \ldots, 6\,.$$

5 Experimental Implementation

5.1 Open-Architecture CNC Milling Machine

The machine used in the experiments is a MillRight Compact Series 18 3-axis CNC milling machine manufactured by MHO Corporation (Figure 10), operating with the OpenCNC controller developed by Manufacturing Data Systems, Inc. Rather than the usual "black-box" type CNC controller, this system follows the open-architecture principle, allowing the user full access to all the functions and data that control the machine. In particular, custom real-time interpolators can be substituted for the standard G code interpolators. Precision ground ball-screws powered by brushless DC motors drive the three machine axes, with shaft mounted encoders providing position feedback and allowing for data collection. The digital controller operates with a sampling frequency $f = 1024$ Hz (sampling interval $\Delta t = 1/f \approx 0.001$ sec) and a single processor provides the user interface, controls the motors, and interpolates the tool paths in real time.

Tool paths to be executed are transmitted to the machine in the form of *part program* files, which are read by a pre-processor that computes geometric and feedrate information prior to invoking the appropriate real-time interpolator for the type of path and feedrate to be executed. In the present context, the machine is driven directly from the analytic description of the path as a Bézier curve, and the corresponding smoothed time-optimal feedrate functions are specified in terms of piecewise-rational functions. Unlike common practice in NC part pro-

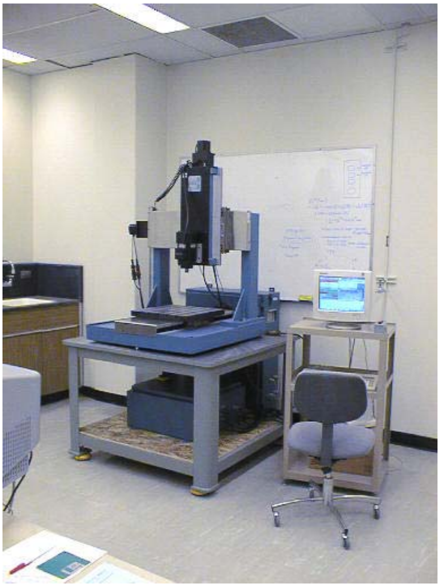


Fig. 10. The MHO 3-axis mill with OpenCNC open-architecture software controller used for the time-optimal feedrate smoothing experiments

grams, it is not necessary to approximate the path by piecewise-linear/circular G codes.

5.2 Results

Figure 11 shows the measured time-optimal feedrate corresponding to an actual run of the machine along the Bézier curve shown in Figure 3 with acceleration bounds ±10, 000 in/min^2 on each axis. Axis location data is measured by the encoders in each sampling interval of the digital controller, and axis velocities are computed *a posteriori* from the saved position data by first-order differencing. The feedrate is computed as the magnitude of the vector whose components are defined by the individual axis velocities. The measured feedrate profile seen in Figure 11 differs somewhat from the “theoretical” profile in Figure 4, primarily due to the fact it is plotted against the elapsed time t rather than the curve parameter ξ (these variables are related by $\mathrm{d}\xi/\mathrm{d}t = v/\sigma$).

Figure 12 shows measured feedrate data from runs with the smoothed time-optimal feedrate on the same curve (Figure 3), with smoothing intervals $\Delta\xi = 0.04$ and 0.08. The traversal time for the unsmoothed feedrate was 3688 time steps Δt, whereas the smoothed feedrates required an additional 3 time steps and 7 time steps, respectively – these amount to only 0.08% and 0.19% increases in the total traversal time for the smoothed feedrates, a modest price to pay

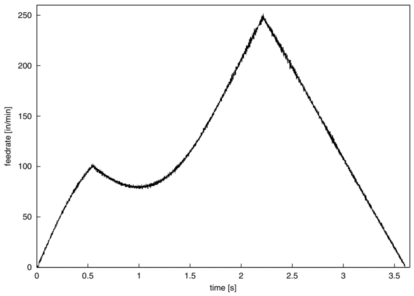


Fig. 11. Measured time-optimal feedrate for curve shown in Figure 3

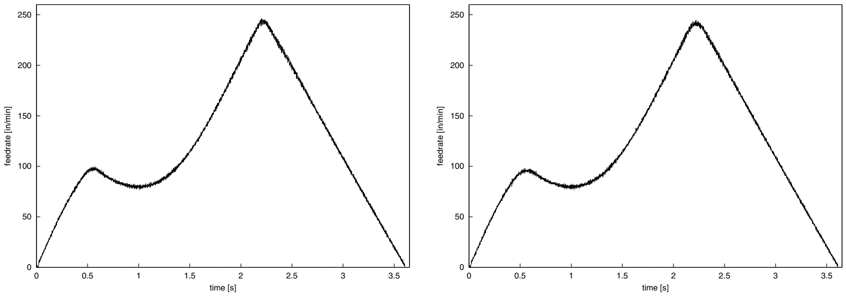


Fig. 12. Smoothed feedrate for curve in Figure 3 with $\Delta\xi = 0.04$ (left) and 0.08 (right)

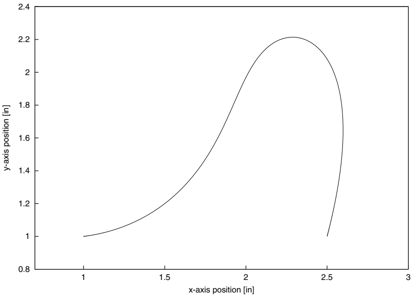


Fig. 13. Another quintic Bézier test curve for time-optimal traversal

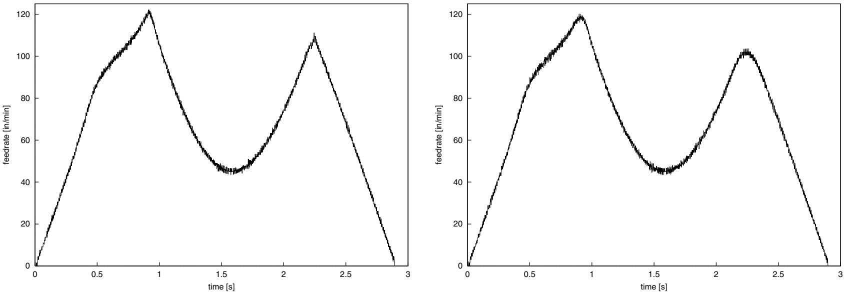


Fig. 14. Time-optimal feedrate (left) for the curve shown in Figure 13, and smoothed feedrate (right) using the smoothing interval $\Delta\xi = 0.08$

for effectively eliminating the derivative discontinuities from the time-optimal feedrate (and consequent abrupt changes in motor torque output).

Finally, Figures 13-14 show another Bézier curve and corresponding feedrates as determined from the encoder data. In this case, the unsmoothed feedrate requires 2963 time steps, and using a smoothing interval of $\Delta\xi = 0.08$ requires an additional 6 time steps (a 0.20% increase in traversal time) while eliminating the derivative discontinuities.

6 Closure

The implementation of time-optimal control on Cartesian machines with fixed acceleration bounds on each axis is particularly attractive, since this problem admits an essentially closed-form solution given a univariate polynomial root-solver to identify the feedrate break points and switching points. For a path defined by a polynomial curve $\mathbf{r}(\xi)$, the square of the time-optimal feedrate may be determined as a piecewise-rational function $q(\xi)$ of the curve parameter. Furthermore, this feedrate may be realized by a real-time CNC interpolator that works directly off the analytic curve description, obviating the need to invoke cumbersome and problematic G code [1] approximations.

The formal solution to the time-optimal feedrate problem is a C^0 function, that exhibits derivative discontinuities at certain points. Use of this function incurs instantaneous changes in the torque demand imposed on the axis drive motors, which is physically unrealizable and may cause damage. In this paper we propose a simple scheme to remove the sudden jumps in acceleration incurred by the formal time-optimal feedrate solution. This yields a more practical, smoother motion with only a very modest (typically $< 1\%$) increase in the overall path traversal time. The smoothed feedrate coincides with the exact time-optimal feedrate over most of its extent, and the form of the smoothing elements is designed to facilitate a simple real-time CNC interpolator algorithm.

References

1. EIA Standard RS–274–D (1979), Interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines, Electronic Industries Association, Engineering Dept., Washington, D.C.
2. J. E. Bobrow, S. Dubowsky, and J. S. Gibson (1985), Time–optimal control of robotic manipulators along specified paths, *International Journal of Robotics Research* **4** (3), 3–17.
3. J.–J. Chou and D. C. H. Yang (1991), Command generation for three–axis CNC machining, *ASME Journal of Engineering for Industry* **113** (August), 305–310.
4. G. Farin (1997), *Curves and Surfaces for Computer Aided Geometric Design* (4th Edition), Academic Press, San Diego.
5. R. T. Farouki, J. Manjunathaiah, D. Nicholas, G.–F. Yuan, and S. Jee (1998), Variable feedrate CNC interpolators for constant material removal rates along Pythagorean–hodograph curves, *Computer Aided Design* **30**, 631–640.
6. R. T. Farouki and S. Shah (1996), Real–time CNC interpolators for Pythagorean–hodograph curves, *Computer Aided Geometric Design* **13**, 583–600.
7. R. T. Farouki and Y–F. Tsai (2001), Exact Taylor series coefficients for variable–feedrate CNC curve interpolators, *Computer Aided Design* **33**, 155–165.
8. C. Florian (1969), *An Introduction to the Theory of Equations*, Macmillan Company, New York.
9. H. Halkin (1965), A generalization of LaSalle's "bang–bang" principle, *SIAM Journal on Control* **2**, 199–202.
10. J.–T. Huang and D. C. H. Yang (1992), A generalized interpolator for command generation of parametric curves in computer–controlled machines, Proceedings, *Japan/USA Symposium on Flexible Automation*, Vol. 1, ASME, 393–399.
11. R. Komanduri, K. Subramanian, and B. F. von Turkovich (eds.) (1984), *High Speed Machining*, PED–Vol. 12, ASME, New York.
12. J. P. LaSalle (1960), The time optimal control problem, in *Contributions to the Theory of Nonlinear Oscillations* (L. Cesari, J. P. LaSalle, and S. Lefschetz, eds.), Vol. 5, Princeton Univ. Press.
13. F. Pfeiffer and R. Johanni (1987), A concept for manipulator trajectory planning, *IEEE Journal of Robotics and Automation* **RA–3** (2), 115–123.
14. Z. Shiller and H. H. Lu (1990), Robust computation of path constrained time optimal motions, Proceedings, *IEEE International Conference on Robotics and Automation*, Cincinnati, OH, 144–149.

15. J. J. E. Slotine and H. S. Yang (1989), Improving the efficiency of time–optimal path–following algorithms, *IEEE Transaction on Robotics and Automation* **5** (1), 118–124.
16. S. Smith and J. Tlusty (1997), Current trends in high–speed machining, *ASME Journal of Manufacturing Science and Engineering* **119**, 664–666.
17. S. D. Timar, R. T. Farouki, and C. L. Boyadjieff (2005), Time-optimal feedrates along curved paths for Cartesian CNC machines with prescribed bounds on axis velocities and accelerations, preprint.
18. S. D. Timar, R. T. Farouki, T. S. Smith, and C. L. Boyadjieff (2005), Algorithms for time-optimal control of CNC machines along curved tool paths, *Robotics and Computer-Integrated Manufacturing* **21** (1), 37–53.
19. J. Tlusty (1993), High–speed machining, *CIRP Annals* **42**, 733–738.
20. Y–F. Tsai, R. T. Farouki, and B. Feldman (2001), Performance analysis of CNC interpolators for time–dependent feedrates along PH curves, *Computer Aided Geometric Design* **18**, 245–265.
21. J. V. Uspensky (1948), *Theory of Equations*, McGraw-Hill, New York.
22. D. C. H. Yang and T. Kong (1994), Parametric interpolator versus linear interpolator for precision CNC machining, *Computer Aided Design* **26**, 225–234.

Plausible 3D Colour Surface Completion Using Non-parametric Techniques

Toby P. Breckon and Robert B. Fisher

Institute for Perception, Action & Behaviour,
School of Informatics, University of Edinburgh, UK
toby.breckon@ed.ac.uk, rbf@inf.ed.ac.uk

Abstract. We consider the combined completion of 3D surface relief and colour for the hidden and missing portions of objects captured with $2\frac{1}{2}D$ (or 3D) capture techniques. Through an extension of non-parametric texture synthesis to facilitate the completion of localised 3D surface structure (relief) over an underlying geometric surface completion we achieve realistic, *plausible completion*[1] and extension of $2\frac{1}{2}D$ partially visible surfaces. Additionally we show how this technique can be extended to the completion of increasingly available colour $2\frac{1}{2}$D / 3D range data.

1 Introduction

3D data acquisition techniques in computer vision, such as range scanning and stereo photography, suffer from the common problem of their $2\frac{1}{2}D$ sensing limitation—meaning the back-facing or occluded portions of surfaces within a scene cannot be realised from a uni-directional capture. As a result capturing a complete object in 3D can involve the time-consuming process of multi-directional capture and subsequent data fusion and registration [1, 2]. Often, despite multi-directional capture some small regions of the object remain uncaptured, giving rise to the need for hole-filling techniques to produce a completed 3D model [3].

To date, work on this problem has been primarily limited to the completion of smooth surface continuation in small missing surface patches [4, 3, 5, 6, 7, 8] or the completion of geometrically conforming shapes through the use of shape fitting and parameterisation [9, 10, 11, 12, 13]. Such techniques, although valid, limit truly plausible surface completion (i.e. cases where the original surface portion and the completion are indistinguishable) to a smooth or geometrically conforming subset of all real-world occurring surfaces.

Contrastingly, here we consider the localised completion of *real* 3D surfaces in terms of both its 3D surface structure (relief) and colour detail. Our approach has two parts: firstly we assume the underlying surface can itself be completed

[1] By plausible, we mean passable to the viewer as if it were original and indistinguishable from any existing original part.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 102–120, 2005.

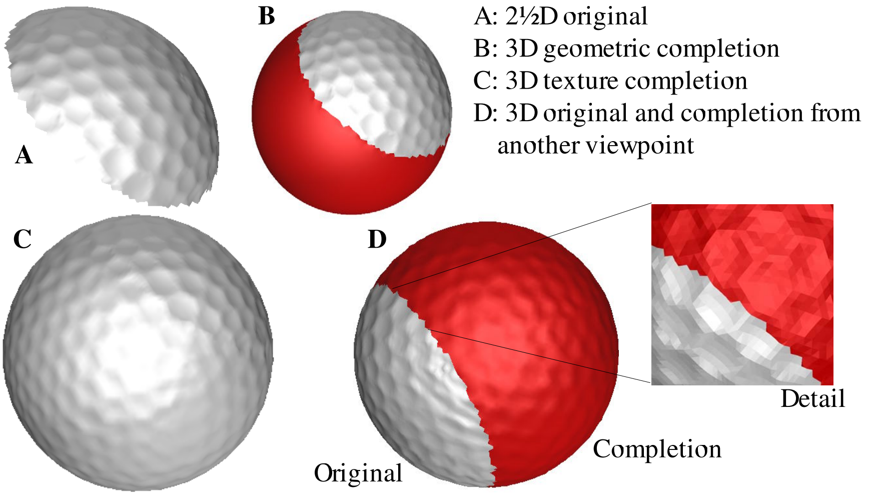

A: 2½D original
B: 3D geometric completion
C: 3D texture completion
D: 3D original and completion from another viewpoint
A
B
C
D
Detail
Completion
Original

Fig. 1. Completion of a $2\frac{1}{2}$D golfball

using one of these prior smooth completion techniques (1); over which the localised relief and colour completion is then achieved through the propagation of knowledge from the original to the unknown surface portion (2).

As an example, we complete both the geometric sphere and surface dimples of a $2\frac{1}{2}$D golfball as shown in Fig. 1. Here we see the successful plausible completion of the surface relief pattern (Fig. 1(C,D)) over a geometric completion (Fig. 1(B)) of the original $2\frac{1}{2}$D capture (Fig. 1(A)).

Concurrent work [14] has also considered a similar approach based on propagating 3D surface patches from visible to unknown surface portions. However, as shown in [14], this patched based approach relies on the existence of suitable propagatable patches in the original surface portion. Although computationally more expensive, the fine-detailed per-{point—vertex—range sample} based approach proposed here does not suffer this limitation. Instead it lends itself well to the propagation of both tile-able surface textures (see Figs. 1 and 10) and the completion/extension of more stochastic surface textures (see Figs. 11 and 12) derived from the original—without any apparent 'tiling' or similar repetitive artifacts. In addition our combination of both localised 3D structure and colour completion extends beyond that of [14].

Considerable recent interest in the texturing of 3D objects with 2D colour textures is also of note (e.g. [15, 16, 17, 18, 19])—in some cases to give *illusion* of a real 3D surface displacement texture [18, 19] or merely to enhance the 3D appearance of a surface in a given rendering [15, 16, 17]. Although related in some aspects, many of these approaches fall short of the problem we pose here, concentrating instead on the consistent mapping of often fairly arbitrary, synthetic textures onto relatively smooth 3D surface. The main advantage of such smooth surfaces is the ease of which consistent localised reference frames can be derived automatically [17, 18] or specified from user input [15, 16, 19] to facilitate the orientation of a 2D texture over the 3D surface.

Related work has also considered aspects of geometric texture transfer - the semi-automated transfer of surface relief from one geometric object to another [20, 21, 22, 23, 24]. Although addressing a similar problem a number of issues, in addition to the techniques utilised, differentiate it from the approach we consider here: the use of smooth surfaces [20, 21, 22], synthetic relief textures [21, 23], the overall lack of relief/colour integration and in some cases a degree of user interaction [20, 22].

In contrast, we consider the completion and extension of existing non-smooth surfaces containing both highly regular 3D textures (e.g. Figs. 1 and 10) and more stochastic natural textures (e.g. Figs. 9 and 11) in terms of both *real* 3D surface displacement texture (relief) and colour. Our overall aim being the *plausible completion* and extension of a given real-world surface sample so that seamless *good continuation* exists between the original and the completion.

In this paper we combine our prior work in completion of 3D surface relief [25] with an additional aspect of colour completion. Firstly we outline and present our background work in this area [25] together with some results and then detail its recent extension to dual surface structure / colour completion.

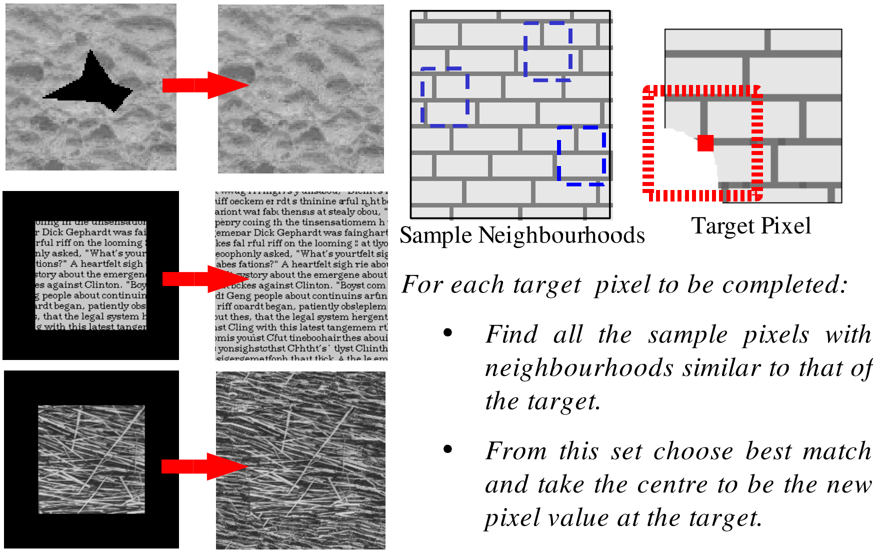
2 2D Non-parametric Texture Synthesis

Non-parametric sampling, lying at the heart of our approach, was proposed as a method for texture synthesis in 2D images based on using a statistical non-parametric model and an assumption of spatial locality [26]. Unlike other approaches in the texture synthesis arena (e.g. [27, 28]) which attempt to explicitly model the texture prior to synthesis, this approach samples directly from the texture sample itself—a kind of implicit modelling akin to the robotics paradigm ‘the world is its own best model’. As a result it is “very powerful at capturing statistical processes for which a good model hasn’t been found” [26] and thus highly suited to our work in 3D.

In 2D operation non-parametric sampling is very simple—it successively grows a texture outwards from an initial seed area, one pixel at a time, based on finding the pixel neighbourhood in the sample image that best matches that of the current target pixel (i.e. the one being synthesised). It then uses the central pixel’s value as the new value for the target (see Fig. 2).

Matching is based upon using the normalised sum of squared difference metric (SSD) between two pixel neighbourhoods (i.e. the textured pixels surrounding the target and those surrounding each sample pixel). A 2D Gaussian kernel is used across the neighbourhood to assign weights reflecting pixel influence in inverse proportion to distance from the target.

The neighbourhoods are defined as $W \times W$ square windows around each pixel where W, window size, is a parameter perceptually linked to the scale of the largest regular feature present in the texture [26]. From the set of all sample neighbourhoods, the top $\eta\%$ of matches are selected as those with the lowest



For each target pixel to be completed:

- *Find all the sample pixels with neighbourhoods similar to that of the target.*
- *From this set choose best match and take the centre to be the new pixel value at the target.*

Fig. 2. 2D non-parametric texture synthesis

SSD values from which one is then randomly selected to provide the value at the target. As an additional constraint the randomly selected match is only used to fill the target provided it has a normalised SSD value less than a specified error threshold, e, related to the acceptable level of noise in the synthesised texture—a factor directly related to that present in the original sample. Here in our 3D approach, as in the original 2D work, we set $\eta = 10$.

3 3D Non-parametric Completion

Here we review our initial technique for the completion of 3D surface relief from [25]. The extension to combined relief/colour completion is detailed further on in this paper.

The basic aspects of non-parametric sampling map well from 2D to 3D: the 2D image becomes a 3D surface, the individual pixel becomes a point on that surface, a pixel neighbourhood becomes the set of nearest neighbours to a surface point and the actual pixel values being synthesised become displacement vectors mapping discrete points on a textured surface to the geometric surface derived from prior fitting.

The pre-processing stage estimates the underlying geometric surface model for the original scene portion [29, 30] from which a set of displacement vectors, $\vec{D}(i)$, and a corrected surface normal, n_i, for each point i can be derived (see Fig. 5). Additionally we derive a completed 'smooth' portion of the invisible surface based on parametric shape completion [9, 11, 10] (e.g. Figure 1(B)).

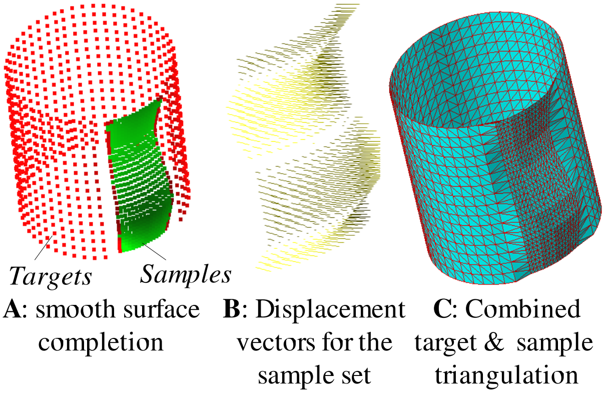

Targets
Samples
A: smooth surface completion
B: Displacement vectors for the sample set
C: Combined target & sample triangulation

Fig. 3. ‘Smooth’ surface completion and displacement vectors

The main input to our non-parametric completion process is a geometrically complete version of the 3D surface represented as a discrete set of labelled points, P. The originals, labelled as textured, are the sample points, $s \in \mathit{samples}$, whilst those forming the completed ‘smooth’ portion, labelled untextured, are the target points, $t \in \mathit{targets}$, as shown in Fig. 3(A). Each point also has an associated surface normal, n, and each sample point an associated displacement vector, $\overrightarrow{D}(s)$, as shown in Figs. 3(B) and Fig. 5. For convenience and to aid the construction and spatial use of point neighbourhoods on the surface this input is represented as a combined homoeomorphic surface triangulation [31, 32] of both target and sample points (see Fig. 3(C)). Henceforth we now consider our points, $i \in P$, as vertices, $i \in \mathit{triangulation}(P)$.

The reconstruction algorithm adapts to 3D by considering vertex neighbourhoods on the 3D surface in place of the pixel neighbourhoods of [26]. Each vertex neighbourhood, $N(i)$, is the set of vertices lying within a radius of W edge connections from the vertex being reconstructed (see Fig. 4). W forms the window size parameter synonymous to that of the earlier 2D approach. The algorithm now proceeds by finding the best sample region matching the textured portion of a target vertex's neighbourhood, as follows.

Firstly, the set of target vertices currently lying on the textured/untextured surface boundary are identified as the current target list, L. The first target vertex, $t \in L$, is then matched, using neighbourhood based matching, against every available vertex $s \in \mathit{samples}$. A match is then randomly chosen from the best 10% of this set, based upon matching score. Provided the matching score for this choice is below the specified acceptable error threshold parameter, e, this choice is accepted and the current target vertex, t, is textured by mapping the displacement vector, $\overrightarrow{D}(s)$, from the chosen sample vertex, s, to t. The current target, t, is now labelled as textured and the algorithm proceeds to the next vertex in L. If the match is not accepted (or no match was possible) the vertex is simply skipped and returned to the pool of target vertices for future synthesis—in this specific case the window size, W, associated with t for future

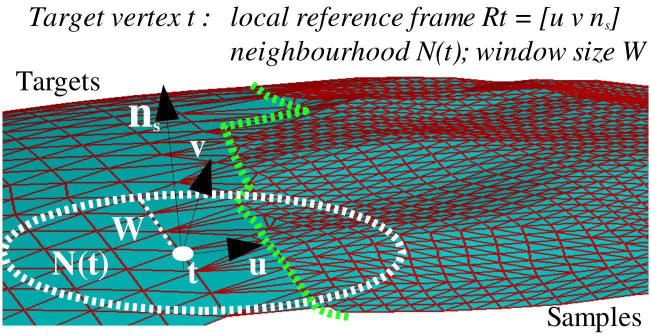

Target vertex t : local reference frame Rt = [u v n_s]
neighbourhood N(t); window size W
Targets
n_s
v
W
N(t)
t
u
Samples

Fig. 4. 3D vertex neighbourhoods

matching is reduced in size, $W_t = W - 1$, to facilitate matching on a scale of reduced constraint, global → local, where required.

Once L is exhausted, the next set of boundary targets are identified, based on the updated vertex labelling, and the process is continued until all $t \in targets$ are labelled as textured. To ensure target vertices are processed in the order of most to least constrained, L is sorted by decreasing number of textured neighbours prior to processing. Additionally, synthesis progress is monitored over each target list constructed—should no match choices be accepted over an entire list, the acceptable error threshold e is raised slightly (10%) to relax the acceptable error constraint for synthesis as per [26].

The remaining key element in this algorithm outline is the matching of textured target neighbourhoods (as shown in Fig. 4) to vertices in the sample region. This is performed using an adaptation of the SSD metric based on the projection of neighbourhood vertices onto the surface at each sample point. In order to compute the match between target vertex t, with textured neighbourhood vertices $Nt(t)$, and a sample vertex s with textured neighbourhood $Nt(s)$, $Nt(t)$ is first transformed rigidly into the co-ordinate system of s. This is based on knowing the local reference frames at s and t, denoted R_s and R_t respectfully, which combined with the positional translations given by t and s facilitate the transformation of $Nt(t)$ relative to s as $Nt(t)'$. However, as t is itself untextured whilst s is textured, the natural misalignment (owning to the presence or absence of texture) has to be avoided by transforming to the corresponding untextured position of s on the underlying surface, s', calculated using the displacement vector at s, $\overrightarrow{D}(s)$, as $s' = s - \overrightarrow{D}(s)$. Overall we have a resulting, $t \longrightarrow s'$, homogeneous co-ordinate transformation as follows:

$$Nt(t)' = \begin{bmatrix} [R_s] & s' \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} [R_t] & t \\ 0\,0\,0 & 1 \end{bmatrix}^{-1} Nt(t)$$

In order to estimate this spatial transformation the reference frames R_s and R_t are required. Given each vertex normal this can be generally derived using either localised curvature or more global fitting based techniques. Both, however,

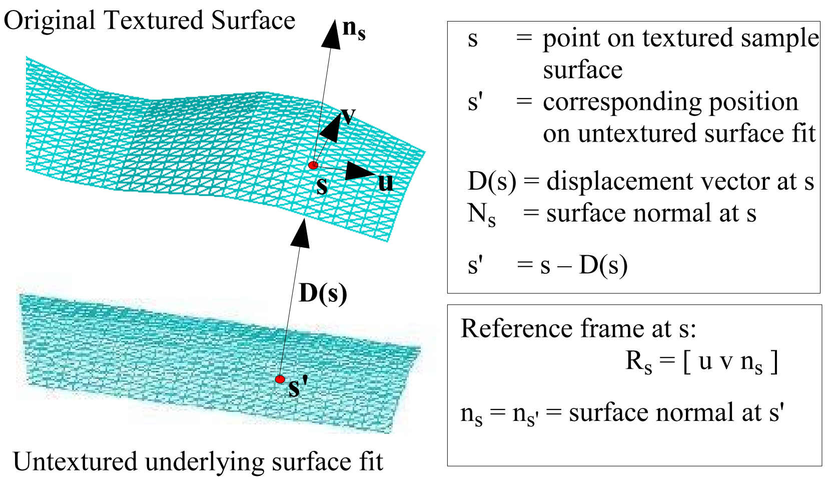

Original Textured Surface
n_s
v
s
u
D(s)
s'
Untextured underlying surface fit
s = point on textured sample surface
s' = corresponding position on untextured surface fit
D(s) = displacement vector at s
N_s = surface normal at s
s' = s – D(s)
Reference frame at s:
R_s = [u v n_s]
$n_s = n_{s'}$ = surface normal at s'

Fig. 5. Sample vertex geometry example

have disadvantages—notably their intolerance to noise and additionally the underlying ambiguity of surface orientation on many common geometric surfaces. Here, localised reference frames are derived deterministically based on finding mutually perpendicular vectors, $\overrightarrow{u}\ \overrightarrow{v}$, to the surface normal, $n = (x, y, z)$:

$$\begin{aligned}&\text{if } x = \min(|x|, |y|, |z|),\\ &\text{choose } u = (0, -z, y),\\ &v = n \times u,\end{aligned}$$

and by similar construct when y or z is the smallest.

Although far from perfect, this ensures at least localised consistency whilst the problems of global inconsistency are solved by simply augmenting the algorithm to match the target neighbourhood to every sample region at R different rotational orientations around the normal axis - additional parameter R specifies the divisions of 2π giving a set of rotations (e.g. $R = 4$ gives 4 orientations at 0, $\pi/2$, π, $3\pi/2$).

To aid understanding, an overview of the surface geometry described here is shown in Figs. 4 and 5.

The task now is to compute the SSD as a vertex matching problem between this transformed neighbourhood, $Nt(t)'$, and the textured surface vertices at s. Although this seems to be a simple 3D point matching problem the presence of sampled surface texture means that simple Euclidean space 'nearest point' matching using the raw textured vertices can produce artificial matches in common scenarios as shown in Fig. 6(A). Although such problems could be overcome by enforcing a scheme of one-to-one minimal distance cross-matching between the sets, this relies on the assumption that the densities of both point sets are

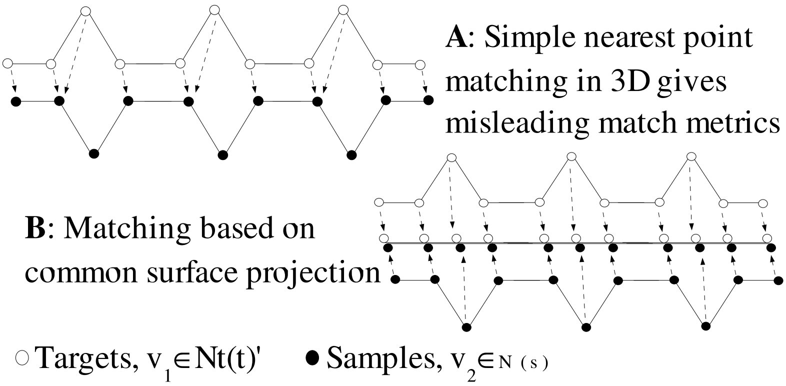

A: Simple nearest point matching in 3D gives misleading match metrics
B: Matching based on common surface projection
○ Targets, v1∈Nt(t)'
● Samples, v2∈N (s)

Fig. 6. Point matching via surface projection

equal—this is both difficult to assert uniformly and, as we shall discuss later, their inequality becomes a salient issue.

Here we ensure consistent vertex matching, independent of relative density, by matching vertices, $v_1 \rightarrow v_2$, $v_1 \in Nt(t)'$, $v_2 \in Nt(s)$, based on their relative projected positions on the common surface model, embodied in the displacement vector associated with every vertex, $v_i' = v_i - \overrightarrow{D}(v_i)$. This effectively matches vertices based solely on their relative spatial surface position rather than relative textured-related depth as shown in Fig. 6(B). From these pairings in surface projected space, $v_1' \rightarrow v_2'$, the SSD is calculated based on the original vertex positions, $v_1 \rightarrow v_2$.

It should also be noted that here we are *not* performing a neighbourhood, $Nt(t)'$, to closed neighbourhood, $Nt(s)$, match. Although our notation, $Nt(s)$, conceptually represents the surface vertices in the local region of s, $Nt(t)'$ actually is matched against the unrestricted set of textured vertices, $N(s) = (i \in P \,|\, \mathit{label}(i) = \mathit{textured})$, with a viable match only being considered when all matching partners, v_2, of $v_1 \in Nt(t)'$ are themselves also textured (i.e. v_2 has assigned label *textured*). When a viable match is found the SSD is calculated based on the distance of each target vertex, $v_1 \in Nt(t)'$, directly to the complete triangulated surface (not just the closest vertex)—i.e. the minimum squared distance to any surface triangle, $\triangle_j$, that has v_2 as a vertex, $\triangle_j \in \mathit{triangles}(v_2)$:

$$SSD_{shape} = \sum_{v_1}^{Nt(t)'} d_{v_1} \min_{\triangle_j \in triangles(v_2)} (dist(v_1, \triangle_j)^2)$$

Additionally, as in [26], a weight d_{v_i}, based on a 2D Gaussian kernel is used to weight the SSD vertex matches, $v_1 \rightarrow v_2$, relative to the distance $t \rightarrow v_1, v_1 \in N(t)$ (i.e. spatial proximity to t).

Pseudocode for the outlined technique is given in Appendix A.

4 Sampling Theory of 3D Surfaces

As identified in [25], one aspect highly relevant to this work is the adaptation of common sampling theory to 3D capture [33].

Although the concepts of under-sampling, aliasing and the Nyquist frequency for a given real world signal are common to general signal processing in lower dimensions [34] it would appear to have received little attention in 3D vision [33]. The specific sampling question that concerns us here is: given an existing surface capture, what is the required target vertex density to achieve synthesis without suffering aliasing effects? This is synonymous to obtaining the Nyquist frequency for the capture itself.

Based upon the Nyquist sampling theorem, that a signal must be sampled at twice the frequency of its highest frequency component, it can thus be derived that the upper limit on the Nyquist frequency, f_{Ny}, of a given signal capture is $1/\delta$ where δ represents the signal sampling density. This represents the minimum frequency at which the capture must be sampled in order to allow perfect reconstruction and is equal to twice the highest frequency component, $f_{\max}$, of the signal, $f_{Ny} = 1/\delta = 2f_{\max}$.

Transferring this principle back into the context of 3D triangulated surfaces, where the vertices are the sample points and the depth value is the signal, we have to consider that the sampling frequency across the whole surface may be non-uniform due to common-place variation in the original capture process. Hence only a lower limit on the sampling density required to successfully represent the maximum detail or highest frequency components can be considered based on the maximum surface sample density. This translates as the minimum distance between any two signal samples or conversely the minimum edge length, $\min(e)$, present in a Delaunay based triangulation (e.g. [31, 32]). This gives an upper limit on the Nyquist frequency, $f_{Ny} = 1/\min(e)$, and an upper spectral component limit, $f_{\max} = 1/2\min(e)$, for the surface capture.

Surface extension must thus use a vertex sampling density, δ, of at least $\min(e)$ to avoid the effects of aliasing and ensure restoration of the surface (i.e. $\delta \geq \min(e)$). This is illustrated in Fig. 7 where for a synthetic surface case we see that using a sampling density for the target vertices set below that associated with the Nyquist frequency causes aliasing (Fig. 7(A)), whilst using the minimum edge length removes the aliasing artifacts, (Fig. 7(B)).

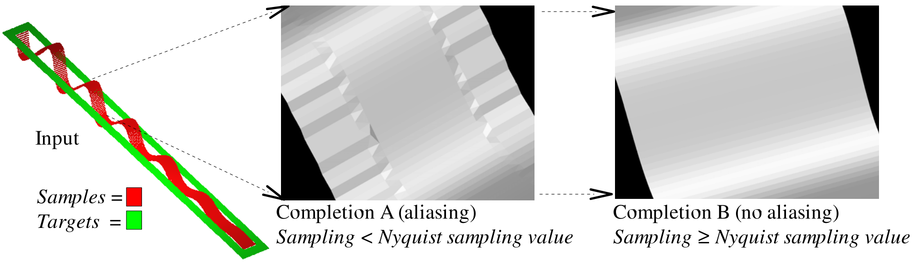

Input
Samples = 🟥
Targets = 🟩
Completion A (aliasing)
Sampling < Nyquist sampling value
Completion B (no aliasing)
Sampling ≥ Nyquist sampling value

Fig. 7. Aliasing in 3D completions

Our final issue in 3D sampling arises from remembering that here we are sampling and reconstructing from a finite digitised representation of a signal, a set of vertices representing surface sample points, rather than the infinite analogue signal commonly considered. Although the infinite surface is arguably represented by the surface lying through these points, embodied here in a triangulation, the nature of the non-parametric sampling technique requires finite to finite domain reconstruction, represented here by the sets of sample and target vertices. This introduces an issue relating to vertex alignment between the two regions. If there exists a significant phase shift between the target vertex set and the samples this results in a scenario where the suitable displacement value for a given target vertex, given its spatial position on the surface, is not adequately represented in the sample set—it in fact lies at some other point on the infinite surface. Due to the nature of this technique and limitations in the ability to identify and correct phase shifts in this domain we solve this problem by oversampling the original surface capture—creating the intermediate samples as required. It should now be clear that having an approach that is independent of a common point density for the sample and target portions is highly desirable. Practically, oversampling is achieved by subdividing the surface using an adaptation to surface tessellation such that each triangle is replaced by 4 coplanar triangles. For i original vertices, by reference to Euler's formula, this results in i' vertices where $i' \geq 2i$ but with no increase in the surface detail, and hence no increase in the Nyquist related surface properties.

Overall, from our 3D sampling discussion, we now have a practical means of determining a suitable surface reconstruction, the minimum triangulation edge length, and an oversampling solution for phase alignment problems.

5 Surface Relief Results

Here we present a number of examples of isolated localised surface structure (relief) completion using our approach. Firstly, in Fig. 8 we see the successful completion of synthetic wave and noise patterns over planar surfaces and the completion of localised surface shape on cylindrical surfaces. Surface completions based on using real objects portions, scanned with our 3D Scanners' Reversa laser scanner, are presented in Figures 1, 9, and 10. These show the successful completion of a range of surface types from the propagation of golfball dimples across the completed sphere (Fig. 1), natural tree bark texture realistically completed over an extended cylinder (Fig. 9) and structured surface completion of a scale model of the Pisa tower (Fig. 10).

These results were produced using Euclidean [30] or least squares [29] fitting for initial geometric completion. The original sample surface was oversampled once, triangulation utilised the Cocone algorithm [31, 32] and Mersenne twister [35] provided the random source. All completions use only the original, oversampled set of textured points as sample vertices—the variation, called 'boot-strapped' completion, whereby the usable sample set grows as the textured surface area grows, is not considered here.

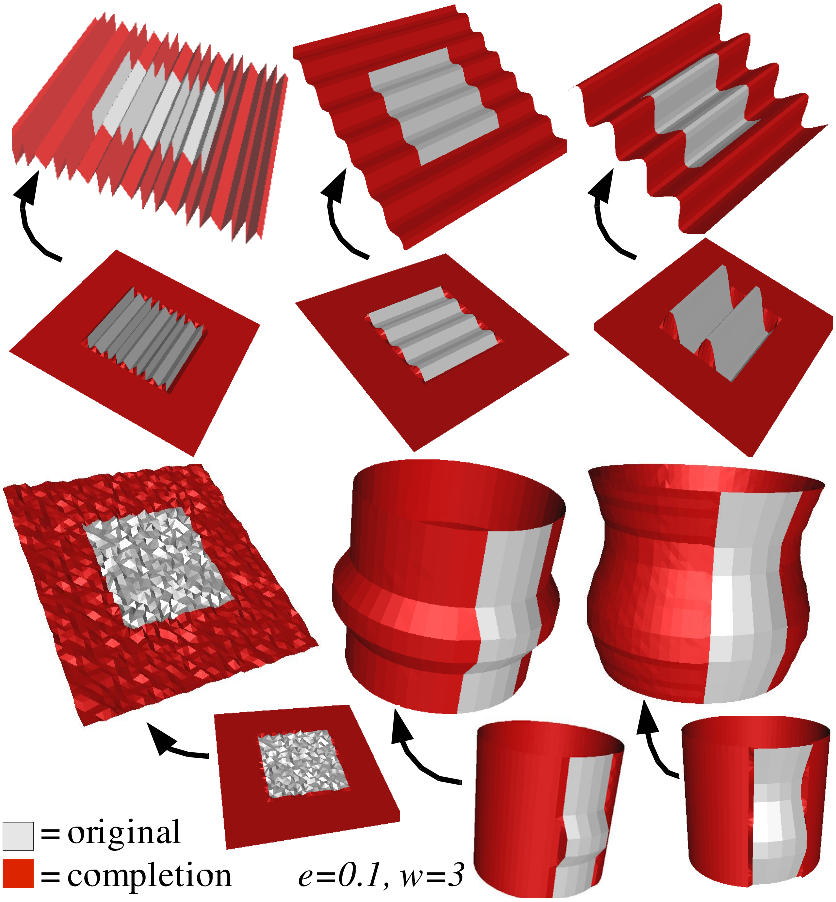


Fig. 8. Completion of synthetic examples

As a means of quantitative evaluation, the mean integral of the volume between the geometric surface fit and the original and synthetic (completed) surface portions for a sample of results are shown in Table 1. These statistics support the visual similarity of the results (i.e. Figs. 8 and 10) but also show a statistical increase in difference where the texture is stochastic in nature (i.e. Figs. 9 and 12). In both cases the statistics identify a difference not apparent to visual inspection (see Figs. 9 and 12) and hence arguably within the bounds of visually plausible completion—our desired goal.

Overall the results produce realistically structured and textured surface completions representing *plausible completion*. However, two limitations experienced in the original 2D work [26] where similarly encountered. Firstly, erroneous completions were encountered in some cases due to the effects of accumulated error and illustrate the reliance on good parameter choice (see [25]). Additionally, despite extensive pre-computation and memoisation, this technique is computa-

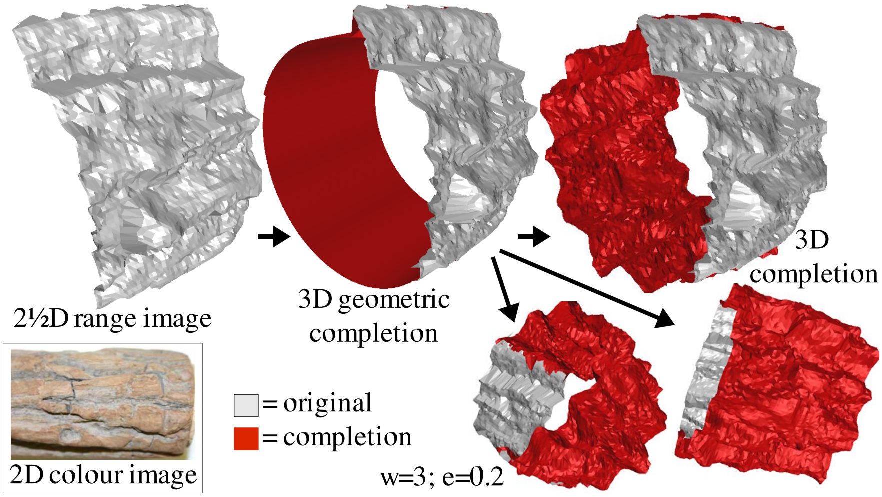


Fig. 9. Completion of natural textures—tree bark

Table 1. Mean integral below surface texture

Object	Original	Completion	% diff.
Fig. 8 bottom right	0.247123	0.252846	2.32%
Fig. 10 bottom right	0.807048	0.828891	2.71%
Fig. 9	1.18208	1.24769	5.55%
Fig. 12	0.15616	0.166552	6.65%

tionally very expensive ($O(stw)$ for s samples and t targets and window size w. Fig. 9 requires about 13 hours on a 2.6Ghz Pentium 4 with $t = 7200, s = 12852$).

6 Extension to Colour

In addition to completing the localised surface relief it is also possible to extend our approach to perform combined surface relief and colour completion on increasingly available colour $2\frac{1}{2}$D or 3D range data.

The SSD equation utilised previously, SSD_{shape}, can be adjusted as follows to take account of both surface shape and colour, SSD_{colour}:

$$SSD = (\mu)SSD_{shape} + (1 - \mu)SSD_{colour}$$

where μ defines the relative weighting of shape (i.e. relief) and colour in the overall matching value. This weight balances the relative importance of localised surface shape against colour in the completion problem. Commonly, as in many natural objects, both are closely interrelated and equal weighting may be suit-

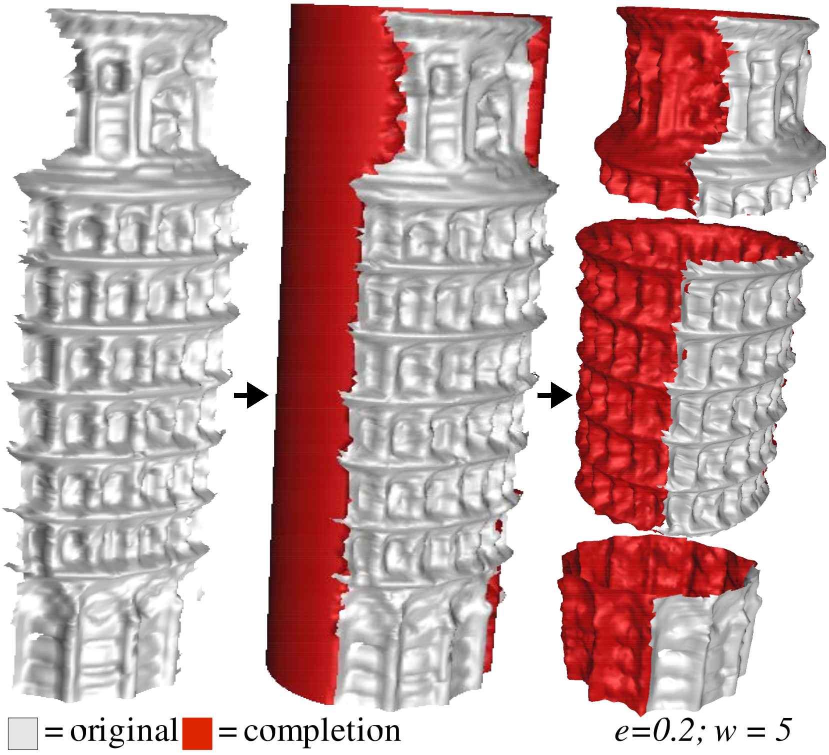


Fig. 10. Completion of tower of Pisa

able. More generally the correlation of colour to surface relief (or relative 3D position) is left as an aspect of psychological vision research [36].

The SSD_{colour} is calculated based on comparing the RGB colour, $c \in \{r, g, b\}$, of each vertex $v_1 \in Nt(t)'$ to that of the nearest sample vertex, v_2, on the surface at s:

$$SSD_{colour} = \sum_{v_1}^{Nt(t)'} d_{v_1} \sum_{c}^{\{r,g,b\}} \frac{\mid c_{v_1} - c_{v_2} \mid}{3}.$$

As before, d_{v_1}represents the Gaussian weight associated with the target neighbourhood vertex, v_1, based on its spatial proximity to t in $Nt(t)$.

Both SSD_{shape} and $SSSD_{colour}$ are normalised to the range $[0, 1]$ utilising a known upper bound on each. For colour this is based on the size of the utilised colour map and for shape based on maximum difference between two displacement vectors in the set of samples taking into consideration displacement occurring on either side of the surface fit. This maximum difference is computed by:

$$\max_{s_i, s_j \in samples} \parallel (n_{s_i} . \frac{\overrightarrow{D}(s_i)}{\parallel \overrightarrow{D}(s_i) \parallel}) \overrightarrow{D}(s_i) - (n_{s_j} . \frac{\overrightarrow{D}(s_j)}{\parallel \overrightarrow{D}(s_j) \parallel}) \overrightarrow{D}(s_j) \parallel$$

where n_s is the surface normal at sample s.

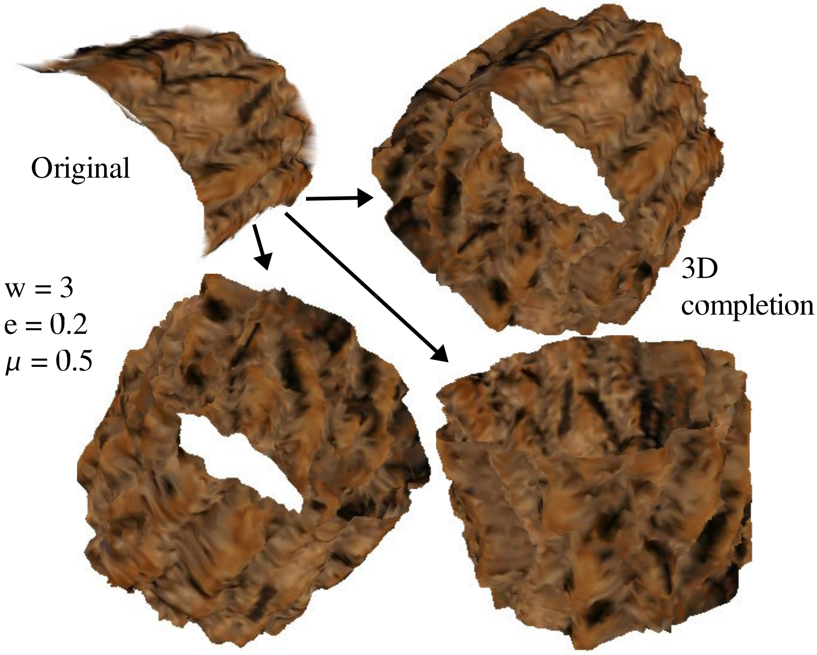


Fig. 11. Completion of tree bark surface relief and colour

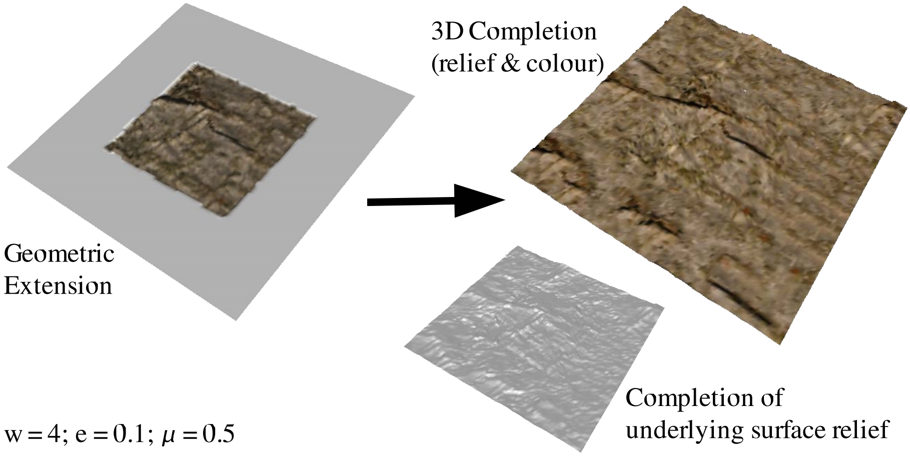


Fig. 12. Extension of tree bark relief and colour

Using this alternative SSD equation the algorithm operates as previously outlined with the additional step of colour propagation from sample to target when a suitable match is chosen (see Appendix A). Oversampling of the coloured

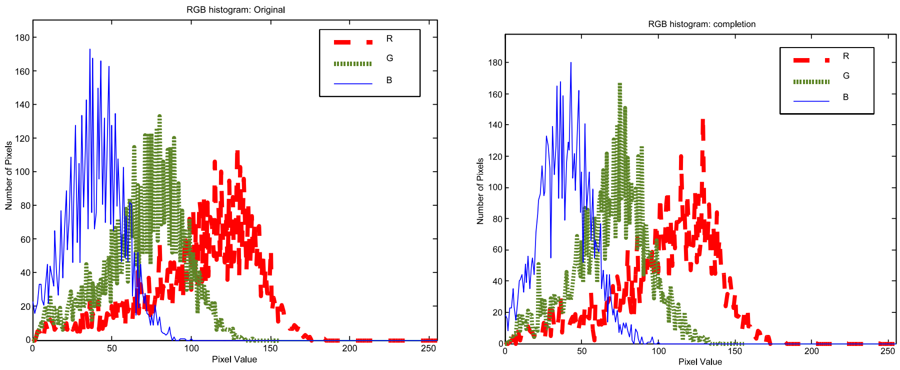


Fig. 13. RGB histogram of original (left) and completed (right) tree bark portion in Fig. 11

sample surface is performed by colouring each new sample, due to tessellation, using a Gaussian weighted sum of the k nearest neighbours.

The combined colour and surface shape completion of a section of tree bark, captured using a high resolution stereo capture rig, is shown in Fig. 11[2]. This shows a highly *plausible* completion of the bark surface texture and associated colour over the underlying geometric cylinder completion of the original bark sample (as per Fig. 9). In addition to this visual comparison, the colour histograms for the original and completed portions, separated by RGB colour channel (Fig. 13), show a strong correlation between the colour distribution achieved in the completed portion and that of the original. A further example of successful colour and surface shape completion is shown in Fig. 12 where we see the clear extension of existing features from the original (i.e. continuation major ridges present in bark sample) and the derivation of similar distinct features (i.e. new ridges formed in foreground of completion) in addition to the general plausibility of the completed relief and colour correlation.

7 Conclusions and Further Work

In this paper we have presented a novel method for 3D surface completion that, given the underlying surface geometry, facilitates the *plausible completion* and *good continuation* of surfaces without strict surface localised surface geometry both in terms of localised surface structure (relief) and colour.

This works extends, and indeed complements, earlier work on the smooth surface completion and hole-filling [4, 3, 5, 6, 7, 8] and on strict geometric completion [9, 10, 11, 12, 13] with both its completion abilities for localised surface structure and also the integration of concurrent structure and colour completion.

[2] Colour versions of Figures 11 and 12 are also available from the Web site http://www.iplab.inf.ed.ac.uk/mvu/breckon/

Our successful completion / extension of *real* surface relief, combined with colour, presents interesting future directions for work in creating illusory surface relief [18, 19] and enhancing 3D model rendering through realistic and consistent texturing with a 2D sample [15, 16, 17]. Similarly it offers an interesting parallel to work in the transfer of surface reliefs [20, 21, 22, 23, 24] both in terms of the technique proposed and its practical application.

In contrast to the specific completion work of [14], our technique does not suffer the limitations of such a patch based approach, at the expense of computational cost, but does rely on knowledge of the underlying 'smooth' surface completion—here derived from geometric fitting but possibly obtainable from prior techniques in smooth surface completion [3, 6, 5, 8, 7] and fitting [37] in subsequent work.

In terms of future work a number of possibilities remain—notably the extension to surfaces without strict underlying geometry and also work to address the issues of computational complexity and accumulated error identified previously [25]. It is hoped that the investigation of a multi-resolution variant on the proposed technique, together with the adaptation of other 2D texture synthesis techniques to this problem domain, will allow future progress in these areas.

In addition, interesting issues relating to the approximation of the Nyquist frequency of a 3D surface and in synthesising surfaces through infinite representation models still require investigation—an area of equal interest in 3D storage, transmission and compression as it is in synthesis.

Acknowledgements. This work was supported by EPSRC and QinetiQ PLC. Textures in Fig. 2 reproduced by kind permission of A. Efros [26].

References

1. Besl, P.J., McKay, N.D.: A method for registration of 3D shapes. IEEE Transactions Pattern Analysis and Machine Intelligence **14** (1992) 239–256
2. Rodrigues, M., Fisher, R., Liu, Y.: Special issue on registration and fusion of range images. Computer Vision and Image Understanding **87** (2002) 1–7
3. Davis, J., Marschner, S., Garr, M., Levoy, M.: Filling holes in complex surfaces using volumetric diffusion. In: Proc. First Int. Sym. on 3D Data Processing, Visualization and Transmission. (2002) 428– 861
4. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. In: Proc. 28th SIGGRAPH, ACM Press (2001) 67–76
5. Wang, J., Oliveira, M.: A hole-filling strategy for reconstruction in smooth surfaces in range images. In: 16th Brazilian Symp. on Computer Graphics and Image Processing, IEEE Computer Society (2003) 11–18
6. Liepa, P.: Filling holes in meshes. In: Proc. of the Eurographics/ACM SIGGRAPH symp. on Geometry processing, Eurographics Association (2003) 200–205
7. Ju, T.: Robust repair of polygonal models. ACM Transactions on Graphics **23** (2004) 888–895
8. Tekumalla, L., Cohen, E.: A hole-filling algorithm for triangular meshes. Technical Report UUCS-04-019, School of Computing, University of Utah (2004)

9. Stulp, F., Dell'Acqua, F., Fisher, R.: Reconstruction of surfaces behind occlusions in range images. In: Proc. 3rd Int. Conf. on 3D Digital Imaging and Modelling. (2001) 232–239
10. Dell'Acqua, F., Fisher, R.B.: Reconstruction of planar surfaces behind occlusions in range images. IEEE Transactions Pattern Analysis and Machine Intelligence **24** (2002) 569–575
11. Castellani, U., Livatino, S., Fisher, R.: Improving environment modelling by edge occlusion surface completion. In: Proc. First Int. Sym. on 3D Data Processing, Visualization and Transmission. (2002) 672–675
12. Fisher, R.: Solving architectural modelling problems using knowledge. In: Proc. 4th Int. Conf. on 3D Digital Imaging and Modelling. (2003) 343–351
13. Chalmoviansky, P., Juttler, B.: Filling holes in point clouds. In Wilson, M., Martin, R., eds.: Mathematics of Surfaces X. Number 2768 in LNCS, Spring-verlag (2003) 196–212
14. Sharf, A., Alexa, M., Cohen-Or, D.: Context-based surface completion. ACM Transactions on Graphics **23** (2004) 878–887
15. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: Proc. ACM SIGGRAPH. (2000) 465–470
16. Turk, G.: Texture synthesis on surfaces. In: Proc. ACM SIGGRAPH. (2001) 347–354
17. Wei, L.Y., Levoy, M.: Texture synthesis over arbitrary manifold surfaces. In: SIGGRAPH: Proc. 28th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press (2001) 355–360
18. Gorla, G., Interrante, V., Sapiro, G.: Texture synthesis for 3d shape representation. IEEE Transactions Visualization and Computer Graphics **9** (2003) 512–524
19. Zelinka, S., Garland, M.: Interactive texture synthesis on surfaces using jump maps. In: Proc. 14th Eurographics workshop on Rendering, Eurographics Association (2003) 90–96
20. Biermann, H., Martin, I., Bernardini, F., Zorin, D.: Cut-and-paste editing of multiresolution surfaces. ACM Transactions on Graphics **21** (2002) 312–321
21. Bhat, P., Ingram, S., Turk, G.: Geometric texture synthesis by example. In: Proc. Eurographics/ACM SIGGRAPH symp. on Geometry processing, New York, NY, USA, ACM Press (2004) 41–44
22. Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proc. Eurographics/ACM SIGGRAPH symp. on Geometry processing, Eurographics Association (2004) 179–188
23. Zelinka, S., Garland, M.: Similarity-based surface modelling using geodesic fans. In: Proc. Eurographics/ACM SIGGRAPH symp. on Geometry processing, New York, NY, USA, ACM Press (2004) 204–213
24. Lai, Y.K., Hu, S.M., Gu, D., Martin, R.: Geometric texture synthesis and transfer via geometry images. ACM Solid and Physical Modelling **(in publication)** (2005)
25. Breckon, T., Fisher, R.: Non-parametric 3D surface completion. In: Proc. 5th Int. Conf. on 3D Digital Imaging and Modelling, IEEE Press (2005) (to appear)
26. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: IEEE Int. Conf. on Computer Vision. (1999) 1033–1038
27. Zhu, S., Wu, Y., Mumford, D.: Filters, random-fields and maximum-entropy (frame): Towards a unified theory for texture modelling. Int. Journal of Computer Vision **27** (1998) 107–126
28. Kokaram, A.: Parametric texture synthesis for filling holes in pictures. In: Proc. Int. Conf. on Image Processing. (2002) I: 325–328

29. Forbes, A.: Least-squares best-fit geometric elements. Technical Report 140/89, National Physical Laboratory, Teddington, UK (1989)
30. Faber, P., Fisher, R.: Euclidean fitting revisited. In: 4th International Workshop on Visual Form. Volume 2059 of LNCS., Springer-verlag (2001) 165
31. Dey, T.K., Giesen, J.: Detecting undersampling in surface reconstruction. In: Proc. 17th ann. symp. on Computational Geometry, ACM Press (2001) 257–263
32. Dey, T.K., Goswami, S.: Tight cocone: a water-tight surface reconstructor. In: Proc. 8th ACM sym. on Solid modelling and applications, ACM Press (2003) 127–134
33. Taubin, G.: Geometric signal processing on polygonal meshes. In: Proc. EUROGRAPHICS: State of the Art Report (STAR), Interlaken, Switzerland (2000)
34. Shannon, C.: Communication in the presence of noise. Proc. Inst. of Radio Engineers **37** (1949) 10–21
35. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Modelling and Computer Simulation **8** (1998) 3–30
36. Wandell, B.: Foundations of Vision. Sinauer Associates (1995)
37. Krishnamurthy, V., Levoy, M.: Fitting smooth surfaces to dense polygon meshes. In: Proc. SIGGRAPH, ACM Press (1996) 313–324

A Algorithm Pseudocode

This technique is now outlined in pseudocode, following in the style of the original 2D work [26], based on the definition of the following key data items:

- surface = triangulated surface with vertices labelled as textured/untextured.
- targets = list of untextured vertices (targets) of A.
- samples = list of textured vertices (samples) of A.
- $\overrightarrow{D}(s_i)$= surface displacement vector of textured vertex S$_i$, $S_i \in$ *samples.*

GrowSurface(surface, targets, samples)

```
while targets is not empty
  progress = 0
  theTargets = vertices on textured/untextured boundary of surface.
  for each vertex v in theTargets do
    BestMatches = FindMatches(v, surface, samples)
    if BestMatches not empty
       BestMatch = random selection from (BestMatches)
       if (BestMatch.error < MaxErrThreshold) then
          Propagate D(BestMatch) to v
          progress = 1
          Remove v from targets
          Label v as textured
       end if
    end if
  end for
  if progress == 0
```

```
      MaxErrThreshold = MaxErrThreshold * 1.1;
    end if
  end while
```

FindMatches(v, surface, samples)

```
  neighbours = textured vertices in vertex neighbourhood of v
  for every vertex v’ of neighbours
    calculate Gaussian weight w(v’) relative to distance(v’, v)
  end for
  TotalWeight = sum(i in neighbours, w(i))
  for every vertex s in samples do
    neighbours’ = neighbours transformed relative to s
    ssd = 0
    validMatch = TRUE
    for each point i in neighbours’ do
      closestVertex = closest vertex on surface to i
      if (closestVertex is in samples)
        tri = closest triangle of surface to i
        (v1, v2, v3) = vertices of triangle tri
        if (v1, v2, v3 are in samples)
          distance = min distance from i to tri
          ssd += w(i) * distance
        else
          validMatch = FALSE
        end if
      else
        validMatch = FALSE
      end if
    end for
    if validMatch do
      ssd = ssd / TotalWeight
      Add (s, ssd) to Matches
    end if
  end for
  BestMatches = all i in Matches with i.ssd <= min(i.ssd) * (1.1)
  Return BestMatches
```

For the extension to combined relief and colour, SSD and propagation are redefined as stated earlier.

Determining the Topology of Real Algebraic Surfaces

Jin-San Cheng[1], Xiao-Shan Gao[1], and Ming Li[2]

[1] Key Lab of Mathematics Mechanization,
Institute of Systems Science, AMSS, Academia Sinica, Beijing 100080, China
jcheng@amss.ac.cn, xgao@mmrc.iss.ac.cn
[2] School of Computer Science, Cardiff University, Cardiff, CF24 3AA, UK
m.li@cs.cardiff.ac.uk

Abstract. An algorithm is proposed to determine the topology of an implicit real algebraic surface in $\mathbb{R}^3$. The algorithm consists of three steps: surface projection, projection curve topology determination and surface patches composition. The algorithm provides a curvilinear wireframe of the surface and the surface patches of the surface determined by the curvilinear wireframe, which have the same topology as the surface. Most of the surface patches are curvilinear polygons. Some examples are used to show that our algorithm is effective.

1 Introduction

An implicit real algebraic surface (or curve, or hypersurface) $\mathcal{S}$ in $\mathbb{R}^u$ with degree d is defined by $f(x_1, x_2, \cdots, x_u) = 0$ where $f(x_1, x_2, \cdots, x_u) \in \mathbb{Q}[x_1, x_2, \cdots, x_u]$ is a polynomial of degree d, and $\mathbb{R}$ and $\mathbb{Q}$ are the fields of real and rational numbers, respectively. Determining the topology of an algebraic surface is not only an interesting mathematical problem, but also a key issue in computer graphics and CAGD [4, 5, 20, 22].

When $u = 1$, $\mathcal{S}$ is a set of discrete points on a line. When $u = 2$, $\mathcal{S}$ is a plane algebraic curve. Topology determination for plane algebraic curves has been studied thoroughly [1, 3, 6, 7, 9, 12, 13, 14, 16, 21]. Algorithms to determine the topology of spatial algebraic curves are also proposed in the following papers [4, 6, 9, 10]. When $u = 3$, the problem is more complex. The topology of $\mathcal{S}$ with $d = 2$ is well known. They are quadratic surfaces. But when $d \geq 3$, there are only some special surfaces whose topology can be efficiently determined [11, 12]. Fortuna et al presented an algorithm to determine the topology of non-singular, orientable real algebraic surfaces in the projective space [8]. Morse theory is used to represent an implicit algebraic surface by polyhedra in theory by Hart et al [15, 20, 22]. Theoretically, the CAD (Cylindrical Algebraic Decomposition) method proposed by Collins can be used to provide information about the topology of an algebraic surface [2, 3]. But in the general case, there exist no complete algorithms to determine the topology of an implicit algebraic surface.

In this paper, we present an algorithm to determine the topology of $\mathcal{S}$ for $u = 3, d \geq 3$. In the rest of this paper, we replace $f(x_1, x_2, x_3) = 0$ with $f(x, y, z) = 0$.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 121–146, 2005.

We obtain a curvilinear wireframe of the surface. The surface patches of the surface are determined by the curvilinear wireframe. Most of the surface patches are curvilinear polygons. The wireframe and the surface patches have the same topology as the surface. If needed, we can easily modify our algorithm to ensure that all the surface patches are curvilinear triangles.

The basic idea of our algorithm is as follows. We first ensure that the surface is a normal surface by performing certain transformations. We then project $\mathcal{S}$: $f(x, y, z) = 0$ to a proper plane and obtain a plane algebraic curve $\mathcal{C}$: $g(x, y) = 0$. Thirdly, we analyze the topology of $\mathcal{C}$ in a finite box, by finding its singularities, dividing the curve into plane curve segments, and dividing the box in the plane into cells. At the fourth step, we divide the spatial curve defined by $\{f(x, y, z) = 0, g(x, y) = 0\}$ into spatial curve segments and compute the number of surface patches connected with each spatial curve segment. This is the key step of the algorithm. In order to determine the number of curve segments connected with a singular point and the number of surface patches connected with a curve segment, we introduced certain minimal circles and find these numbers from the information of the intersections of the circle with the surface. The main steps of the algorithm are similar to Collins' CAD method. But, the purpose of our algorithm is different from that of the CAD method, and many aspects of the algorithm are totally new. Main parts of the algorithm are implemented in Maple and nontrivial examples are used to show that the algorithm is effective.

This paper is divided into six sections. The aim of the second section is to obtain projection curve of the surface. The third section presents an algorithm to determine the topology of the plane projection curve. Space curve segmentation, surface patch composition and the surface topology representation are discussed in the fourth section. The fifth section presents the main algorithm to obtain the topology of a given algebraic surface. Then we draw a conclusion in the last section.

2 Projection Curve of a Surface

In the following, we always assume $\mathcal{S}$ is an algebraic surface: $f(x, y, z) = 0$, where $f(x, y, z) \in \mathbb{Q}[x, y, z]$. Suppose that

$$f(x, y, z) = f_1(x, y, z)^{m_1} \cdots f_n(x, y, z)^{m_n}, \tag{1}$$

where $f_i(x, y, z) \in \mathbb{Q}[x, y, z] (i = 1, \cdots, n)$ are irreducible polynomials. If a component contains variable z only, it represents some parallel planes. We can delete this kind of components before we compute the projection curve and add these planes into the topology structure and compute the intersection curve with other components after we finish the analysis. So we suppose that there does not exist this kind of components. It is clear that $f(x, y, z) = 0$ and $f_1(x, y, z) \cdots f_n(x, y, z) = 0$ have the same topology. We still denote

$$f(x, y, z) = f_1(x, y, z) \cdots f_n(x, y, z). \tag{2}$$

Let

$$g(x,y) = Res(f(x,y,z), \frac{\partial f(x,y,z)}{\partial z}, z), \tag{3}$$

where $Res(f(x,y,z), \frac{\partial f(x,y,z)}{\partial z}, z)$ is the Sylvester resultant [26] of $f(x,y,z)$ and $\frac{\partial f(x,y,z)}{\partial z}$ with respect to z. Suppose $g(x,y) = g_1(x,y)^{n_1} \cdots g_m(x,y)^{n_m}$, where $g_i(x,y)(i = 1, \cdots, m)$ are irreducible polynomials. Still denote

$$g(x,y) = g_1(x,y) \cdots g_m(x,y). \tag{4}$$

Then the *projection curve* of the surface $\mathcal{S} : f(x,y,z) = 0$ is a plane curve defined by $g(x,y) = 0$. In this section, we will prove some properties of the projection curve of a given surface $\mathcal{S}$.

In order to determine the topology of $\mathcal{S}$ effectively and efficiently, we assume that

C1. There exist no points $P_0(x_0, y_0)$ satisfying $f(x_0, y_0, z) \equiv 0$.

C2. $\sum_{i+j+k=d} a_{i,j,k} \cdot x^i \cdot y^j \cdot z^k$ has no factors like $T(x,y)$, where d is the total degree of $f(x,y,z)$, $a_{i,j,k}$ is the coefficient of the term $x^i \cdot y^j \cdot z^k$ in $f(x,y,z)$. $T(x,y)$ is a bivariate polynomial.

A *normal surface* is an algebraic surface defined by a square-free polynomial (the multiple of the irreducible factors of the polynomial is no more than 1) satisfying conditions *C1* and *C2*.

If condition *C1* does not hold, represent $f(x,y,z)$ as follows.

$$f(x,y,z) = c_k(x,y) \cdot z^k + c_{k-1}(x,y) \cdot z^{k-1} + \cdots + c_0(x,y), \tag{5}$$

where $c_i(x,y) \in \mathbb{Q}[x,y](i = 1, \cdots, k)$ and $c_k(x,y)$ is a nonzero polynomial. Then, the variety $\{c_0(x,y) = 0, c_1(x,y) = 0, \cdots, c_k(x,y) = 0\}$ has real roots, and the line $\{x = x_0, y = y_0\}$ is on the surface $\mathcal{S}$. In this case, it is difficult to analyze the topology of the surface near this line. Here is an example.

$$f(x,y,z) = x^2 \cdot y^2 + z^2 \cdot y^2 + x^2 \cdot z^2 - 7/2 \cdot x \cdot y \cdot z. \tag{6}$$

We have $f(0,0,z) \equiv 0$ and $\{x = 0, y = 0\}$ is a line on the surface.

Represent $f(x,y,z)$ as follows.

$$f(x,y,z) = L_d(x,y,z) + L_{d-1}(x,y,z) + \cdots + L_0, \tag{7}$$

where $L_t(x,y,z) = \sum_{i+j+k=t} a_{i,j,k} \cdot x^i \cdot y^j \cdot z^k (t = 0, \cdots, d)$. It is clear that all the asymptotic surfaces are contained in the surface defined by the equation $L_d(x,y,z) = 0$. If condition *C2* does not hold, there exists an asymptotic surface of $f(x,y,z) = 0$ of the form $T(x,y) = 0$, which is vertical to XY-plane. For example, the surface $f(x,y,z) = x \cdot y \cdot z - 1 = 0$ does not satisfy condition *C2*, because $x = 0, y = 0$ are asymptotic planes of it.

Lemma 1. *If a surface is not normal, we can find a coordinate transformation like (8) such that the surface obtained with this transformation has the same topology as the original one and is normal.*

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{8}$$

where (X, Y, Z) and (x, y, z) are points in the new and old coordinate systems, respectively, and a, b are rational numbers.

Proof. Taking the coordinate transformation as (8) and representing $f(x, y, z)$ as (7), we have

$$\begin{aligned} &F(X, Y, Z) \\ &= f(X + a \cdot Z, Y + b \cdot Z, Z) \\ &= L_d(X + a \cdot Z, Y + b \cdot Z, Z) + L_{d-1}(X + a \cdot Z, Y + b \cdot Z, Z) + \cdots + L_0 \\ &= L_d(a, b, 1) \cdot Z^d + C_{d-1}(X, Y)Z^{d-1} + \cdots + C_0, \end{aligned}$$

where $C_i(X, Y), i = 0, 1, \cdots, d-1$ is the coefficients of $F(X, Y, Z)$ in variable Z. We can find rational numbers a_0, b_0, such that, $L_d(a_0, b_0, 1) \neq 0$. Denote the corresponding $F(X, Y, Z)$ as $F_0(X, Y, Z)$. We will show that $F_0 = 0$ is a normal surface. Since $L(a_0, b_0, 1)$ is a nonzero constant, $F_0 = 0$ satisfies condition *C1*. It is clear that all the asymptotic surface of $F_0 = 0$ are hidden in $L_d(X + a_0 \cdot Z, Y + b_0 \cdot Z, Z) = 0$. There is a term $L_d(a, b, 1) \cdot Z^d$ in $L_d(X + a_0 \cdot Z, Y + b_0 \cdot Z, Z)$, so there are no factors like $T(X, Y)$ hidden in it. $F_0 = 0$ satisfies condition *C2*. So $F_0 = 0$ is a normal surface. □

For the non-normal surface $f(x, y, z) = x \cdot y \cdot z - 1 = 0$, we choose $a = 1, b = 1$. The new surface is $F(X, Y, Z) = Z^3 + (Y + X) \cdot Z^2 + X \cdot Y \cdot Z - 1 = 0$. It is a normal surface.

For the surface defined by (6), we can choose $(a, b) = (1, 1)$. The new surface is $F(X, Y, Z) = 3 \cdot Z^4 + 4 \cdot Y \cdot Z^3 + 4 \cdot X \cdot Z^3 - 7/2 \cdot Z^3 + 2 \cdot Y^2 \cdot Z^2 + 2 \cdot X^2 \cdot Z^2 - 7/2 \cdot Y \cdot Z^2 - 7/2 \cdot X \cdot Z^2 + 4 \cdot X \cdot Y \cdot Z^2 - 7/2 \cdot X \cdot Y \cdot Z + 2 \cdot X \cdot Y^2 \cdot Z + 2 \cdot X^2 \cdot Y \cdot Z + X^2 \cdot Y^2 = 0$. It is a normal surface.

Following the discussion above, we can derive the following algorithm to obtain a *normal projection curve* (the projection curve of a normal surface) for a given irreducible surface $f(x, y, z) = 0$.

Algorithm 1. *The input is an irreducible polynomial $f(x, y, z)$. The output is a normal projection curve $g(x, y) = 0$ of the surface $f(x, y, z) = 0$.*

1. Represent $f(x, y, z)$ as (5) and check whether the variety $\{c_k(x, y), c_{k-1}(x, y), \cdots, c_0(x, y)\}$ has a real solution. If it has, go to 3.
2. Represent $f(x, y, z)$ as (7) and check whether $L_d(x, y, z)$ has a factor which does not involve variable z. If it does not have this kind of factors, go to 4.
3. Apply the transformation (8), choose a rational number pair (a, b) such that (a, b) is not a point on curve $L_d(x, y, 1) = 0$, where $L_d(x, y, z)$ is the sum of terms whose degrees equal the total degree of $f(x, y, z)$, and compute the corresponding new surface $F(X, Y, Z) = 0$ in the new coordinate system. Still denote $F(X, Y, Z)$ as $f(x, y, z)$.
4. Compute $g(x, y) = Res(f(x, y, z), \frac{\partial f(x,y,z)}{\partial z}, z)$.
5. If $g(x, y)$ is irreducible, return $g(x, y) = 0$. Else, factor it as $g(x, y) = g_1(x, y)^{m_1} \cdot g_2(x, y)^{m_2} \cdot \cdots \cdot g_t(x, y)^{m_t}$, where $g_i(x, y)$ is irreducible. Still denote $g(x, y) = g_1(x, y) \cdot g_2(x, y) \cdot \cdots \cdot g_t(x, y)$ and return $g(x, y) = 0$.

When $f(x,y,z)$ is reducible as (2), the problem is more complex. We can use Algorithm 1 to compute its projection curve, but the computation takes much time. We can check whether each component $f_i(x,y,z)$ is a normal surface. If all components are normal surfaces, we can compute the projection curve of $f(x,y,z)=0$ as follows.

Lemma 2. *Let $\mathcal{S} : f(x,y,z)=0$, where $f(x,y,z)$ is defined by (2), $n \geq 2$. The projection curve of $\mathcal{S}$ is the curve defined by the square-free part of the following polynomial.*

$$g(x,y) = \prod_{1\leq i\leq j\leq n} T_{i,j}(x,y), \tag{9}$$

where $T_{i,i}(x,y) = Res(f_i(x,y,z), \frac{\partial f_i(x,y,z)}{\partial z}, z)$, $T_{i,j}(x,y) = Res(f_i(x,y,z), f_j(x,y,z), z)$, $i,j = 1,\cdots,n, i \neq j$.

Proof. By (3) and the property of resultant [25], we can derive that

$$\begin{aligned}
&Res(f(x,y,z), \frac{\partial f(x,y,z)}{\partial z}, z)) \\
&= Res(\prod_{1\leq j\leq n} f_j(x,y,z), \sum_{i=1}^{n} \frac{f(x,y,z)}{f_i(x,y,z)} \cdot \frac{\partial f_i(x,y,z)}{\partial z}, z) \\
&= \prod_{1\leq j\leq n} Res(f_j(x,y,z), \sum_{i=1}^{n} \frac{f(x,y,z)}{f_i(x,y,z)} \cdot \frac{\partial f_i(x,y,z)}{\partial z}, z) \\
&= c \cdot \prod_{1\leq j\leq n} Res(f_j(x,y,z), \frac{f(x,y,z)}{f_j(x,y,z)} \cdot \frac{\partial f_j(x,y,z)}{\partial z}, z) \\
&= c \cdot \prod_{1\leq j\leq n} (Res(f_j(x,y,z), \frac{\partial f_j(x,y,z)}{\partial z}, z) \cdot \prod_{1\leq i\leq n, i\neq j} Res(f_i(x,y,z), f_j(x,y,z), z)) \\
&= c \cdot \prod_{1\leq i\leq n} T_{i,i}(x,y) \cdot \prod_{1\leq i,j\leq n, i\neq j} T_{i,j}(x,y),
\end{aligned}$$

where c is a constant. It is clear that $g(x,y)$ is a factor of $Res(f(x,y,z), \frac{\partial f(x,y,z)}{\partial z}, z)$ and any irreducible factor of $Res(f(x,y,z), \frac{\partial f(x,y,z)}{\partial z}, z)$ is contained in $g(x,y)$. So the projection curve of $\mathcal{S}$ is defined by the square-free polynomial whose components are all the irreducible components of $g(x,y)$. So the lemma holds. □

If there exists any component which is not a normal surface, take a transformation of coordinate system as (8) to insure that all components are normal surfaces in the new coordinate system. Then compute the projection curve of the new surface with the method mentioned above. For any surface $f(x,y,z)=0$, we present the following algorithm to compute its projection curve.

Algorithm 2. *The input is a polynomial $f(x,y,z)$. The output is a square-free polynomial $g(x,y)$, where $\mathcal{C}$: $g(x,y)=0$ is the normal projection curve of $f(x,y,z)$.*

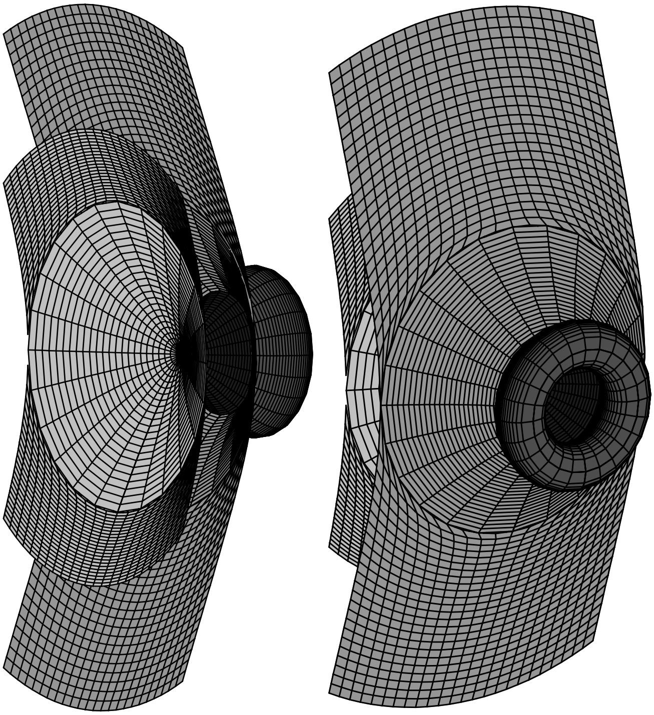


Fig. 1. An irreducible surface

1. Factor $f(x, y, z)$. Suppose $f(x, y, z)$ has a representation as (1), still denote $f(x, y, z)$ as (2).
2. If $n = 1$, compute the projection curve of $\mathcal{S}$ by Algorithm 1 and return it.
3. Else $(n > 1)$, do
   (a) Check whether $f_i(x, y, z)$ is a normal surface for all i. If there exists a component which is not a normal surface, it is clear that we can find a transformation as (8), such that each component is a normal surface in the new coordinate system. Still denote the surface as $f(x, y, z) = 0$.
   (b) Compute the projection curve of $f(x, y, z)$ by Lemma 2 and return its square-free part.

Example 1. Let us consider the following surface.

$$f(x, y, z) = (y^2 + z^2 - x^2 + 1/2 \cdot x^3 - 4)^2 - 16 \cdot x^2 + 8 \cdot x^3 = 0. \tag{10}$$

It is irreducible and normal. As is shown in Fig. 1. We can compute its projection curve by Algorithm 1.

$$Res(f(x, y, z), \frac{\partial f(x, y, z)}{\partial z}, z) = 4096 \cdot g_1(x, y)^4 \cdot g_2(x, y)^2 \cdot g_3(x, y) \cdot g_4(x, y),$$

where $g_1(x,y) = x$, $g_2(x,y) = x-2$, $g_3(x,y) = 2\cdot y^2 + 8\cdot y - 2\cdot x^2 + x^3 + 8$, $g_4(x,y) = 2\cdot y^2 - 8\cdot y - 2\cdot x^2 + x^3 + 8$. So we can derive its projection curve as follows.

$$g(x,y) = g_1(x,y)\cdot g_2(x,y)\cdot g_3(x,y)\cdot g_4(x,y). \tag{11}$$

3 Projection Curve Topology Determination

In this section, we will present algorithms to determine the topology of the normal projection curve obtained in the preceding section. Such algorithms already exist([14, 16]). But their outputs do not satisfy the requirement of our algorithm for the surface topology determination. Also, our algorithm gives an intrinsic representation for the topology of the given curve.

3.1 Notations

Definition 1. *A point $P_0(x_0,y_0)$ is said to be a singularity of an implicit algebraic curve $\mathcal{C}$: $g(x,y)=0$ if $g(x_0,y_0) = g_x(x_0,y_0) = g_y(x_0,y_0) = 0$, where $g(x,y)$ is square-free [23].*

Let $\mathcal{C} : g(x,y) = 0$ be the normal projection curve. We will consider the part of $\mathcal{C}$ inside a bounding box

$$\mathcal{B} = \{(x,y)|x_l \le x \le x_r, y_b \le y \le y_u\}$$

to be determined later. The intersection points of $\mathcal{C}$ and the boundary of $\mathcal{B}$ are called boundary points of $\mathcal{C}$. The part of $\mathcal{C}$ inside $\mathcal{B}$ (including the boundaries of $\mathcal{B}$) is denoted as $\mathcal{C}_{\mathcal{B}}$.

Definition 2. *A plane algebraic curve segment in a finite box is said to be a complete curve segment(CCS) if it is one of the following cases:*

1. *An isolated singularity P_i of $\mathcal{C} : g(x,y) = 0$, denoted as C_{P_i}.*
2. *A continuous curve segment from a singularity or a boundary point to a singularity or a boundary point, such that there is no singularities of $\mathcal{C}$ on the curve segment between the two endpoints. Denote $C^k_{i,j}$ to be the k-th curve segment from the singularity P_i to the singularity P_j; or $C_{i,j}$: the curve segment from the singularity P_i to the boundary point B_j; or $B^{-1}_{i,j}$: the curve segment from the boundary point B_i to the boundary point B_j. Note that $C^k_{i,j} = C^k_{j,i}$, $B^{-1}_{i,j} = B^{-1}_{j,i}$, $C_{i,j} \ne C_{j,i}$.*
3. *A closed continuous curve without singularities, denoted as C_Q, where Q is a point on the curve.*

Definition 3. *A cell of a plane curve in a bounding box is a closed region whose boundaries are CCSes or part of the boundaries of the box.*

Definition 4. *A curve segment sequence of a singularity of a plane curve is an ordered sequence of CCSes originating from the singularity. They are listed from left-up in the counter-clockwise order.*

Definition 5. *The topological representation of a plane algebraic curve within a bounding box consists of the following information.*

A bounding box: $\mathcal{B} = \{(x,y) | x_l \leq x \leq x_r, y_b \leq y \leq y_u\}$;
Boundary points: $\{B_i(x_{B_i}, y_{B_i})(b_i)[B_{i,j_1}, C_{j,i}(\text{ or } B^{-1}_{i,j}), B_{i,j_2}], i \in I_B\}$;
Singularities: $\{P_i(x_{P_i}, y_{P_i})(r_i)[\text{ Curve segment sequence of } P_i], i \in I_S\}$;
CCSes: $\{C^k_{i,j}(C_{c_1}, C_{c_2}), c_1, c_2 \in I_C\}$;
Cells: $\{C_k[\text{ The ordered boundaries of the cell}], k \in I_C\}$.

Here $(x_{B_i}, y_{B_i}), (x_{P_i}, y_{P_i})$ *are coordinates of points* B_i, P_i*, respectively;* I_S, I_B, I_C *are indexes of singularities, boundary points and cells, respectively;* $r_i(b_i)$ *is the discriminate distance (a positive number which will be defined below) of singularity* $P_i(B_i)$*;* C_{c_1}, C_{c_2} *are two cells beside the CCS* $C^k_{i,j}$*, here* $C^k_{i,j}$ *can also be CCS* $B^{-1}_{i,j}, C_{i,j}, C_{P_i}$ *or* C_Q*.*

3.2 Topology Determination

If the normal projection curve $\mathcal{C} : g(x,y) = 0$ of $\mathcal{S} : f(x,y,z) = 0$ is irreducible, then we use the following plane curve topology determination algorithm to compute the topology of $\mathcal{C}$, which is based on the algorithms in [9, 16].

Algorithm 3 (Irreducible algebraic curve topology determination). *The input is an irreducible plane algebraic curve* $\mathcal{C} : g(x,y) = 0$*. The output is the topological representation of* $\mathcal{C}_\mathcal{B}$*.*

1. Compute the discriminant $D(y) = \sum_{i=0}^{m} d_i y^i$ of $g(x,y)$ with respect to x and let y_u be a rational number which is larger than $\frac{\max\{|d_0|,\cdots,|d_{m-1}|\}}{|d_m|}$. Then by Cauchy's inequality, all the roots of $D(y) = 0$ are in the interval $(y_b = -y_u, y_u)$.
2. Compute the discriminant $\bar{D}(x)$ of $g(x,y)$ with respect to y and determine its real roots: $\alpha_1 < \ldots < \alpha_{s-1}$. Select two rational numbers x_l and x_r such that $x_l < \alpha_1$ and $x_r > \alpha_{s-1}$ and let $\alpha_0 = x_l$, $\alpha_s = x_r$. Now we have determined the bounding box $\mathcal{B}$. Then all the finite singularities of the curve are in the box.
3. Compute the real intersection points of $g(x,y) = 0$ and the lines $x - \alpha_0 = 0$ and $x - \alpha_s = 0$ in the interval $[y_b, y_u]$ and compute the real intersection points of $g(x,y) = 0$ and the lines $y - y_b = 0$ and $y - y_u = 0$ in the interval (x_l, x_r). The four vertexes of the box are $(x_l, y_u), (x_l, y_b), (x_r, y_u), (x_r, y_b)$. Denote these points in order as $B_i, i \in I_B$. Insure that the four endpoints are not on $\mathcal{C}$. If B_i is between its two adjacent boundary points B_{i_1}, B_{i_2}, then the discriminate distance of B_i is $b_i = \min\{\| B_i B_{i_1} \|, \| B_i B_{i_2} \|\}$. Compute the discriminate distance for each B_i (not including the vertexes).
4. For every $\alpha_i (i = 1, \cdots, s-1)$, do
   (a) Compute within $\mathcal{B}$ the real roots of $g(\alpha_i, y)$, $\beta_{i,0} < \ldots < \beta_{i,t_i}$.
   (b) For each point $P_{i,j} = (\alpha_i, \beta_{i,j})$, do
      i. Count the numbers of branches of $\mathcal{C}$ in $\mathcal{B}$ to the left and to the right. Denote $P_{i,j}$ as $P_l (l \in I_S)$ in order if $g_x(\alpha_i, \beta_{i,j}) = g_y(\alpha_i, \beta_{i,j}) = 0$, label $r_l = \min\{\alpha_i - \alpha_{i-1}, \alpha_{i+1} - \alpha_i, \beta_{i,j} - \beta_{i,j-1}, \beta_{i,j+1} - \beta_{i,j}\}(\beta_{i,-1} =$

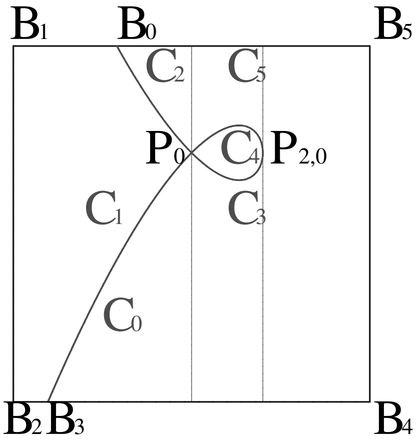


Fig. 2. Determine the topology of an irreducible curve

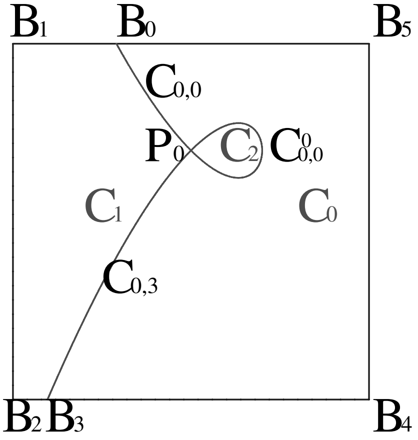


Fig. 3. The final topological figure of an irreducible plane curve

$y_b, \beta_{i,t_i+1} = y_u$) and record an ordered sequence of branches originating from the singularity from left-up to right-up in the counter-clockwise order, transform the branches to corresponding CCSes in the end.

ii. Label each cell in $\mathbf{D}_i = (\alpha_{i-1}, \alpha_i) \times (y_b, y_u)$; combine the two closed regions sharing the line segment $\overline{P_{i-1,j}P_{i-1,j+1}}$ and relabel the closed region. If any closed region is a cell, denote it as $C_k (k \in I_C)$ in order.

iii. Label each curve segment in the interval $\mathbf{D}_i$ and record the cells besides it, combine two curve segments(one in $\mathbf{D}_{i-1}$, the other in $\mathbf{D}_i$) if their unique common point $P_{i,j}$ is non-singular; relabel the new curve segment and record the cell(s) besides it. Now, we can obtain a set of CCSes and the corresponding cell(s) besides them.

5. Return corresponding information.

Example 2. Let us consider a component of the projection curve $\mathcal{C}$ defined by (11). Its equation is $g_3(x, y) = 2 \cdot y^2 - 8 \cdot y - 2 \cdot x^2 + x^3 + 8$ (Fig. 2).

Following Algorithm 3, we can obtain a finite box $\mathcal{B} = [-5, 5] \times [-5, 5]$. The boundary points are B_0, B_3 and four endpoints of the box are B_1, B_2, B_4, B_5. We can obtain $b_0 = 5 + a_1, b_3 = 5 + a_2$ (where a_1, a_2 will be defined in Example 3). And $\alpha_0 = -5, \alpha_1 = 0, \alpha_2 = 2, \alpha_3 = 5$.

Solve $g(\alpha_1, y) = 0$. We obtain one real root $y_{1,0} = 0$. The point $P_0 = (\alpha_1, y_{1,0})$ is a singularity of the curve. There are two branches originating from it on the left side and right side, respectively. The discriminate distance for P_0 is 2 ($\min\{\alpha_1 - \alpha_0, \alpha_2 - \alpha_1, y_u - y_{1,0}, y_{1,0} - y_d\}$). The closed region in $\mathbf{D}_1$ are C_0, C_1, C_2 as shown in Fig. 2. We can check that C_1 is a cell.

Solving $g(\alpha_2, y) = 0$, we obtain one point $P_{2,0}$. It is not a singularity. There are two branches originating from it on its left side and no branches originating from it on its right side. There is no boundary points in $\mathbf{D}_2$. Therefore we can connect the branches in order in $\mathbf{D}_2$. And the two curve segments from P_0 to $P_{2,0}$ compose a CCS of the given curve. Denote it as $C^0_{0,0}$. The closed region in $\mathbf{D}_2$ are C_3, C_4, C_5. C_0 and C_3, C_2 and C_5 share common line segments, and we can combine them as C_0, C_2, respectively. Since there is no boundary point on the boundary of $\mathbf{D}_3$, and there is no branches originating from $P_{2,0}$ in $\mathbf{D}_3$, the curve has no points in $\mathbf{D}_3$. C_0, C_2 share common line segments with $\mathbf{D}_3$ and we can combine them as C_0. In the end, we obtain the decomposition of the curve in Fig. 3.

The outputs about the topological representation of the curve are as follows.
The bounding box: $\mathcal{B} = \{(x, y)| -5 \leq x \leq 5, -5 \leq y \leq 5\}$.
Boundary points: $\{B_0(a_1, 5)(5 + a_1)[B_{0,1}, C_{0,0}, B_{0,5}]$, $B_1(-5, 5)$, $B_2(-5, -5)$, $B_3(a_2, -5)(5 + a_2)[B_{3,2}, C_{0,3}, B_{3,4}]$, $B_4(5, -5)$, $B_5(5, 5)\}$.
Singularities: $\{P_0(0, 2)(2)[C_{0,0}, C_{0,3}, C^0_{0,0}, C^0_{0,0}]\}$.
CCSes: $\{C_{0,0}(C_0, C_1), C_{0,3}(C_0, C_1), C^0_{0,0}(C_0, C_2)\}$.
Cells: $\{C_0[B_{3,4}, B_{4,5}, B_{5,0}, C_{0,0}, C^0_{0,0}, C_{0,3}]$, $C_1[B_{0,1}, B_{1,2}, B_{2,3}, C_{0,3}, C_{0,0}]$, $C_2[C^0_{0,0}]\}$.

The following algorithm is to determine the topology of any square-free algebraic curve.

Algorithm 4 (Plane curve topology determination). *The input is $\mathcal{C}$: $g(x, y) = 0$. The output is the same as Algorithm 3.*

1. If $g(x, y)$ is irreducible, determine the topology of $\mathcal{C}$ by Algorithm 3.
2. Else ($g(x, y)$ is reducible), suppose $g(x, y)$ has a representation as (4).
   (a) Compute a bounding box for each component $g_i(x, y)(i = 1, \cdots, m)$; Compute the intersection points of any two components $g_i(x, y), g_j(x, y)$ $(i, j = 1, \cdots, m, i \neq j)$. Choose a box which contains all boxes and intersection points as the bounding box of $g(x, y) = 0$. Compute the boundary points of $g(x, y) = 0$. Compute the discriminate distance for each boundary points.
   (b) Separate the vertical lines which have a form as $A \cdot x + B = 0$ from $g(x, y)$ if they exist. Of course, we can denote them as $L_t(x, y) = x - c_t = 0$ $(t = 0, \cdots, L)$. Denote all the remainder components of $g(x, y)$ as $g_0(x, y)$. Suppose it is $g_0(x, y) = g_1(x, y) \cdots g_s(x, y)$, where $s = m - L$.
   (c) Solve $Res(g_i(x, y), \frac{\partial g_i(x,y)}{\partial y}, y) = 0$ and $Res(g_i(x, y), g_j(x, y), y) = 0$ for all $i, j = 0, \cdots, m_L (i \neq j)$. Put their roots and $c_t(t = 0, \cdots, L)$ together and rewrite them as $\alpha_k(k = 1, \cdots, l-1, \alpha_k < \alpha_{k+1})$. Let $\alpha_0 = x_l, \alpha_l = x_r$ be rational numbers such that $\alpha_0 \leq \alpha_1, \alpha_l \geq \alpha_{l-1}$.
   (d) For every α_k, to $g_0(x, y)$, we can do the same work as Algorithm 3 in step 4. Note that when $\alpha_k = c_t(t = 0, \cdots, L)$, all the real intersection points of $g_0(x, y) = 0$ and the line $x - \alpha_k = 0$ in the interval (y_b, y_u), denoted as $P_{k,j}(j = 0, \cdots, t_k)$, are singularities of $g(x, y) = 0$, and line segments $\overline{P_{k,j}P_{k,j+1}}(j = 0, \cdots, t_k - 1)$,$\overline{B_{k,0}P_{k,0}}$, $\overline{P_{k,t_k}B_{k,1}}$ are CCSes of $g(x, y) = 0$, where $B_{k,0}, B_{k,1}$ are intersection points of the line $x - \alpha_k = 0$ and the boundary of the box. We obtain the topology information of $g(x, y) = 0$ in the end.
3. Return the corresponding topological information of $\mathcal{C}$.

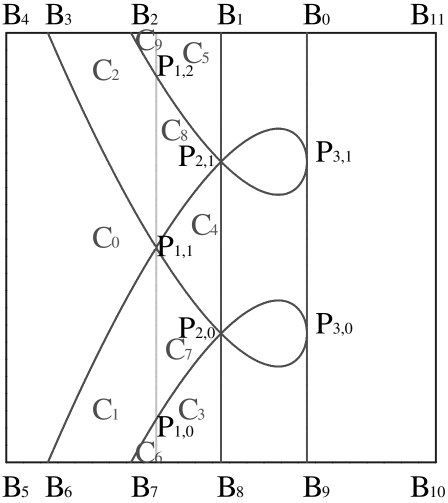


Fig. 4. Topology determination of a curve

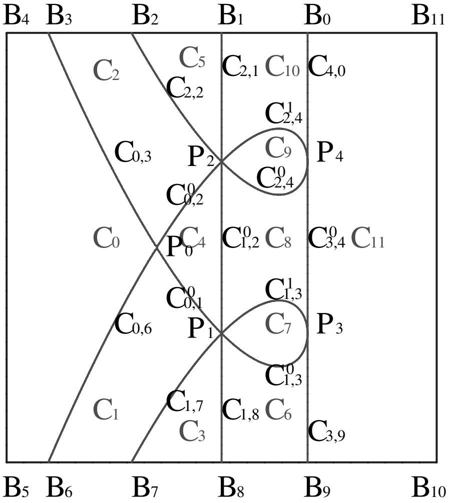


Fig. 5. The topology of a curve

Remark. If $\mathcal{C}$ has no critical points (the points which satisfy $g(x,y)=g_y(x,y)=0$) on $\mathcal{C}$, we can solve $g(0,y)=0$. If $g(0,y)=0$ has no real roots, solve $f(0,0,z)=0$. The surface $\mathcal{S}$ has no real parts if it has no real roots. And $\mathcal{S}$ is topologically equivalent to n parallel planes if the equation $f(0,0,z)=0$ has n real roots. If $g(0,y)=0$ has real roots $y_0,\cdots,y_m$, let the finite box be $\mathcal{B}=[-1,1]\times[y_0-1,y_m+1]$, we can obtain the boundary points $B_i(i=0,\cdots,2m+1)$ and CCSes $B^{-1}_{0,2m+1}, B^{-1}_{1,2m},\cdots,B^{-1}_{m,m+1}$ of $\mathcal{C}$.

Example 3. Consider the projection curve defined by (11) as an example of a reducible curve.

Following Algorithm 4, here $g_1(x,y)$ and $g_2(x,y)$ are vertical lines, we remove them from $g(x,y)$.

$Res(g_3(x,y),\frac{\partial g_3(x,y)}{\partial y},y)=16\cdot x^3-32\cdot x^2=0$, whose real roots are 2, 0. $Res(g_4(x,y),\frac{\partial g_4(x,y)}{\partial y},y)=16\cdot x^3-32\cdot x^2=0$, whose real roots are also 2, 0. $Res(g_3(x,y),g_4(x,y),y)=-2\cdot x^2+x^3+8=0$, whose real root is $x_{3,4}=-\frac{1}{3}\cdot\sqrt[3]{100+12\cdot\sqrt{69}}-\frac{4}{3\cdot\sqrt[3]{100+12\cdot\sqrt{69}}}+\frac{2}{3}$. So we have $\alpha_0=-5$, $\alpha_1=x_{3,4}$, $\alpha_2=0$, $\alpha_3=2$, $\alpha_4=5$.

Then we can obtain the bounding box $\mathcal{B}=[-5,5]\times[-5,5]$ and the boundary points: $B_0(2,5)$, $B_1(0,5)$, $B_2(a_1,5)$, $B_3(a_2,5)$, $B_6(a_2,-5)$, $B_7(a_1,-5)$, $B_8(0,-5)$, $B_9(2,-5)$. Add the endpoints $B_4(-5,5)$, $B_5(-5,-5)$, $B_{10}(5,-5)$, $B_{11}(5,5)$ in the boundary point list. And $b_0=2, b_1=2, b_2=a_1-a_2, b_3=5+a_2, b_6=5+a_2, b_7=a_1-a_2, b_8=2, b_9=2$. Here $a_1=-\frac{\sqrt[3]{235+9\cdot\sqrt{681}}}{3}-\frac{4}{3\cdot\sqrt[3]{235+9\cdot\sqrt{681}}}+\frac{2}{3}$, $a_2=-\frac{\sqrt[3]{1315+21\cdot\sqrt{3921}}}{3}-\frac{4}{3\cdot\sqrt[3]{1315+21\cdot\sqrt{3921}}}+\frac{2}{3}$.

Solving $g_3(\alpha_1, y) \cdot g_4(\alpha_1, y) = 0$, we can get $y = -4, 0, 4$. They correspond to $P_{1,0}, P_{1,1}, P_{1,2}$ in Fig. 4. We can easily find that only point $P_{1,1}$ is a singularity. We rename it as P_0. Its corresponding positive number is $\min\{4-0, 0-(-4), \alpha_2 - \alpha_1, \alpha_1 - \alpha_0\} = -\alpha_1$. We can show that the curve segments in $\mathbf{D}_1 = [\alpha_0, \alpha_1] \times [y_b, y_u]$ are $\wideparen{P_{1,0}B_7}, \wideparen{P_{1,1}B_6}, \wideparen{P_{1,1}B_3}, \wideparen{P_{1,2}B_2}$. And $C_{0,6} = \wideparen{P_{1,1}B_6}, C_{0,3} = \wideparen{P_{1,1}B_3}$ are CCSes of the given curve. The regions in the interval $\mathbf{D}_1$ are C_6, C_1, C_0, C_2, C_9. Their boundaries are as shown in Fig. 4. And only C_0 is a cell. Its boundaries are $C_{0,3}, B_{3,4}, B_{4,5}, B_{5,6}, C_{0,6}$.

Solve $g_3(\alpha_2, y) \cdot g_4(\alpha_2, y) = 0$. Its real roots are $-2, 2$. We can find two singularities $P_{2,0}, P_{2,1}$, whose corresponding positive numbers are $-\alpha_1$. We rename them as P_1, P_2. In the interval $\mathbf{D}_2 = [\alpha_1, \alpha_2] \times [y_b, y_u]$, the curve segments and regions are shown in Fig. 4. We can combine curve segment $\wideparen{P_{1,0}B_7}$ in $\mathbf{D}_1$ and curve segment $\wideparen{P_{2,0}P_{1,0}}$ in $\mathbf{D}_2$ as a CCS $C_{1,7} = \wideparen{P_{2,0}B_7}$ and combine regions C_3, C_6 as C_3. Combine C_1, C_7 as C_1. Combine C_2, C_8 as C_2, and combine C_5, C_9 as C_5. Since $x - \alpha_2$ is $g_1(x, y)$, the line segments $\overline{P_{2,0}B_6}, \overline{P_{2,0}P_{2,1}}, \overline{P_{2,1}B_1}$ are CCSes $C_{1,6}, C^0_{1,2}, C_{2,1}$ of $g(x, y) = 0$. C_4 is a cell. The real roots of $g_3(\alpha_3, y) \cdot g_4(\alpha_3, y) = 0$ are -2, 2. And $x - \alpha_3$ is $g_2(x, y)$. Following Algorithm 4, we can derive the topology of $g(x, y) = 0$ as Fig. 5. The positive numbers of the five singularities are $-\alpha_1, -\alpha_1, -\alpha_1$, 2 and 2 respectively.

The output are as follows.

The bounding box: $\mathcal{B} = [-5, 5] \times [-5, 5]$.

Boundary points: $\{B_0(2, 5)(2)[B_{0,1}, C_{4,0}, B_{0,11}]$, $B_1(0, 5)(2)[B_{1,2}, C_{2,1}, B_{1,0}]$, $B_2(a_1, 5)(a_1 - a_2)[B_{2,3}, C_{2,2}, B_{2,1}]$, $B_3(a_2, 5)(5 + a_2)[B_{3,4}, C_{0,3}, B_{3,2}]$, $B_4(-5, 5)$, $B_5(-5, -5)$, $B_6(a_2, -5)(5 + a_2)[B_{6,5}, B_{6,7}, C_{0,6}]$, $B_7(a_1, -5)(a_1 - a_2)[B_{7,6}, B_{7,8}, C_{1,7}]$, $B_8(0, -5)(2)[B_{8,7}, B_{8,9}, C_{3,9}]$, $B_9(2, -5)(2)[B_{8,9}, B_{8,10}, C_{3,9}]$, $B_{10}(5, -5)$, $B_{11}(5, 5)\}$. B_4, B_5, B_{10}, B_{11} are vertexes of the box.

Singularities: $\{P_0(\alpha_1, 0)(-\alpha_1)[C_{0,3}, C_{0,6}, C^0_{0,1}, C^0_{0,2}]$, $P_1(0, -2)(-\alpha_1)[C^0_{0,1}, C_{1,7}, C_{1,8}, C^0_{1,3}, C^1_{1,3}, C^0_{1,2}]$, $P_2(0, 2)(-\alpha_1)[C_{2,2}, C^0_{0,2}, C^0_{1,2}, C^0_{2,4}, C^1_{2,4}, C_{2,1}]$, $P_3(2, -2)(2)[C^1_{1,3}, C^0_{1,3}, C_{3,9}, C^0_{3,4}]$, $P_4(2, 2)(2)[C^1_{2,4}, C^0_{2,4}, C^0_{3,4}, C_{4,0}]\}$.

CCSes: $\{C_{0,6}(C_1, C_0)$, $C_{0,3}(C_0, C_2)$, $C_{1,7}(C_1, C_3)$, $C^0_{0,2}(C_2, C_4)$, $C^0_{0,1}(C_4, C_1)$, $C_{2,2}(C_2, C_5)$,$C_{1,8}(C_3, C_6)$,$C^0_{1,2}(C_4, C_8)$,$C_{2,1}(C_5, C_{10})$, $C^0_{1,3}(C_6, C_7)$, $C^1_{1,3}(C_7, C_8)$, $C^0_{2,4}(C_8, C_9)$, $C^1_{2,4}(C_9, C_{10})$, $C_{3,9}(C_6, C_{11})$, $C^0_{3,4}(C_8, C_{11})$, $C_{4,0}(C_{10}, C_{11})\}$.

Cells: $\{C_0[B_{3,4}, B_{4,5}, B_{5,6}, C_{0,6}, C_{0,3}]$, $C_1[B_{6,7}, C_{1,7}, C^0_{0,1}, C_{0,6}]$, $C_2[C_{0,3}, C^0_{0,2}, C_{2,2}, B_{2,3}]$, $C_3[B_{7,8}, C_{1,8}, C_{1,7}]$, $C_4[C^0_{0,2}, C^0_{0,1}, C^0_{1,2}]$, $C_5[C_{2,2}, C_{2,1}, B_{1,2}]$, $C_6[B_{8,9}, C_{3,9}, C^0_{1,3}, C_{1,8}]$, $C_7[C^0_{1,3}, C^1_{1,3}]$, $C_8[C^1_{1,3}, C^0_{3,4}, C^0_{2,4}, C^0_{1,2}]$, $C_9[C^0_{2,4}, C^1_{2,4}]$, $C_{10}[C^1_{2,4}, C_{4,0}, B_{0,1}, C_{2,1}]$, $C_{11}[B_{9,10}, B_{10,11}, B_{11,0}, C_{4,0}, C^0_{3,4}, C_{3,9}]\}$.

4 Space Curve Segmentation and Surface Patch Composition

In this section, we will determine the position of each space curve segment and each surface patch of $\mathcal{S}$. The algorithm works as follows:

First, we need to determine the points of $\mathcal{S}$ on each line lifted from a boundary point $B_i(\bar{x}_i, \bar{y}_i)(i \in I_B)$ or a singularity $P_i(x_i, y_i)(i \in I_S)$ of $\mathcal{C}$: $g(x, y) = 0$.

These points are the endpoints of the space curve segments. Second, we need to determine how many space curve segments originating from each endpoint. Then we can determine all space curve segments of $\mathcal{S}$. Third, we need to compute the number of surface patches originating from each space curve segment. Finally, we can determine the surface patches in each region from bottom to top by pointing out their boundaries.

4.1 Notations

In order to describe our algorithm clearly, we present the following definitions. Let us assume that we have already obtained a topological representation for the projection curve of $\mathcal{S}$.

Definition 6. *A complete cylindrical patch (CCP) $SC_{i,j}^k$ is a cylindrical patch lifted from a CCS $C_{i,j}^k$ obtained in section 3. Then $SC_{i,j}^k = C_{i,j}^k \times [-N, N]$, where N is a positive number that will be defined later. $SC_{i,j}$, $SB_{i,j}$, $SB_{i,j}^{-1}$, SC_{P_i}, SC_{Q_i} can be defined similarly.*

Definition 7. *A cell body is a body lifted from a cell obtained in section 3. We can denote it as CC_i, where C_i is a cell of the projection curve. Two cell bodies share a CCP as a boundary. When a CCS is an isolated singularity, there is only one cell body beside the CCP corresponding to the CCS.*

Definition 8. *A complete space curve segment (CSCS) of $\mathcal{S}$: $f(x,y,z) = 0$ is a space curve segment which is an intersection of a CCP and $\mathcal{S}$. We denote it as $C_{i,j}^{k,l}(C_{i,j}^{-1,l}, B_{i,j}^l, B_{i,j}^{-1,l}, V_{i,l}, C_{Q_i}^l)$ if its corresponding CCS in the plane is $C_{i,j}^k(C_{i,j}, B_{i,j}, B_{i,j}^{-1}, C_{P_i}, C_{Q_i})$, where l is an index starting from bottom to up.*

Definition 9. *A complete surface patch (CSP) of $\mathcal{S}$: $f(x,y,z) = 0$ is a surface patch which is part of $\mathcal{S}$, and its boundaries are several CSCSes. We can denote it as S_i^l if it is the l-th surface patch in cell body CC_i from bottom to up.*

Definition 10. *A critical curve of a surface is a space curve satisfying $f(x,y,z) = f_z(x,y,z) = 0$, where (x,y,z) is any point on the curve.*

Definition 11. *A singular curve of a surface is a space curve, which satisfies $f(x,y,z) = f_x(x,y,z) = f_y(x,y,z) = f_z(x,y,z) = 0$ for any point (x,y,z) on the curve.*

Let $\mathcal{S}$ be the surface, $\mathcal{C}$ the projection curve of $\mathcal{S}$, $\mathcal{B}$ the bounding box of $\mathcal{C}$, $\mathcal{C}_{\mathcal{B}}$ part of the projection curve within $\mathcal{B}$, $V_{i,j}$(or $V_{i,j}^0$)$(j = 0, \cdots, t_i)$ the points of $\mathcal{S}$ on the line lifted from a singularity P_i (or a boundary point B_i)

Definition 12. *The topological information of a surface include the following information:*

The point lists: $\{V_{i,j}$ (or $V_{i,j}^0)(j = 0, \cdots, t_i, i \in I_S$ (or $i \in I_B))\}$, which are corresponding to certain singularities (or boundary points) of $\mathcal{C}_{\mathcal{B}}$. For example,

$B_i[V^0_{i,0}, \cdots, V^0_{i,t_i}], P_j[V_{j,0}, \cdots, V_{j,t_j}]$ are point lists corresponding to a boundary point and a singularity.

The CSCS lists: {The CSCS list corresponding to each line segment $B_{i,j}$ and each CCS of $\mathcal{C}_\mathcal{B}$}. For instance, $B_{i,j}[B^0_{i,j}(V^0_{i,0}, V^0_{j,0}), \cdots, B^p_{i,j}(V^0_{i,i_1}, V^0_{j,j_1})]$, $C^k_{i,j}[C^{k,0}_{i,j}(V_{i,0}, V_{j,0}), \cdots, C^{k,l}_{i,j}(V_{i,i_2}, V_{j,j_2})]$, where the two points for each CSCS are their endpoints.

The CSP lists: {The CSPs (including their boundary CSCSes) corresponding to each cell}. For instance, $C_i(n)\{S^0_i$[The ordered boundary CSCSes of the surface patch]$, \cdots, S^{n-1}_i$[The ordered boundary CSCSes of the surface patch]$\}$.

The CSCSes lists form a curvilinear wireframe of the surface. The boundaries of the CSPs in the CSP lists are the CSCSes in the CSCS lists. So they are determined by the curvilinear wireframe.

4.2 Basic Theorems and Algorithms

Theorem 1. *Let $\mathcal{S} : f(x, y, z) = 0$ be a normal surface, $\mathcal{C} : g(x, y) = 0$ the projection curve computed by Algorithm 2, $\mathcal{B}$ the finite box obtained in Algorithm 4 for $g(x, y)$. For any point (x_0, y_0) inside $\mathcal{B}$, the real roots of $f(x_0, y_0, z) = 0$ are finite, that is, there exists a positive number N such that for any real root z_0 of $f(x_0, y_0, z) = 0$, $-N < z_0 < N$.*

Proof. It is clear that there is no surface patch of $\mathcal{S}$ which is approaching to infinity inside $\mathbf{B}$. This is guaranteed by conditions *C1* and *C2*. So the theorem holds. □

In fact, we can compute the number N. We can compute the maximum and minimum of the z-coordinate inside $\mathcal{B}$ (including its boundary). We use equation (10) as an example. As we know, $\mathcal{B} = [x_l, x_r] \times [y_b, y_f] = [-5, 5] \times [-5, 5]$. Compute the maximum and minimum in z-direction of $f(x, y, z) = 0$ for $(x, y) \in [-5, 5] \times [-5, 5]$. We can use Wu's finite kernel method([24]). The number with the largest absolute value is $2 + 5/2 \cdot \sqrt{14}$. Choose N to be a rational number which is larger than the absolute value of the computed number. Here we can choose $N = 12$.

Theorem 2. *All the notations are with the same meaning as Theorem 1. $\mathcal{S}$ and the part of $\mathcal{S}$ in the cube $\mathbf{B} = \mathcal{B} \times [-N, N]$ have the same topology.*

Proof. Denote the part of $\mathcal{S}$ inside the cube $\mathbf{B}$ as $\mathcal{S}_\mathbf{B}$. Let $\mathbf{B}_1$ be a cube containing $\mathbf{B}$ strictly. We will show that the part of $\mathcal{S}$ inside $\mathbf{B}$ and the part of $\mathcal{S}$ inside $\mathbf{B}_1$ have the same topology. This is because, the part of $\mathcal{S}$ between $\mathbf{B}$ and $\mathbf{B}_1$ can be seen as surfaces (or lines) lifted from the intersection of $\mathcal{S}$ and $\mathbf{B}$ without adding new intersections. Topologically, they are the same with cylindrical surfaces. Hence, adding these surfaces does not change the topology of $\mathcal{S}_\mathbf{B}$. So the theorem holds. □

We further assume that there is no singularities on the CCP lifted from any CCS of $\mathcal{C}$. In fact, we can find a new coordinate system such that the isolated singularities and the intersection points of the critical curves of the surface are projected onto the singularities of the projection curve.

Definition 13. *Given a univariate function $P(x)$, let $P_0(x) = P(x), P_1(x) = P'(x)$ and define the Sturm functions by*

$$P_i(x) = -(P_{i-2}(x) - P_{i-1}(x)[\frac{P_{i-2}(x)}{P_{i-1}(x)}]),$$

where $[\frac{P_{i-2}(x)}{P_{i-1}(x)}]$ is a polynomial quotient. The chain is terminated when $P_n(x)$ is a constant. Then $P_0(x)$, $P_1(x)$, $\cdots$, $P_n(x)$ is the Sturm functions (more details can be found in [17, 26]) of $P(x)$.

Definition 14. *Sign-changing number of Sturm functions of $P(x)$ at point $x = a$ is the number of sign changes on the Sturm functions of $P(x)$ evaluated at point $x = a$. That is, the number of sign changes of $P_0(a), P_1(a), \cdots, P_n(a)$.*

The following algorithm is to isolate the real roots of a polynomial $T(x) \in \mathbb{R}[x]$. The difference between the algorithm and general algorithm is that the isolated points of our algorithm is not a root of $T(x)$. For more detail, one can see [17, 18, 26].

Algorithm 5. *(Real Root-Isolating) The input are Sturm functions of a polynomial $T(x)$ and an interval (a, b) (where a, b are rational numbers,$T(a) \neq 0, T(b) \neq 0$, $T(x) \in R[x]$). The output is a series of ordered rational numbers in (a, b), such that there is a real root of $T(x) = 0$ between each pair of adjacent numbers.*

1. Compute the sign-changing numbers $V(a), V(b)$ of the Sturm functions of $T(x)$ at $x = a, x = b$, respectively. $V(a) - V(b)$ is the number of real roots between (a, b) by Sturm theorem. Let the rational number set be $N_s := \{a, b\}$. If $V(a) - V(b) = 0$, return $\emptyset$. If $V(a) - V(b) = 1$, return N_s.
2. When $V(a) - V(b) > 1$, if $T(\frac{a+b}{2}) \neq 0$, let $c = \frac{a+b}{2}$, else choose another rational number c near $\frac{a+b}{2}$ in (a, b) insuring that $T(c) \neq 0$.
   (a) If $V(a)-V(c) > 1$ and $V(c)-V(b) > 1$, $N_s := N_s \bigcup \{c\}$; let i. $a = a, b = c$, respectively, ii. $a = c, b = b$, respectively, go to 2.
   (b) Else if $V(a) - V(c) = 1$ and $V(c) - V(b) > 1$, $N_s := N_s \bigcup \{c\}$; let $a = c, b = b$, respectively, go to 2.
   (c) Else if $V(a) - V(c) > 1$ and $V(c) - V(b) = 1$, $N_s := N_s \bigcup \{c\}$; let $a = a, b = c$, respectively, go to 2.
   (d) Else if $V(a) - V(c) = 0$ and $V(c) - V(b) > 1$, let $a = c, b = b$, go to 2.
   (e) Else if $V(a) - V(c) > 1$ and $V(c) - V(b) = 0$, let $a = a, b = c$, go to 2.
3. Return the ordered rational numbers N_s.

Example 4. Continuing from Example 3, we want to isolate the points on the line lifted from $P_3(2, -2)$ in Fig. 5. Here the input are $f(2, -2, z) = z^4$ and $(a, b) = (-12, 12)$. The equation has only one real root $z = 0$. We can obtain its isolated points $W_{3,0}(2, -2, -12), W_{3,1}(2, -2, 12)$. There is a point $V_{3,0}$ of the surface on the line segment $\widetilde{W_{3,0}W_{3,1}}$ between $W_{3,0}, W_{3,1}$.

Given a point, a positive number and a plane curve (the point can be on the curve or not on the curve), the following algorithm is to find the circle whose center is the point, which is the minimal circle among the circles tangent to the curve.

Algorithm 6. *The inputs are a plane algebraic curve $T(x, y) = 0$, a positive number r and a point $P_0(x_0, y_0)$. The output is a positive number which is equal to half of the minimal of the extremum distance r_{min} from P_0 to the curve $T(x, y) = 0$ and r.*

1. Let $L(x, y, \lambda) = (x - x_0)^2 + (y - y_0)^2 + \lambda T(x, y)$.
2. Eliminating x and λ from $\{2(x-x_0)+\lambda T_x(x, y), 2(y-y_0)+\lambda T_y(x, y), T(x, y)\}$ in the order $\{\lambda \succ x \succ y\}$, we can obtained a univariate polynomial $P(y)$.
3. Solve $P(y)$ in the interval (y_0-r, y_0+r). If there is no real root in the interval, return $r/2$; Else, get corresponding $x_{i,j}$ for each real root y_i in the interval $(x_0 - r, x_0 + r)$. If there is no real roots in the interval, return $r/2$; else, let $R = min_{i,j} \sqrt{(x - x_{i,j})^2 + (y_0 - y_i)}$, if $R \leq r$, return $R/2$, else, return $r/2$.

Remark. The step 2 of this algorithm is based on a method of Wu to find extremal values. One can find more details in [24].

Example 5. Continuing from Example 4, let $g(x, y)$ be the curve defined by (11), $P_3 = (2, -2)$. The input is $g(x, y)$ and the positive number 2 corresponding to P_3. With this algorithm, we can find that the minimal positive extremum from P_3 to $g(x, y)$ is 2. So the output is 1.

4.3 Compute the Space Curve Segments

To each singularity $P_i(x_i, y_i)(i \in I_S)$ (or boundary point $B_i(x_i, y_i)(i \in I_B)$)of $g(x, y) = 0$, there is a sequence of CCSes $C^{k_1}_{i,j_1}, \cdots, C^{k_t}_{i,j_t}$ originating from it. Here the CCSes in the sequence can also be $C_{i,j}$ or boundary line segments $B_{i,j}$ (for B_i only). Lifting them up, we can obtain a sequence of CCPs $SC^{k_1}_{i,j_1}, \cdots, SC^{k_t}_{i,j_t}$. The point $P_i(x_i, y_i)$ corresponds to a vertical line $\{x = x_i, y = y_i\}$. There are some points $V_{i,j}(j = 0, \cdots, s_i)$ of $\mathcal{S}$ on the line. There are some CSCSes $C^{k_l,m}_{i,j_l}(m = 0, \cdots, t_{i,j,k})$ on each CCP $SC^{k_l}_{i,j_l}(l = 1, 2, \cdots, t)$ originating from $V_{i,j}$. We need to determine the CSCSes originating from each $V_{i,j}$ on each CCP. The following algorithm is to do this.

Algorithm 7. *The inputs are a real algebraic surface $\mathcal{S} : f(x, y, z) = 0$, its projection curve $\mathcal{C} : g(x, y) = 0$, a point $P_i(x_i, y_i)$ on $\mathcal{C}$, the discriminate distance r_i of P_i and a sequence of CCSes $\{C^{k_1}_{i,j_1}, \cdots, C^{k_t}_{i,j_t}\}$ originating from P_i. The outputs are a sequence of points $V_{i,j}(j = 0, \cdots, s_i)$ of $\mathcal{S}$ on the line lifted from P_i, a set of sequences of CSCSes $\{C^{k_l,m}_{i,j_l}, m = 0, \cdots, t_{i,j,k}\}$ for each $C^{k_l}_{i,j_l}$. Note that we only know one endpoint of the CSCSes. But we can compute the corresponding information for the other endpoint by this algorithm, then the CSCS is determined.*

1. Isolate the real roots of $f(x_i, y_i, z) = 0$ by Algorithm 5 and obtain the isolating values $z_{i,0}, z_{i,1}, \cdots, z_{i,s_i}$. Denote $(x_i, y_i, z_{i,j})$ as $W_{i,j}$. There exists a point of $\mathcal{S}$, $V_{i,j}$, which is on the line $\{x = x_i, y = y_i\}$ and between points $W_{i,j}$ and $W_{i,j+1}$. For an instance, please see Fig. 6.
2. From $r_i, P_i, g(x, y) = 0$, we can obtain a positive number R_i by Algorithm 6. It is clear that the number of intersection points of the circle $(x - x_i)^2 + (y - y_i)^2 = r^2 (0 < r \leq R_i)$ and $\mathcal{C}$ is equal to the number of the CCSes in the input sequence.
3. In plane $z = z_{i,j} (j = 0, 1, \cdots, s_i)$, from $r_i, P_i, f(x, y, z_{i,j}) = 0$, we can obtain a positive number $r_{i,j}$ by Algorithm 6. Still denote the minimal among $\{R_i, r_{i,0}, \cdots, r_{i,s_i}\}$ as $r_i (r_i \leq R_i)$.
4. Compute the real intersection points of the equations $\{(x - x_i)^2 + (y - y_i)^2 = r_i^2, g(x, y) = 0\}$. We can determine a point $P_{i,j_l}^{k_l}$ on $C_{i,j_l}^{k_l}, l = 1, \cdots, t$. Denote them as $\{P_{i,j_1}^{k_1}, P_{i,j_2}^{k_2}, \cdots, P_{i,j_t}^{k_t}\}$.
5. For each $P_{i,j_l}^{k_l}(x_{i,j_l,k_l}, y_{i,j_l,k_l}) (l = 1, \cdots, t)$, compute the number of real roots of $f(x_{i,j_l,k_l}, y_{i,j_l,k_l}, z) = 0$ in the interval $(z_{i,j}, z_{i,j+1}) (j = 0, 1, \cdots, s_i - 1)$. It is the number of CSCSes originating from $V_{i,j}$ on the CCP $SC_{i,j_l}^{k_l}$. So we can determine the CSCSes on each CCP: one of their two endpoints is on the line lifted from P_i. Their order on the CCPs is from bottom to top. Denote them as $C_{i,j_l}^{k_l,m}$. If there does not exist a real root in the interval $(z_{i,0}, z_{i,s_i})$, delete the CCS from the topology information and combine the cells divided by it.
6. Return the corresponding information.

Remark. In Algorithm 7, if the singularity is an isolated point of $\mathcal{C}$, we need not to compute it by this algorithm. If the input sequence of CCSes may include CCS like $C_{i,j}$ (the endpoints are a singularity and a boundary point), the algorithm is also valid. The CSCSes on the CCP $SC_{i,j}$ are determined by computing P_i with this algorithm. For a boundary point, the algorithm is also valid. Since the numbers of CSCSes originating from the points of $\mathcal{S}$ on the line lifted from P_i, P_j are the same, we can determine all the CSCSes on $SC_{i,j}^k$ after we compute P_i, P_j for the surface with this algorithm.

Theorem 3. *Algorithm 7 provides the correct output.*

Proof. We will prove that we can obtain what we want from Algorithm 7. From step 2 and step 3, it is clear that there is no other critical curves of $\mathcal{S}$ in the cylindrical body $\mathbf{D} = \{(x, y, z) | (x - x_i)^2 + (y - y_i)^2 \leq r_i^2, -N < z < N\}$, which can be projected onto the XY-plane except $\{C_{i,j_1}^{k_1}, \cdots, C_{i,j_t}^{k_t}\}$. And the discs $\{(x, y, z_{i,j}) | (x - x_i)^2 + (y - y_i)^2 \leq r_i^2\} (j = 0, \cdots, s_i)$ isolate the CSCSes originating from each $Vi, j (j = 0, \cdots, s_i - 1)$ on each CCP $SC_{i,j_l}^{k_l} (l = 1, \cdots, t)$ in $\mathbf{D}$. Then we will prove that the number of CSCSes originating from $V_{i,j} (j = 0, \cdots, s_i - 1)$ on the CCP $SC_{i,j_l}^{k_l} (l = 1, \cdots, t)$ is equal to the number of real roots of equation $f(x_{i,j_l,k_l}, y_{i,j_l,k_l}, z) = 0$ in the interval $(z_{i,j}, z_{i,j+1}) (j = 0, 1, \cdots, s_i - 1)$. Since the total number of CSCSes originating from $V_{i,j}$ for each j is equal to the number of CSCSes originating from the points of $\mathcal{S}$ on the line lifted

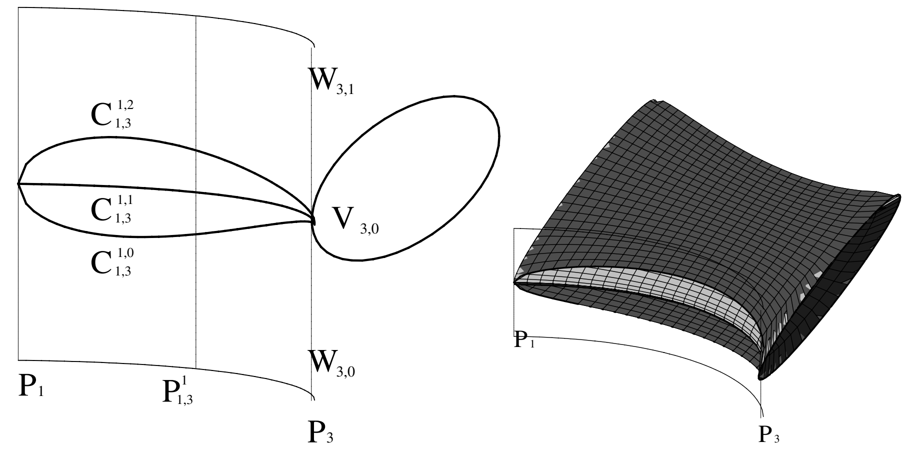


Fig. 6. Compute P_3 with Algorithm 7

form P_{j_l}, each CSCS originating from $V_{i,j}$ should connect one point on the line lifted from P_{j_l}. So if the conclusion is not right, there must exist a point on one CSCS originating from $V_{i,j}$ in **D**, which is also a point on a critical curve of $\mathcal{S}$. Projecting the critical curve onto the XY-plane, it must share a singular point with CCS $C^{k_l}_{i,j_l}$. This is in contradiction with the given condition. So the algorithm is valid. □

Example 6. Continuing from Example 5, let us consider P_3 with this algorithm. The inputs are $f(x,y,z)=0, g(x,y)=0, P_3(2,-2)(2)[C^1_{1,3}, C^0_{1,3}, C_{3,9}, C^0_{3,4}]$. We have known that there is only one real point $V_{3,0}$ of the surface on the line lifted from P_3 and R_3 equals 1. Its isolated points are $W_{3,0}(2,-2,-12), W_{3,1}(2,-2,12)$ (Fig. 6). In step 3, we can obtain 1 by Algorithm 6 if the input is 2 and $f(x,y,-12)$(or $f(x,y,12)$). So $r_3=1$. In order to illustrate our method simply, we choose the discriminate distance as a number less than 1: $\sqrt{13}/4$. Solving the equations $\{(x-2)^2+(y+2)^2-13/16=0, g(x,y)=0\}$, we can obtain the following points: $(\frac{3}{2},-\frac{5}{4})$, $(\frac{3}{2},-\frac{11}{4})$, $(2,-2-\frac{\sqrt{13}}{4})$ and $(2,-2+\frac{\sqrt{13}}{4})$. Comparing their coordinates and the curve segment sequence of P_3, we can find that they correspond to the CCSes $C^1_{1,3}, C^0_{1,3}, C_{3,9}, C^0_{3,4}$, respectively. Denote them as $P^1_{1,3}, P^0_{1,3}, P^{-1}_{3,9}, P^0_{3,4}$. Then compute the number of real roots of $f(\frac{3}{2},-\frac{5}{4},z)=0$, $f(\frac{3}{2},-\frac{11}{4},z)=0$, $f(2,-2-\frac{\sqrt{13}}{4},z)=0$ and $f(2,-2+\frac{\sqrt{13}}{4},z)=0$ in the interval $(-12,12)$. They are 3, 1, 0, 2, respectively. This is shown in the left part of Fig. 6. That means the numbers of the CSCSes originating from $V_{3,0}$ on the CCPs $SC^1_{1,3}, SC^0_{1,3}, SC_{3,9}, SC^0_{3,4}$ are 3, 1, 0, 2, respectively. There is no real points of the surface on the line lifted from the point $P^{-1}_{3,9}$ which is on the CCS $C_{3,9}$. So we need to delete the boundary point B_9, CCS $C_{3,9}$ from the topology information of $\mathcal{C}: g(x,y)=0$ and combine the cells C_6 and C_{11} as C_6. $V_{3,0}$ is one endpoint of the CSCSes $C^{1,0}_{1,3}, C^{1,1}_{1,3}, C^{1,2}_{1,3}, C^{0,0}_{1,3}, C^{0,0}_{3,4}, C^{0,1}_{3,4}$. As is shown in the right part of Fig. 6.

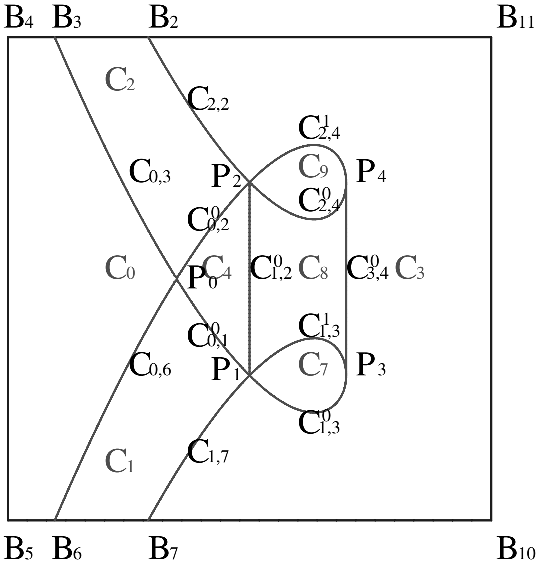


Fig. 7. Topology determination of the projection curve of a surface

The output is $\{V_{3,0}\{C^1_{1,3}(P^1_{1,3}(\frac{3}{2},-\frac{5}{4}))[C^{1,0}_{1,3},C^{1,1}_{1,3},C^{1,2}_{1,3}]$, $C^0_{1,3}(P^0_{1,3}(\frac{3}{2},-\frac{11}{4}))[C^{0,0}_{1,3}]$, $C^0_{3,4}(P^0_{3,4}(2,-2-\frac{\sqrt{13}}{4}))[C^{0,0}_{3,4},C^{0,1}_{3,4}]\}\}$.

After computing all boundary points and singularities of $\mathcal{C}$ by Algorithm 7, we can determine the position of all CSCSes of $\mathcal{S}$. And the projection curve of the surface is simplified as Fig. 7.

4.4 Compute the Surface Patches

Now, we need to compute the numbers of CSPs originating from each CSCS in the two cell bodies connected with the CCP which the CSCS lies on respectively. The following algorithm is for the purpose.

Algorithm 8. *The inputs are a real algebraic surface* $\mathcal{S} : f(x,y,z) = 0$*, the projection curve* $\mathcal{C} : g(x,y) = 0$ *of* $\mathcal{S}$*, a CCS* $C^k_{i,j}$ *(or* $C_{i,j}$*) on* $\mathcal{C}$ *and two cells beside it:* C_{k_1}, C_{k_2}*, a non-singular point on the CCS:* $P^k_{i,j}$ *(or* $P^{-1}_{i,j}$*)*(x_0, y_0) *and a sequence of CSCSes* $\{C^{k,m}_{i,j}, m = 0, \cdots, l_{i,j,k} - 1\}$ *on the CCP lifted from the CCS. The output are two ordered number lists of CSPs originating from each CSCS of the sequence in the two cell bodies from bottom to top respectively.*

1. Compute the tangent line of $\mathcal{C}$ at point $P^k_{i,j}$; compute the vertical line of the tangent line at $P^k_{i,j}$ and parameterize it as $(ta + x_0, tb + y_0)$.
2. Compute the real roots of the equation $g(ta + x_0, tb + y_0) = 0$. Record the root whose absolute value is the minimal among the nonzero real root(s). If the root does not exist, denote r as a constant, such as 1, else denote r as the absolute value of the root with minimal absolute value.

3. Isolate the real roots of $f(x_0, y_0, z) = 0$ by Algorithm 5, to obtain a sequence of rational number $\{z_0, z_1, \cdots, z_{l_{i,j,k}}\}$.
4. Compute the real roots of the equation $f(ta + x_0, tb + y_0, z_i) = 0$ for each $i = 0, 1, \cdots, l_{i,j,k}$. Record the root whose absolute value is the minimal among the real root(s). Denote the absolute value of the root as r_i. Let $R = \min\{r, r_0, r_1, \cdots, r_{l_{i,j,k}}\}/2$.
5. Compute the number of real roots of $f(Ra+x_0, Rb+y_0, z) = 0$ and $f(-Ra+x_0, -Rb+y_0, z) = 0$ in the interval $(z_m, z_{m+1})(m = 0, \cdots, l_{i,j,k} - 1)$ respectively. They are the numbers of CSPs originating from the CSCS $C_{i,j}^{k,m}$ in the cell bodies CC_{k_1}, CC_{k_2}.
6. Return the corresponding information.

Remark. If the CCS is an isolated singularity of $\mathcal{C}$, we need only to lift the point up, isolate the real roots of S on the line obtained in Algorithm 5, and find a line segment (its direction is parallel to XY-plane) which passes through the point as Algorithm 8. Then we can easily determine the number of CSPs originating from the points of S on the lifted line. If the CCS is a closed curve, Q is a point on the CCS, we can also easily compute the number of CSPs originating from the CSCSes on the CCP lifted from the CCS like Algorithm 8.

Theorem 4. *Algorithm 8 provides the correct output.*

Proof. The proof for this algorithm is same to the one for Algorithm 7 and is much easier. In this algorithm, we just replace the discs in Algorithm 7 with line segments. □

Example 7. Continuing from Example 6, we will compute the number of CSPs originating from the CSCSes on the CCP $SC_{0,3}$ as an example for this algorithm. The inputs are $g(x, y) = 0$, $f(x, y, z) = 0$, $C_{0,3}(C_0, C_2)[C_{0,3}^{-1,0}(V_{0,0}, V_{3,0}^0)$, $C_{0,3}^{-1,1}(V_{0,1}, V_{3,1}^0)$, $C_{0,3}^{-1,2}(V_{0,2}, V_{3,2}^0)]$, $P_{0,3}^{-1}(-2, -2 + 2 \cdot \sqrt{2})$. In step 1, we can obtain the line $(5 \cdot t - 2, 2 \cdot \sqrt{2} \cdot t - 2 + 2 \cdot \sqrt{2})$. Isolating $f(-2, -2 + 2 \cdot \sqrt{2}, z) = 0$, we can obtain $-12, -2, 5, 12$. We can find that R is a positive number more than $\frac{1}{20}$. In order to simplify our illustration, here we choose $\frac{1}{20}$ as R. Computing the number of real roots of $f(\frac{1}{20}a + x_0, \frac{1}{20}b + y_0, z) = 0$ and $f(-\frac{1}{20}a + x_0, -\frac{1}{20}b + y_0, z) = 0$ in the interval $(-12, -2), (-2, 5), (5, 12)$, respectively, we can obtain $\{1, 0, 1\}, \{1, 2, 1\}$. It means that there are 1,0,1(1,2,1) CSP(s) originating from the CSCSes $C_{0,3}^{-1,0}, C_{0,3}^{-1,1}, C_{0,3}^{-1,2}$ in the cell body lifted from $C_2(C_0)$, respectively. As is shown in Fig. 8. The output is $C_{0,3}(C_{0,3}^{-1,0}, C_{0,3}^{-1,1}, C_{0,3}^{-1,2})\{C_0[1, 2, 1], C_2[1, 0, 1]\}$.

Computing all the CSCSes of $\mathcal{S}$ with Algorithm 8, we can determine all the CSCSes and the number of CSPs originating from each CSCS in the two cell bodies beside it. Then we can form the CSPs of $\mathcal{S}$.

For each cell body lifted from a cell of $\mathcal{C}$, because the number of CSPs originating from all the CSCSes on each CCPs of the cell body is the same, we can determine each CSP in the cell body by pointing out its boundaries: CSCSes.

The following algorithm is to determine the CSPs of $\mathcal{S}$ by the topology information of $\mathcal{C}$ obtained by Algorithm 4.

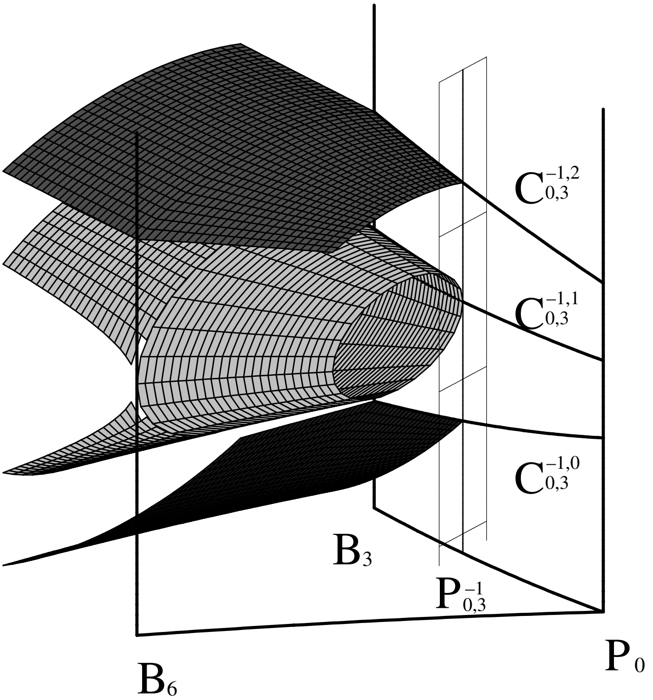


Fig. 8. Compute the CCS $C_{0,3}$ with Algorithm 8

Algorithm 9. *The inputs are $\mathcal{S}$: $f(x, y, z) = 0$ and the output of Algorithm 4. The output is the topological information of $\mathcal{S}$.*

1. Compute all the singularities and boundary points of $\mathcal{C}$ by Algorithm 7; determine all the CSCSes on each CCP lifted from the CCS of $\mathcal{C}$.
2. Compute the number of CSPs originating from each CSCS in two cell bodies beside it by Algorithm 8.
3. For each cell body lifted from a cell of $\mathcal{C}$, since the number of CSPs originating from the CSCSes on each CCP of the cell body is the same, we can determine each CSP by point out its boundaries–CSCSes.
4. Return the corresponding information of $\mathcal{S}$.

Example 8. Continuing from Example 7, we have determined all the CSPs of $\mathcal{S}$. The set of CSPs of $\mathcal{S}$ obtained by Algorithm 9 has the same topology as $\mathcal{S} : f(x, y, z) = 0$.

For the same example above, the outputs of the surface with Algorithm 9 are as follows. The figure of this surface is in Fig. 2. The figure of its real projection curve is as Fig. 7.

Points:
$\{\{B_2[V_{2,0}^0],\ B_3[V_{3,0}^0, V_{3,1}^0, V_{3,2}^0],\ B_4[V_{4,0}^0, V_{4,1}^0, V_{4,2}^0, V_{4,3}^0],\ B_5[V_{5,0}^0, V_{5,1}^0, V_{5,2}^0, V_{5,3}^0],$

$B_6[V_{6,0}^0, V_{6,1}^0, V_{6,2}^0]$, $B_7[V_{7,0}^0]$, $B_{10}[\,]$, $B_{11}[\,]\}$, $\{P_0[V_{0,0}, V_{0,1}, V_{0,2}]$, $P_1[V_{1,0}]$, $P_2[V_{2,0}]$, $P_3[V_{3,0}]$, $P_4[V_{4,0}]\}\}$.

For example, $B_3[V_{3,0}^0, V_{3,1}^0, V_{3,2}^0]$ means that there are three points of $\mathcal{S}$ on the line lifted from B_3. They are $V_{3,0}^0, V_{3,1}^0, V_{3,2}^0$, from bottom to top, respectively.

CSCSes:

$\{\{B_{2,3}[B_{2,3}^0(V_{2,0}^0, V_{3,0}^0), B_{2,3}^1(V_{2,0}^0, V_{3,2}^0)],$

$B_{3,4}[B_{3,4}^0(V_{3,0}^0, V_{4,0}^0), B_{3,4}^1(V_{3,1}^0, V_{4,1}^0), B_{3,4}^2(V_{3,1}^1, V_{4,2}^2), B_{3,4}^3(V_{3,2}^0, V_{4,3}^3)],$

$B_{4,5}[B_{4,5}^0(V_{4,0}^0, V_{5,0}^0), B_{4,5}^1(V_{4,1}^0, V_{5,1}^0), B_{4,5}^2(V_{4,2}^0, V_{5,2}^0), B_{4,5}^3(V_{4,3}^0, V_{5,3}^3)],$

$B_{5,6}[B_{5,6}^0(V_{5,0}^0, V_{6,0}^0), B_{5,6}^1(V_{5,1}^0, V_{6,1}^0), B_{5,6}^2(V_{5,2}^0, V_{6,1}^0), B_{5,6}^3(V_{5,3}^0, V_{6,2}^2)],$

$B_{6,7}[B_{6,7}^0(V_{6,0}^0, V_{7,0}^0), B_{6,7}^1(V_{6,2}^0, V_{7,0}^0)],$

$B_{7,10}[\,],$

$B_{10,11}[\,],$

$B_{11,2}[\,]\},$

$\{C_{0,3}[C_{0,3}^{-1,0}(V_{0,0}, V_{3,0}^0), C_{0,3}^{-1,1}(V_{0,1}, V_{3,1}^0), C_{0,3}^{-1,2}(V_{0,2}, V_{3,2}^0)],$

$C_{0,1}^0[C_{0,1}^{0,0}(V_{0,0}, V_{1,0}), C_{0,1}^{0,1}(V_{0,1}, V_{1,0}), C_{0,1}^{0,2}(V_{0,2}, V_{1,0})],$

$C_{0,6}[C_{0,6}^{-1,0}(V_{0,0}, V_{6,0}^0), C_{0,6}^{-1,1}(V_{0,1}, V_{6,1}^0), C_{0,6}^{-1,2}(V_{0,2}, V_{6,2}^0)],$

$C_{0,2}^0[C_{0,2}^{0,0}(V_{0,0}, V_{2,0}), C_{0,2}^{0,1}(V_{0,1}, V_{2,0}), C_{0,2}^{0,2}(V_{0,2}, V_{2,0})],$

$C_{1,7}[C_{1,7}^{-1,0}(V_{1,0}, V_{7,0}^0)],$

$C_{2,2}[C_{2,2}^{-1,0}(V_{2,0}, V_{2,0}^0)],$

$C_{1,2}^0[C_{1,2}^{0,0}(V_{1,0}, V_{2,0}), C_{1,2}^{0,1}(V_{1,0}, V_{2,0})],$

$C_{3,4}^0[C_{3,4}^{0,0}(V_{3,0}, V_{4,0}), C_{3,4}^{0,1}(V_{3,0}, V_{4,0})],$

$C_{1,3}^0[C_{1,3}^{0,0}(V_{1,0}, V_{3,0})],$

$C_{1,3}^1[C_{1,3}^{1,0}(V_{1,0}, V_{3,0}), C_{1,3}^{1,1}(V_{1,0}, V_{3,0}), C_{1,3}^{1,2}(V_{1,0}, V_{3,0})],$

$C_{2,4}^0[C_{2,4}^{0,0}(V_{2,0}, V_{4,0})],$

$C_{2,4}^1[C_{2,4}^{1,0}(V_{2,0}, V_{4,0}), C_{2,4}^{1,1}(V_{2,0}, V_{4,0}), C_{2,4}^{1,2}(V_{2,0}, V_{4,0})]\}\}.$

For example, $B_{2,3}[B_{2,3}^0(V_{2,0}^0, V_{3,0}^0), B_{2,3}^1(V_{2,0}^0, V_{3,2}^0)]$ means that there are two CSCSes on the CCP $SB_{2,3}$: $B_{2,3}^1$, whose endpoints are $V_{2,0}^0, V_{3,0}^0$ and $B_{2,3}^1$, whose endpoints are $V_{2,0}^0, V_{3,2}^0$.

CSPs:

$\{C_0(4)\{S_0^0[C_{0,3}^{-1,0}, B_{3,4}^0, B_{4,5}^0, B_{5,6}^0, C_{0,6}^{-1,0}]$, $S_0^1[C_{0,3}^{-1,1}, B_{3,4}^1, B_{4,5}^1, B_{5,6}^1, C_{0,6}^{-1,1}]$,
$S_0^2[C_{0,3}^{-1,1}, B_{3,4}^2, B_{4,5}^2, B_{5,6}^2, C_{0,6}^{-1,1}]$, $S_0^3[C_{0,3}^{-1,2}, B_{3,4}^3, B_{4,5}^3, B_{5,6}^3, C_{0,6}^{-1,2}]\}$,

$C_1(2)\{S_1^0[C_{0,6}^{-1,0}, B_{6,7}^0, C_{1,7}^{-1,0}, C_{0,1}^{0,0}]$, $S_1^1[C_{0,6}^{-1,2}, B_{6,7}^1, C_{1,7}^{-1,0}, C_{0,1}^{0,2}]\}$,

$C_2(2)\{S_2^0[C_{0,3}^{-1,0}, C_{0,2}^{0,0}, C_{2,2}^{-1,0}, B_{2,3}^0]$, $S_2^1[C_{0,3}^{-1,2}, C_{0,2}^{0,2}, C_{2,2}^{-1,0}, B_{2,3}^1]\}$,

$C_3(0)\{\,\}$,

$C_4(4)\{S_4^0[C_{0,2}^{0,0}, C_{0,1}^{0,0}, C_{1,2}^{0,0}]$, $S_4^1[C_{0,2}^{0,1}, C_{0,1}^{0,1}, C_{1,2}^{0,0}]$,
$S_4^2[C_{0,2}^{0,1}, C_{0,1}^{0,1}, C_{1,2}^{0,1}]$, $S_4^3[C_{0,2}^{0,2}, C_{0,1}^{0,2}, C_{1,2}^{0,1}]\}$,

$C_7(2)\{S_7^0[C_{1,3}^{0,0}, C_{1,3}^{1,0}]$, $S_7^1[C_{1,3}^{0,0}, C_{1,3}^{1,2}]\}$,

$C_8(4)\{S_8^0[C_{2,4}^{0,0}, C_{1,2}^{0,0}, C_{1,3}^{1,0}, C_{3,4}^{0,0}]$, $S_8^1[C_{2,4}^{0,1}, C_{1,2}^{0,0}, C_{1,3}^{1,1}, C_{3,4}^{0,0}]$,
$S_8^2[C_{2,4}^{0,1}, C_{1,2}^{0,1}, C_{1,3}^{1,1}, C_{3,4}^{0,1}]$, $S_8^3[C_{2,4}^{0,2}, C_{1,2}^{0,1}, C_{1,3}^{1,2}, C_{3,4}^{0,1}]\}$,

$C_9(2)\{S_9^0[C_{2,4}^{0,0}, C_{2,4}^{1,0}]$, $S_9^1[C_{2,4}^{0,2}, C_{2,4}^{1,0}]\}\}$.

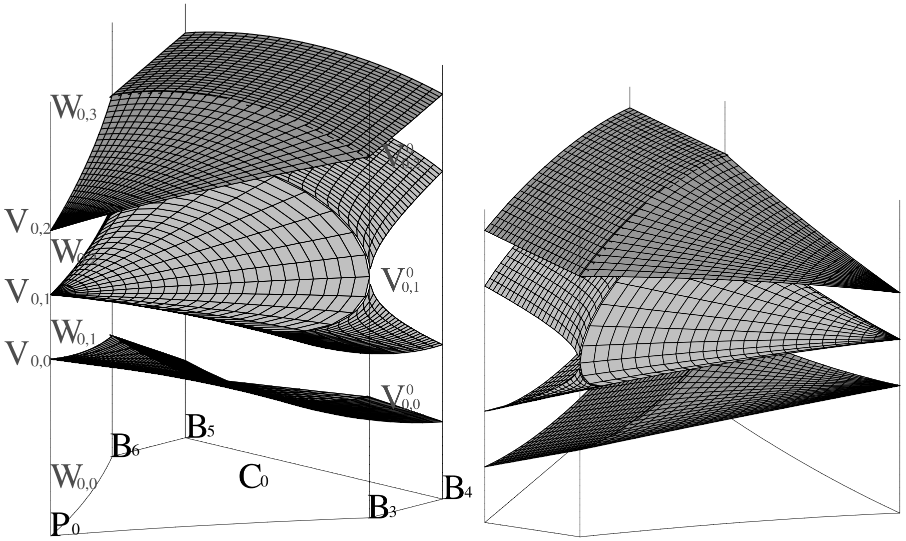


Fig. 9. The CSPs in the cell body lifted from C_0

For example, $C_0(4)\{S_0^0[C_{0,3}^{-1,0}, B_{3,4}^0, B_{4,5}^0, B_{5,6}^0, C_{0,6}^{-1,0}]$, $S_0^1[C_{0,3}^{-1,1}, B_{3,4}^1, B_{4,5}^1, B_{5,6}^1, C_{0,6}^{-1,1}]$, $S_0^2[C_{0,3}^{-1,1}, B_{3,4}^2, B_{4,5}^2, B_{5,6}^2, C_{0,6}^{-1,1}]$, $S_0^3[C_{0,3}^{-1,2}, B_{3,4}^3, B_{4,5}^3, B_{5,6}^3, C_{0,6}^{-1,2}]\}$ means that there are four CSPs, S_0^0, S_0^1, S_0^2, S_0^3 in the cell body CC_0 from bottom to up. $[C_{0,3}^{-1,0}, B_{3,4}^0$, $B_{4,5}^0, B_{5,6}^0, C_{0,6}^{-1,0}]$, $[C_{0,3}^{-1,1}, B_{3,4}^1, B_{4,5}^1, B_{5,6}^1, C_{0,6}^{-1,1}]$, $[C_{0,3}^{-1,1}, B_{3,4}^2, B_{4,5}^2, B_{5,6}^2, C_{0,6}^{-1,1}]$, $[C_{0,3}^{-1,2}, B_{3,4}^3, B_{4,5}^3, B_{5,6}^3, C_{0,6}^{-1,2}]$ are the boundaries of S_0^0, S_0^1, S_0^2, S_0^3, respectively. The CSPs in the cell body CC_0 are shown in Fig. 9.

5 Main Algorithm

By the discussion in the previous sections, we can present the main algorithm to determine the topology of an implicit algebraic surface.

Algorithm 10. *The input is an implicit algebraic surface* $\mathcal{S} : f(x, y, z) = 0$*. The output is a set of surface patches which have the same topology as the original surface* $\mathcal{S}$*.*

1. Compute the projection curve $\mathcal{C}$: $g(x, y) = 0$ of $\mathcal{S}$ by Algorithm 2.
2. Determine the topology of $\mathcal{C}$ by Algorithm 4.
3. Space curve and surface patch segmentation of $\mathcal{S}$ by Algorithm 9.
4. Return the corresponding topology information of $\mathcal{S}$.

Example 9. We will illustrate the algorithm with another example defined by

$$f(x, y, z) = f_1(x, y, z) \cdot f_2(x, y, z),$$

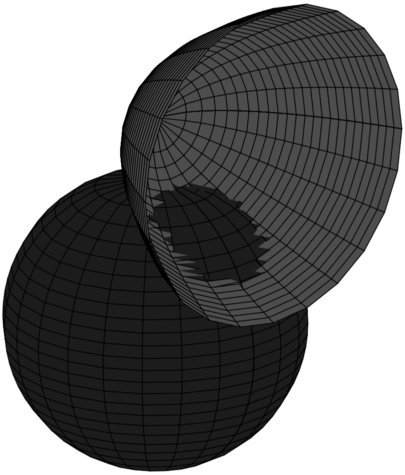


Fig. 10. A reducible surface

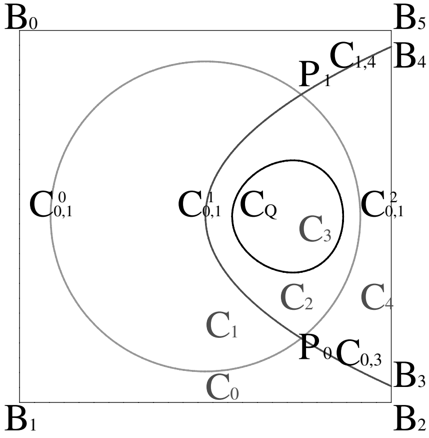


Fig. 11. Topology determination of the projection curve of a reducible surface

where $f_1(x, y, z) = y^2 + (z - 11)^2 - 5 \cdot x$, $f_2(x, y, z) = x^2 + y^2 + (z - 4)^2 - 25$. Its figure is in Fig. 10.

Since $f_1(x, y, z), f_2(x, y, z)$ are normal surfaces, we can derive the projection curve of $\mathcal{S}$ by Algorithm 2.

$$g(x, y) = \prod_{1 \le i \le j \le 2} T_{i,j}(x, y)$$

where $T_{1,1}(x, y) = 4 \cdot (y^2 - 5 \cdot x)$, $T_{1,2}(x, y) = 73 \cdot x^2 + 196 \cdot y^2 + 576 - 740 \cdot x + 10 \cdot x^3 + x^4$, $T_{2,2}(x, y) = 4 \cdot (x^2 + y^2 - 25)$.

With Algorithm 4, we can get the topological information of the projection curve as Fig. 11. We can derive the following information by Algorithm 9.
Points:
$\{\{B_0[\,], B_1[\,], B_2[\,], B_3[V_{3,0}^0], B_4[V_{4,0}^0], B_5[\,]\}, \{P_0[V_{0,0}, V_{0,1}], P_1[V_{1,0}, V_{1,1}]\}\}$.
CSCSes:
$\{\{B_{0,1}[\,], B_{1,2}[\,], B_{2,3}[\,], B_{3,4}[B_{3,4}^0(V_{3,0}^0, V_{4,0}^0), B_{3,4}^1(V_{3,0}^0, V_{4,0}^0)], B_{4,5}[\,], B_{5,0}[\,]\}$,
$\{C_{0,1}^0[C_{0,1}^{0,0}(V_{0,0}, V_{1,0})]$, $C_{0,1}^1[C_{0,1}^{1,0}(V_{0,0}, V_{1,0}), C_{0,1}^{0,1}(V_{0,0}, V_{1,0}), C_{0,1}^{0,2}(V_{0,1}, V_{1,1})]$,
$C_{0,1}^2[C_{0,1}^{2,0}(V_{0,0}, V_{1,0}), C_{0,1}^{2,1}(V_{0,1}, V_{1,1}), C_{0,1}^{2,2}(V_{0,1}, V_{1,1})]$, $C_{1,4}[C_{1,4}^{-1,0}(V_{1,1}, V_{4,0}^0)]$,
$C_{0,3}[C_{0,3}^{-1,0}(V_{0,1}, V_{3,0}^0)], C_Q[C_Q^0, C_Q^1, C_Q^2]\}\}$.
CSPs:
$\{C_0(0)\{\,\}$,
$C_1(2)\{S_1^0[C_{0,1}^{0,0}, C_{0,1}^{1,0}], S_1^0[C_{0,1}^{0,0}, C_{0,1}^{1,1}]\}$,
$C_2(4)\{S_2^0[[C_{0,1}^{1,0}, C_{0,1}^{2,0}], [C_Q^0]], S_2^1[[C_{0,1}^{1,1}, C_{0,1}^{2,0}], [C_Q^1]]$,
$S_2^2[[C_{0,1}^{1,2}, C_{0,1}^{2,1}], [C_Q^1]]$, $S_2^3[[C_{0,1}^{1,2}, C_{0,1}^{2,2}], [C_Q^2]]\}$,

$$C_3(4)\{S_3^0[C_Q^0], S_3^1[C_Q^1], S_3^2[C_Q^1], S_3^3[C_Q^2]\},$$
$$C_4(2)\{S_4^0[C_{0,1}^{2,0}, C_{0,3}^{-1,0}, B_{3,4}^0, C_{1,4}^{-1,0}],\ S_4^1[C_{0,1}^{2,1}, C_{0,3}^{-1,0}, B_{3,4}^1, C_{1,4}^{-1,0}]\}\}.$$

According to our experiments, the topology determination of the projection curve is the most time-consuming phase of the algorithm.

6 Conclusion

In this paper, we present an algorithm, which can be used to give a representation for the topology of an implicit algebraic surface $f(x, y, z) = 0$. We give a curvilinear wireframe of the surface and the surface patches of the surface determined by the curvilinear wireframe, which present the same topology as the surface. Most of the surface patches are curvilinear polygons. If needed, we can easily modify our algorithm to give a polyhedron which has the same topology as the surface.

The algorithm mainly involves computation of resultants, determination of the topology of plane curves, computation of singularities of surfaces and curves, isolating real roots of univariate equations. Many aspect of the algorithm could be further improved. This will be done in our later work.

Acknowledgements. Partially supported by a National Key Basic Research Project of China and by a USA NSF grant CCR-0201253.

References

1. Arnborg, S. and Feng, H., Algebraic decomposition of regular curves. *J. Symbolic Comput.*, 5(1,2)(1988): 131-140.
2. Arnon, D. S., Collins, G. and McCallum S., Cylindrical algebraic decomposition I: the basic algorithm. *Quantifier Elimination and Cylindrical Algebraic Decomposition, Edited by B. Buchberger and G. E. Collins*, Springer: 136-151.
3. Arnon, D. S. and McCallum, S., A polynomial-time algorithm for the topological type of a real algebraic curve, *J. Symbolic Comput.*, 5(1,2)(1988): 213-236.
4. Bajaj, C., Hoffmann, C. M., Lynch, R. E. and Hopcroft, J. E. H., Tracing surface intersection, *Computer Aided Geometric Design*, (1988)5: 285-307.
5. Bajaj, C. and Xu, G. L., Spline approximations of real algebraic surfaces, *J. Symbolic Comput.*, (1997)23: 315-333.
6. Basu, S., Pollack, R. and Roy, M.-F., Algorithms in real algebraic geometry. *Algorithms and Computat. in Mathematics*, (2003)10. Springer-Verlag.
7. Feng, H., Decomposition and computation of the topoplogy of plane real algebraic curves, PhD Thesis. (1992) The Royal Institute of Technology, Stockholm, Sweden.
8. Fortuna, E., Gianni, P., Parenti, P. and Traverso, Algorithms to compute the topology of orientable real algebraic surfaces, *J. Symbolic Comput.*, 36(2003)343-364.
9. Gao, X. S. and Li, M., Rational quadratic approximation to real algebraic curves, *Computer Aided Geometric Design*, (2004)21: 805-828.
10. Gatellier G., Labrouzy A., Mourrain. B. and Técourt. J. P., Computing the topology of three dimensional algebraic curves. *Computational Methods for Algebraic Spline Surfaces*, (2004),Springer-Verlag: 27-44.

11. Geismann, N., Hemmer, M. and Schömer, E., Computing a 3-dimensional cell in an arrangement of quadrics: exactly and actually. *Symposium on Computational Geometry*, 2001: 264-273.
12. Gianni, P. and Traverso, C., Shape determination for real curves and surfaces. *Ann. Univ. Ferrara Sez. VII(N.S.)*, (1983)29, 87-109.
13. Gonzalez-Vega, L. and El Kahoui, M., An improve upper complexity bound for the topology computation of a real algebraic plane curve. *J. Complexity*, (1996)12: 527-544.
14. Gonzalez-Vega, L. and Necula, I., Efficient topology determination of implicitly defined algebraic place curves, *Computer Aided Geometric Design*, 19(2002): 719-743.
15. Hart. J. C., Morse theory for implicit surface modeling. *Mathematical Visualization*, H-C Hege and K. Polthier, Eds., Springer-verlag, Oct. 1998: 257-268.
16. Hong, H., An efficient method for analyzing the topology of plane real algebraic curves. *Math. Comput. Simulation*, (1996) 42(4-6): 571-582.
17. Johnson, J. R., Algorithms for polynomial real root isolation. Ph.d. Thesis. The Ohio state University. 1991.
18. Keyser, J., Culver, T. and Krishnan, S., Efficient and exact manipulation of algebraic points and curve. *Computer Aided Design*, (2000) 32(11): 649-662.
19. Massey, W. S., A basic course in algebraic topology. Springer-verlag, 1991.
20. Ni, X. L., Garland, M. and Hart, J. C., Fair morse functions for extracting topological structure of a surface mesh. *Proc. SIGGRAPH 2004*.
21. Sakkalis, T., The topological configuration of a real algebraic curve. *Bull. Australian Math. Soc.*, (1991) 43(1): 37-50.
22. Stander, B. T. and Hart, J. C., Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Proc. SIGGRAPH 97* Aug. 1997: 279-286.
23. Walker, R. J., Algebraic curves, *Springer-Verlag*, 1978.
24. Wu, W. T., Mathematics Mechanization, *Science Press/Kluwer*, Beijing, 2000.
25. Yang, L., Zhang, J. Z. and Hou X. R., Nonlinear Algebraic Equation System and Automated Theorem Proving, *Shanghai Scientific and Technological Education Publishing House*, Shanghai, 1996.
26. Zhang, S. G., Liu, D. Z. and Feng, G. C., Computer Mathematics: an introduction(in Chinese), Jilin University Press, 1997.

Level Sets of Functions and Symmetry Sets of Surface Sections

André Diatta[1], Peter Giblin[1], Brendan Guilfoyle[2], and Wilhelm Klingenberg[3]

[1] University of Liverpool, UK
[2] Institute of Technology, Tralee, Ireland
[3] University of Durham, UK
{adiatta, pjgiblin}@liv.ac.uk
brendan.guilfoyle@ittralee.ie, wilhelm.klingenberg@durham.ac.uk

Abstract. We prove that the level sets of a real C^s function of two variables near a non-degenerate critical point are of class $C^{[s/2]}$ and apply this to the study of planar sections of surfaces close to the singular section by the tangent plane at an elliptic or hyperbolic point, and in particular at an umbilic point. We go on to use the results to study symmetry sets of the planar sections. We also analyse one of the cases coming from a degenerate critical point, corresponding to an elliptic cusp of Gauss on a surface, where the differentiability is reduced to $C^{[s/4]}$. However in all our applications we assume C^∞ smoothness.

1 Introduction

The medial axis or skeleton of a closed plane curve γ encode a great deal of information about the shape of the region enclosed by the curve. This information has been exploited in many ways[1]. The medial axis can be defined, for a smooth curve γ, as the closure of the locus of centres of 'bitangent' circles—tangent to γ in two places—and whose radius equals the minimum distance from the centre to γ. The symmetry set is the closure of the locus of centres of all bitangent circles. It thus has a richer structure which underlies the transitions of the medial axis. See for example [6].

The local structure of the symmetry set of a generic plane curve is of four kinds: smooth branches, endpoints, cusps and triple crossings. All of these can be seen in the example of Figure 2. (For medial axes there are only smooth branches, endpoints and Y-junctions.) Symmetry sets and medial axes of curves which vary in a generic 1-parameter family γ_k are well understood provided the curves of the family remain nonsingular; a complete local classification was given in [2]. In this paper, we consider instead the family of plane sections γ_k of a smooth surface M in 3-space, where the plane moves parallel to itself and

[1] Typing 'medial axis' into an internet search engine produces hundreds of references, to character recognition, Voronoi diagrams, interrogation, reconstruction, modification and design of shape, etc.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 147–160, 2005.

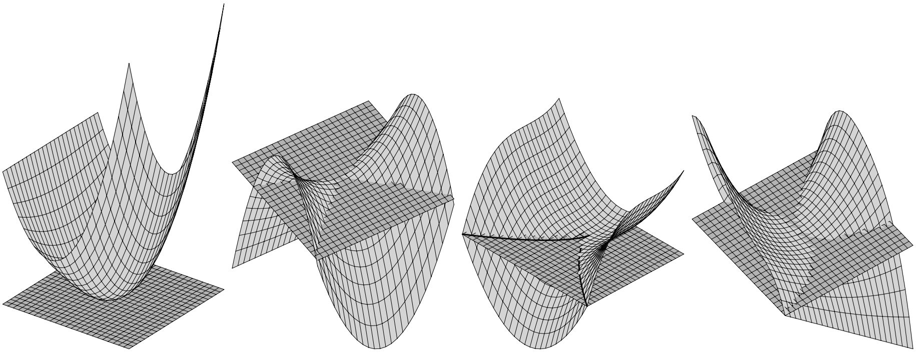


Fig. 1. Surfaces with the tangent planes at elliptic, hyperbolic, parabolic and hyperbolic cusp of Gauss points. The intersections are a point, two smooth transverse curves, a cusped curve and two tangential curves respectively. Figure produced with MAPLE

becomes the tangent plane to M at a point $\mathbf{p}$ for say $k = 0$. Then the section γ_0 will be singular at the tangency point $\mathbf{p}$. For $\mathbf{p}$ an elliptic point γ_0 is, locally, a single point. For $\mathbf{p}$ a hyperbolic point γ_0 is locally two smooth transversally intersecting branches, while at an ordinary parabolic point γ_0 has an ordinary cusp at $\mathbf{p}$. See Figure 1, which also shows a 'hyperbolic cusp of Gauss' where the intersection is two smooth tangential branches. See §4.2. In this situation, the results of [2] are invalid and we need to use other techniques to find out how the symmetry sets (and medial axes) behave in the family of curves.

The motivation for this work comes from the study of isophote curves of a 2D image, which are sections of an intensity surface in 3-space. Results on the pattern of vertices (extrema of curvature) and inflexions (zeros of curvature) of the sections γ_k have been given in [4] while in [5] there are results on the patterns of cusps and triple crossings of the symmetry set. Let the surface M be given locally by an equation $z = f(x, y)$. Then the sections γ_k are by their nature given by *equations* $f(x, y) = k$, and not by *parametrizations* $\gamma(t) = (X(t), Y(t))$. Now plotting symmetry sets of parametrized curves is reasonably straightforward (see §4). But curves given by equations are a different matter altogether: there is not even any foolproof way to determine how many real components a curve $f(x, y) = k$ has. Furthermore it may not be feasible to find an exact parametrization of one component of a curve $f(x, y) = k$.

In this paper we give a result on parametrizing plane sections of surfaces and use it to approximate such a section to any desired degree of accuracy. We apply this to the plotting of symmetry sets and medial axes of the surface sections γ_k. In §2 we give the main theoretical result and its proof (Propositions 1-4), in §3 we give the way in which this is implemented in practice, and in §4 we apply the method to symmetry sets, concentrating in §4.1 on the case of umbilic points (elliptic points where the principal curvatures are equal). In §4.2 we consider the case of an 'elliptic cusp of Gauss', for which the curves γ_k are simple closed curves when $k > 0$.

2 Level Sets of Functions

The intuitive idea here that we 'blow up' the origin to turn the surface $z = f(x, y)$ into one with nonsingular sections. We shall take the surface M in Monge form, that is with

$$\begin{aligned} f(x,y) = {} & \tfrac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2) + b_0x^3 + b_1x^2y + b_2xy^2 + b_3y^3 \\ & + c_0x^4 + c_1x^3y + c_2x^2y^2 + c_3xy^3 + c_4y^4 \\ & + d_0x^5 + d_1x^4y + d_2x^3y^2 + d_3x^2y^3 + d_4xy^4 + d_5y^5 + \text{ h.o.t.} \end{aligned} \tag{1}$$

For simplicity, consider the case where κ_1 and κ_2 are > 0 so that, locally to the origin, $z \geq 0$ at points of M. Let us write $x = tX,\ y = tY, z = t^2$; after cancellation of t^2 the equation $z = f(x, y)$ then becomes

$$1 = \tfrac{1}{2}(\kappa_1 X^2 + \kappa_2 Y^2) + t(b_0X^3 + \ldots) + t^2(c_0X^4 + \ldots) + \ldots,$$

which is a smooth surface in (X, Y, t)-space whose sections $t =$ constant $\neq 0$ are scaled versions of the sections of M. We proceed to give the formal details of this procedure in a more general setting and in the next section show how in practice we have used it.

We prove that the level sets of a real function of two variables near a nondegenerate critical point are of class $\mathrm{C}^{[s/2]-1}$ if the function is of class C^s. To this end we consider $f \in C^s(D)$, $D \subset \mathbb{R}^2$ the open unit disc, $f(0) = df(0) = 0$, and set $q = \nabla df(0)$ to be the hessian of f at the origin. We assume that q is either positive definite or indefinite. See Proposition 4 below for the case of a degenerate critical point.

Proposition 1. *Assume that q is positive definite and let $r_0 : S^1 \to \mathbb{R}^+$, where $S^1 \subset \mathbb{R}^2$ is the unit circle, be defined by $r_0(\theta) = [q(\cos\theta, \sin\theta)]^{-1/2}$, $\theta \in S^1$ corresponding to $e^{i\theta}$.*

Then there exists a function $r : [0, \epsilon) \times S^1 \to \mathbb{R}^+$ for some $\epsilon > 0$ with

(a) *$r \in C^{[s/2]}([0, \epsilon) \times S^1)$,*
(b) *$r(0, \theta) = r_0(\theta)$ on S^1,*
(c) *$f(t\, r(t, \theta) \cos\theta, t\, r(t, \theta) \sin\theta) = t^2$ on $[0, \epsilon) \times S^1$.*

Remark: The equation (c) implies that for fixed $t \in [0, \epsilon)$, the parametrized curve $\theta \mapsto (r(t, \theta) \cos\theta, r(t, \theta) \sin\theta)$ is a rescaled level curve of the function f.

Proof. For $t > 0$ consider the curve

$$C_t = \{f(tx, ty) = t^2,\ z = t^2\} \subset \mathbb{R}^3_{xyz}$$

Consider the surface defined by

$$Q = \{f(\sqrt{|z|}\, x, \sqrt{|z|}\, y) = |z| \text{ for } \ z \neq 0 \quad \text{and} \quad q(x, y) = 1 \text{ for } z = 0\}.$$

We claim that Q is a regular surface of class $C^{[s/2]}$ near $\{z = 0\} \cap Q$. Given this, we have $C_t = Q \cap \{z = t^2\}$, and since the intersection is transversal (as is seen by computing grad $f(\sqrt{|z|}\ x, \sqrt{|z|}\ y) - |z|\ )$, we conclude that $C_t \in C^{[s/2]}$ with uniform bounds. Using the implicit function theorem we derive the existence of r and hence (a) to (c). Note that now C_t is parametrized by

$$\{(t\, r(t,\theta)\cos\theta, t\, r(t,\theta)\sin\theta, t^2) \text{ for } \theta \in S^1\}. \qquad \square$$

The claim above follows from the following proposition:

Proposition 2. *Let $h(x)$ be a real function of one real variable, $h = O(|x|^2)$ and $h \in C^l$. Then $g(x,y) = h(\sqrt{|y|}\ x)$ is of class $C^{[l/2]}$ and $g = O(|y|)$.*

Proof. The proof of this proposition follows from Taylor's formula. □

Finally, we have the analogous result for q indefinite, which is proved in a similar way:

Proposition 3. *Assume that q is indefinite and of the form $q(x,y) = \kappa_1 x^2 - \kappa_2 y^2$ with $\kappa_1 \geq \kappa_2 > 0$. Let $S^+ \subset S^1$ be defined by $q|_{S^+} > 0$. For any $S_0 \subset S^+$, $S_0 \neq S^+$ there exists $\epsilon > 0$ and $r : [0,\epsilon) \times S_0 \to \mathbb{R}^+$ with*

(a) $r \in C^{[s/2]}([0,\epsilon) \times S_0)$,
(b) $f(t\frac{1}{\sqrt{\kappa_1}} r(t,\theta)\cosh\theta, t\frac{1}{\sqrt{\kappa_2}} r(\theta,t)\sinh\theta) = t^2$ *on* $S_0 \times [0,\epsilon)$
(c) $r(\theta,0) = 1$ *on* S_0. □

We have a result similar to Proposition 1 in the case of an elliptic cusp of Gauss. In this case the quadratic terms of f are degenerate, equal to x^2 say, so that the surface $z = f(x,y)$ has a parabolic point at the origin. By making x divide the cubic terms we can still have a closed curve of intersection $f(x,y) = k > 0$ provided the terms in x^2, xy^2, y^4 give a positive definite quadratic form in x and y^2. This is called an elliptic cusp of Gauss. The contact of the surface $z = f(x,y)$ with its tangent plane at the origin is of type A_3 in the notation of Arnold. See for example [1, 7] for many geometrical properties of these points.

Proposition 4. *Assume that $f \in C^s(D)$ and let the point $0 \in M$ be an elliptic cusp of Gauss and $\kappa_1 > 0$. Set $q(x,y) = \frac{1}{2}\kappa_1 x^2 + b_2 xy^2 + c_4 y^4$, with $b_2^2 < 2\kappa_1 c_4$. Let $r_0 : S^1 \to \mathbb{R}^+$ be defined by $r_0(\theta) = q(\cos\theta, \sin\theta)^{-1/4}$. Then there exists a function $r : [0,\epsilon) \times S^1 \to \mathbb{R}^+$ for some $\epsilon > 0$ with*

(a) $r \in C^{[s/4]}([0,\epsilon) \times S^1)$,
(b) $r(0,\theta) = r_0(\theta)$ *on* S^1,
(c) $f((t\, r(t,\theta))^2 \cos\theta, t\, r(t,\theta)\sin\theta) = t^4$ *on* $[0,\epsilon) \times S^1$.

Proof. The rescaling here is nonhomogeneous and given as follows. Let

$$C_t = \{f(t^2 x, ty) = t^4,\ z = t^4\} \subset \mathbb{R}^3_{xyz}.$$

Consider the surface defined by

$$Q = \{f(|z|^{\frac{1}{2}}\ x, |z|^{\frac{1}{4}}\ y) = |z| \text{ for } \ z \neq 0 \quad \text{and} \quad q(x,y) = 1 \text{ for } z = 0\}.$$

Now Q is a regular surface of class $C^{[s/4]}$.This follows from a result analogous to Proposition 2. □

Remark: For a fixed value of t, we are parametrizing the level set $f(x,y)=t^4$ not in 'polar coordinates' where each ray from the origin intersects the curve in one point, but by means of parabolas of the form $y^2=kx$ for constants k.

3 Finding the Parametrization in Practice

All functions and surfaces from now on will be assumed smooth of class C^∞, that is $s=\infty$ in the Propositions of §2.

We seek to approximate the sections $f(x,y)=k$ of the surface M up to a suitable order. Let M be given in Monge form (1). As in §2 the two cases (i) elliptic: $\kappa_1>0, \kappa_2>0$ and (ii) hyperbolic: $\kappa_1>0, \kappa_2<0$ are different: in the former case the sections $f(x,y)=k$ are, locally to the origin, closed curves for small $k>0$ whereas in the latter case they are open curves extending to infinity for both signs of k.

In this article our applications will concentrate on the cases where the intersection is a closed curve, but we give some details of other cases in this section.

- For a hyperbolic point we seek to parametrize the section by

$$\begin{aligned} &x=t\,r(t,\theta)\cosh\theta,\ y=t\,r(t,\theta)\sinh\theta,\ z=t^2, \quad \text{or}\\ &x=t\,r(t,\theta)\sinh\theta,\ y=t\,r(t,\theta)\cosh\theta,\ z=-t^2, \end{aligned} \qquad (2)$$

  for a suitable function r.

- For an elliptic point we seek to parametrize the section by

$$x=t\,r(t,\theta)\cos\theta,\quad y=t\,r(t,\theta)\sin\theta,\quad z=t^2,\ t>0,\ 0\le\theta<2\pi. \qquad (3)$$

- For an 'elliptic cusp of Gauss' we seek to parametrize the section by

$$x=(t\,r(t,\theta))^2\cos\theta,\quad y=t\,r(t,\theta)\sin\theta,\quad z=t^4,\ t>0,\ 0\le\theta<2\pi. \qquad (4)$$

We will write

$$r(t,\theta)=r_0+r_1t+r_2t^2+\dots,$$

where the r_i are functions of θ only. In the elliptic case for small values of $k>0$ the section $z=k$ is, locally to the origin, a closed curve and the r_i will be periodic and hence functions of $\cos\theta$ and $\sin\theta$.

We therefore need to express the r_i in terms of the Monge form coefficients κ_i, b_i, c_i and so on. This is done by comparison of series, and enables us to approximate the surface $z=f(x,y)$, and the sections $z=$ constant up to any desired accuracy.

Consider the terms of degree i in the Taylor expansion of f: this is a homogeneous polynomial of degree i in x and y. Denote by p_i the function of θ obtained from this homogeneous polynomial as follows:

hyperbolic case: replace x by $\cosh\theta$ and y by $\sinh\theta$ when $z > 0$ and x by $\sinh\theta$ and y by $\cosh\theta$ when $z < 0$;
elliptic case: replace x by $\cos\theta$ and y by $\sin\theta$. Thus
$p_2 = \frac{1}{2}(\kappa_1 c^2 + \kappa_2 s^2)$;
$p_3 = b_0c^3 + b_1c^2s + b_2cs^2 + b_3s^3$;
$p_4 = c_0c^4 + c_1c^3s + c_2c^2s^2 + c_3cs^3 + c_4c^4$;
$p_5 = d_0c^5 + d_1c^4s + d_2c^3s^2 + d_3c^2s^3 + d_4cs^4 + d_5s^5$;
and so on,

where
hyperbolic case: $c = \cosh\theta$, $s = \sinh\theta$ if $z > 0$ and $c = \sinh\theta$, $s = \cosh\theta$ if $z < 0$,
elliptic case: $c = \cos\theta$ and $s = \sin\theta$.

Hyperbolic Case. By substitution in the Monge form (1), we obtain in succession the following formulas.
$r_0 = \frac{1}{\sqrt{p_2}}$ when $z > 0$ and $r_0 = \frac{1}{\sqrt{-p_2}}$ when $z < 0$;
$r_1 = -\frac{1}{2r_0p_2} r_0^2 p_3$;
$r_2 = -\frac{1}{2r_0p_2}(r_0^4p_4 + 3r_0^2r_1p_3 + r_1^2p_2)$;
$r_3 = -\frac{1}{2r_0p_2}(2r_1r_2p_2 + r_0^5p_5 + 3r_0^2r_2p_3 + 3r_0r_1^2p_3 + 4r_0^3r_1p_4)$;
$r_4 = -\frac{1}{2r_0p_2}(5r_0^4r_1p_5 + 4r_0^3r_2p_4 + 2r_1r_3p_2 + 6r_0^2r_1^2p_4 + r_2^2p_2 + 6r_0r_1r_2p_3$
$\quad + 3r_0^2r_3p_3 + r_1^3p_3 + r_0^6p_6)$;
$r_5 = -\frac{1}{2r_0p_2}(3r_0r_2^2p_3 + 3r_1^2r_2p_3 + 6r_0r_1r_3p_3 + 3r_0^2r_4p_3 + 12r_0^2r_1r_2p_4$
$\quad + 4r_0^3r_3p_4 + 4r_0r_1^3p_4 + 6r_0^5r_1p_6 + 5r_0^4r_2p_5 + 10r_0^3r_1^2p_5 + 2r_1r_4p_2 + 2r_2r_3p_2)$;
etc.

Taking the quadratic terms of the surface to be $x^2 - \alpha^2y^2$ where $\alpha > 0$, we have

$$r_0^2 = \frac{1}{(1-\alpha^2)\cosh^2\theta + \alpha^2}\ (z > 0); \qquad r_0^2 = \frac{1}{1-(1-\alpha^2)\cosh^2\theta}\ (z < 0).$$

When $z > 0$, the expression for r_0^2 is > 0 for all θ when $0 < \alpha \leq 1$, but for $\alpha > 1$ we need $\cosh^2\theta < \frac{\alpha^2}{\alpha^2-1}$, that is $1 \leq \cosh\theta < \frac{\alpha}{\sqrt{\alpha^2-1}}$.

When $z < 0$, the expression for r_0^2 is > 0 for all θ when $\alpha \geq 1$ but if $0 < \alpha < 1$ we need $1 \leq \cosh\theta < \frac{1}{\sqrt{1-\alpha^2}}$.

Elliptic Case. This is much simpler because, as above, we can expect a global parametrization of the closed curves $f(x, y) = k$, locally to the origin, for $k > 0$. The formulae for p_i are given above and those for r_i are exactly the same as for the hyperbolic case, except that, for $z > 0$ only,

$$r_0^2 = \frac{2}{\kappa_1\cos^2\theta + \kappa_2\sin^2\theta}.$$

Note that we are assuming $\kappa_1 > 0, \kappa_2 > 0$ since the surface is assumed to be above the plane $z = 0$ close to the origin. Thus r_0 is always real.

When the origin is an umbilic point, then $\kappa_1 = \kappa_2 = \kappa$, say, and $r_0^2 = \frac{2}{\kappa}$, a constant. Indeed, we always scale the variables so that the quadratic terms of f

are x^2+y^2 and then r_0^2 has the constant value 1 and we can take $r_0 = 1$ without loss of generality.

Elliptic Cusp of Gauss Case. The relevant parametrization here is (4), and we apply Proposition 4. For a cusp of Gauss, after renaming axes if necessary, we have

$$\kappa_1 \neq 0,\ \kappa_2 = 0,\ b_3 = 0 \text{ and} \begin{cases} b_2^2 < 2\kappa_1 c_4 \text{ elliptic cusp,} \\ b_2^2 > 2\kappa_1 c_4 \text{ hyperbolic cusp.} \end{cases} \tag{5}$$

Consider the elliptic case. The expansion of the function r in Proposition 4 is obtained by a 'weighted' version of the method used in the previous cases. This time let $p_i,\ i \geq 4$, be the result of substituting $x = \cos\theta,\ y = \sin\theta$ in the terms of f of weighted degree i, where x has weight 2 and y has weight 1. Thus writing $c = \cos\theta,\ s = \sin\theta$ we have
$p_4 = \frac{1}{2}\kappa_1 c^2 + b_2 cs^2 + c_4 s^4 = q(c,s)$ (see Proposition 4)
$p_5 = b_1 c^2 s + c_3 cs^3 + d_5 s^5$
and so on. Writing $r = r_0 + r_1 t + r_2 t^2 + r_3 t^3 + \ldots$ as before, where each r_i is a function of θ only, we find on substitution
$r_0^4 = 1/p_4$,
$4p_4 r_1 + r_0^2 p_5 = 0$, from which we solve for r_1,
$4p_4 r_0 r_2 + 6p_4 r_1^2 + 5p_5 r_0^2 r_1 + p_6 r_0^4 = 0$, from which we solve for r_2,
and so on. The assumption of an *elliptic* cusp of Gauss guarantees that p_4 is nonzero for all values of θ.

The same ideas can be used in principle to parametrize the sections near a hyperbolic cusp of Gauss. It is not difficult to show that in this case (where $b_1^2 > 2\kappa_1 c_4$) the expression p_4 above vanishes for exactly four values of θ in the range $0 < \theta < 2\pi$. In increasing order these are of the form $0 < \theta_1 < \theta_2 < \theta_3 = 2\pi - \theta_2 < \theta_4 = 2\pi - \theta_1 < 2\pi$. In the ranges from θ_1 to θ_2 and from θ_3 to θ_4 the expression p_4 is negative and in the other two ranges it is positive. Then we can use
$x = t^2 r^2 \cos\theta,\ y = tr\sin\theta,\ z = -t^4$ in the first two ranges and
$x = t^2 r^2 \cos\theta,\ y = tr\sin\theta,\ z = t^4$ in the other two ranges,
determining r exactly as before, except that $r_0^4 = -1/p_4$ when $p_4 < 0$. Needless to say this case is much more delicate than the elliptic case since the branches are not closed and the relationship between the value of t and the closeness of fit obtained by taking say four terms of the approximation to r will be difficult to determine. An example is shown in Figure 4.

4 Applications to Pre-symmetry Sets and Symmetry Sets

Let γ be a smooth *parametrized* plane curve. We seek first the 'pre-symmetry set' which is the set of parameter pairs (u_1, u_2) such that there exists a circle tangent to γ at $\gamma(u_1), \gamma(u_2)$. The pre-symmetry set is symmetric about the 'diagonal'

$u_1 = u_2$. A convenient way to find these pairs is to look for solutions of the equation given by a scalar product

$$(\gamma_1 - \gamma_2) \cdot (\mathbf{T}_1 - \mathbf{T}_2) = 0, \tag{6}$$

where $\gamma_i = \gamma(u_i)$ and $\mathbf{T}_i$ is the unit oriented tangent to γ at γ_i. These solutions are precisely the pairs required, together with any pairs where $\mathbf{T}_1 = \mathbf{T}_2$: parallel oriented tangent pairs. For a closed convex curve the only such pairs arise from $u_1 = u_2$ and these are easily identified as the diagonal. When a closed curve has inflexions other pairs with $\mathbf{T}_1 = \mathbf{T}_2$ will be present. We need to beware of these when interpreting the pre-symmetry set. The pre-symmetry set contains much information about the symmetry set. Crossings of the diagonal correspond to endpoints of the symmetry set and horizontal or vertical tangents correspond to cusps.

Secondly we want to plot the centres of the bitangent circles identified above. There are many ways to do this but here is a convenient one, couched in the language of complex numbers. Let c be the centre of the circle. Then, for some angle θ,

$$(\gamma_2 - c) = e^{i\theta}(\gamma_1 - c), \quad \mathbf{T}_2 = e^{i\theta}\mathbf{T}_1.$$

Then, as complex numbers, we have by dividing these equations $c = (\gamma_2\mathbf{T}_1 - \gamma_1\mathbf{T}_2)/(\mathbf{T}_1 - \mathbf{T}_2)$. Note that if $\mathbf{T}_1 = \mathbf{T}_2$, or, in practice, if these are sufficiently close together, and $\gamma_1 \neq \gamma_2$, then c is very far away and it will not appear on the diagram of the symmetry set.

The above has been implemented in the Liverpool Surfaces Modelling Package (LSMP, [8]), also known as SingSurf. Note that it is necessary for γ to be *parametrized*, which is the motivation for the theoretical results given in §2. We give below examples of umbilic points and elliptic cusps of Gauss (the latter obtained for the present by a different method). We shall give examples of hyperbolic points elsewhere. See [4, 5] for other methods and theoretical results related to the examples below.

4.1 Umbilic Points

Elliptic points for which the principal curvatures κ_i as in (1) are unequal make rather uninteresting examples since the intersection $f(x, y) = k$ is a curve which is very nearly an ellipse, having exactly four vertices. The symmetry set has two smooth branches and the medial axis has one smooth branch, as for an ellipse. We concentrate here on the case of an *umbilic* point, where $\kappa_1 = \kappa_2$, and without loss of generality we can take these both equal to 1 by scaling the surface. Thus the equation of the surface has the form

$$z = x^2 + y^2 + b_0x^3 + b_1x^2y + b_2xy^2 + b_3y^3 + \text{ h.o.t.}$$

By a technique explained in [4] we can find the locus of vertices of the family of curves $f(x, y) = k$ for $k > 0$, and in [5] there are results on triple intersections and cusps of the symmetry set. We shall not recall these in detail here but will point out how they are verified by the example. Clearly any rotation about the

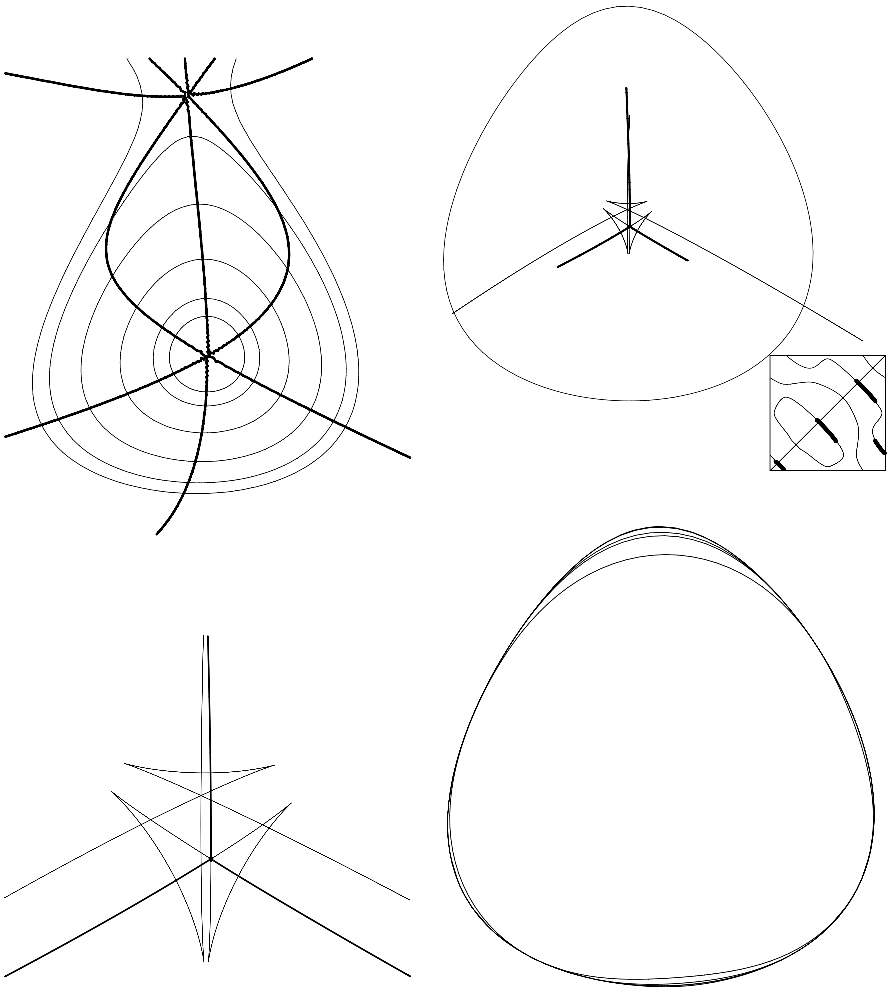


Fig. 2. The umbilic case. Top left: several level sets $f(x,y) = x^2 + y^2 + x^3 + 2x^2y + xy^2 - 3y^3 = k$ for k from 0.001 to 0.018. The curve becomes more circular as k decreases. The thick curves are the locus of vertices, obtained as in [4]. Notice that if k is too big then the level set ceases to be closed. Top right: The symmetry set and (boxed insert) the pre-symmetry set of the curve $f = k$ for a small enough value of k (0.01 in fact) that the structure has stabilised. The heavily drawn part of the pre-symmetry set corresponds to the Y-shaped medial axis. The curve $f = k$ was parametrized by the method of §3, using 10 terms. Bottom left: A closeup of the central part of the symmetry set; note the six cusps and two triple crossings. The medial axis is drawn heavily. Bottom right: The approximations to $f(x,y) = 0.01$ given by taking 1, 3, 4 and 10 terms as in §3: the outermost curve is for 10 terms and is essentially indistinguishable from the exact solution

z-axis will leave the equation of the surface in the same form as above. If we rotate to make $b_0 = b_2$ then it can be shown that the three branches of the vertex locus make angles with the x-axis which are multiples of 60°, as in Figure 2, top left. We pause here to establish the existence of such a rotation.

Lemma 1. *Let f be given by $f(x, y) = x^2 + y^2 + b_0x^3 + b_1x^2y + b_2xy^2 + b_3y^3+$ h.o.t. Then we can rotate the coordinates in the x, y plane so that in the new coordinates u, v, the function f takes the form $f = u^2 + v^2 + B_0u^3 + B_1u^2v + B_0uv^2 + B_3v^3+$ h.o.t.*

Proof. We rotate by an angle ϕ. Expressing f in terms of the new coordinates (u, v), amounts to replacing x by $u \cos\phi + v \sin\phi$ and y by $-u \sin\phi + v \cos\phi$ in the expression for $f(x, y)$. The new expression for f is of the form
$f = u^2 + v^2 + B_0u^3 + B_1u^2v + B_2uv^2 + B_3v^3+$ h.o.t., where

$$B_0 = b_0 \cos^3\phi - b_1 \cos^2\phi \sin\phi + b_2 \cos\phi \sin^2\phi - b_3 \sin^3\phi,$$

$$B_2 = 3b_0 \cos\phi \sin^2\phi - b_1 \sin^3\phi + 2b_1 \cos^2\phi \sin\phi - 2b_2 \cos\phi \sin^2\phi \\ \quad +b_2 \cos^3\phi - 3b_3 \cos^2\phi \sin\phi.$$

We need to show the existence of an angle ϕ_0 for which $B_0 = B_2$. Substitute $U = \tan(\frac{\phi}{2})$ and write $p = b_2 - b_0,\ q = b_3 - b_1$. Then the equation $B_0 = B_2$ reads

$$pU^6 + 6qu^5 - 15pU^4 - 20qU^3 + 15pu^2 + 6qU - p = 0.$$

Note that the equation has pairs of roots of the form $U, -1/U$, corresponding to solutions $\phi, \phi + \pi$. Of course, if $p = 0$ we can take $\phi = 0$. Otherwise the discriminant of this degree 6 equation is a positive constant times $(p^2 + q^2)^5$ and

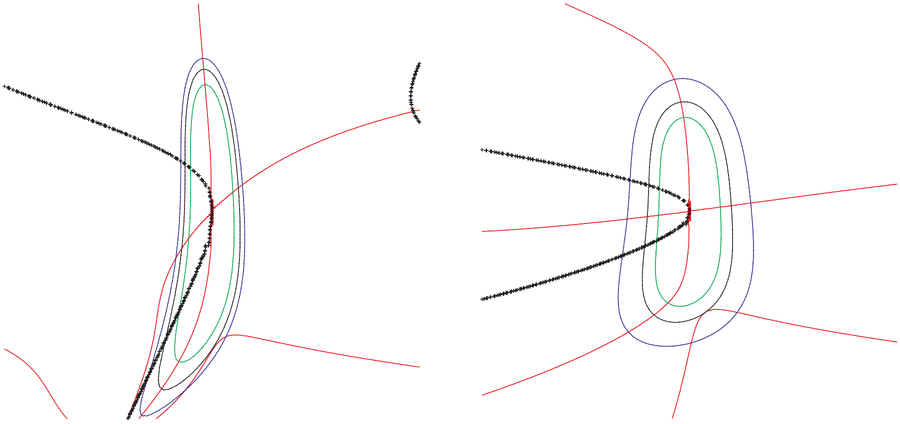


Fig. 3. Elliptic cusps of Gauss, examples 1 (left) and 2 (right). The closed curves are the level sets $f(x, y) = k$, the other thin curves are the loci of vertices of level sets and the thick curves are the loci of inflexions. Notice that for small enough k there are four vertices (as predicted by [4]) but as k increases the level set is first tangent to a branch of the locus of vertices not local to the origin and then intersects this locus in two points, giving six vertices

hence > 0. For a degree six equation this means that there are either two or six real solutions for U. □

Remark. We do not know a geometrical interpretation of the distinction between two and six solutions here. Note that these correspond to respectively one and three pairs of solutions $\phi, \phi + \pi$. There are a number of situations where through an umbilic pass one or three geometrically defined curves, for example ridges and sub-parabolic curves; see [3]. The present case appears to be different from these.

In Figure 2 we show an example to illustrate the above methods. It confirms the results of [4] and [5], for small enough $k > 0$, namely: (a) there are six vertices on the level set, resulting in three branches, of which one connects a maximum to maximum of curvature, one a minimum to minimum and one a maximum to

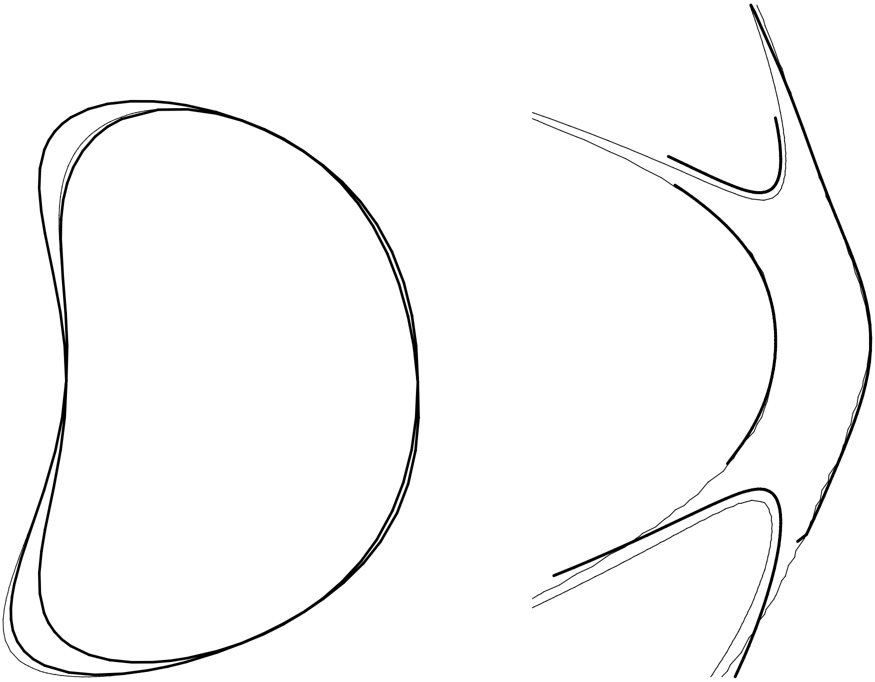


Fig. 4. Left: the thin curve is a level set $f(x, y) = k$ for an elliptic cusp of Gauss as in Example 1 in the text, and the thick lines are obtained by taking $r = r_0$ and $r = r_0 + r_1 t$ in the approximations described in §3. The approximation obtained by taking three terms $r = r_0 + r_1 t + r_2 t^2$ is visually indistinguishable from the true level set. Right: the thin curves are two level sets $f(x, y) = k$ for values of k with opposite signs in the case of a hyperbolic cusp of Gauss $f(x, y) = x^2 + 3xy^2 + 3xy^3 + y^4$. The thick curve is obtained with the first term $r = r_0$ of the approximation of §3. Clearly there are more problems here with capturing all the local features of the level sets. *Note:* in these figures the horizontal scale has been exaggerated compared to the vertical scale, as an aid to clarity

minimum, (b) there are two triple crossings on the symmetry set, (c) there are six cusps on the symmetry set. Note that full detail is given on the symmetry set since the polar approximation is smooth and extremely close to the level set $f = k$.

4.2 Cusps of Gauss

We consider here elliptic cusps of Gauss, as in (5), and assume $\kappa_1 > 0$ so that the level set $f = k$ is, locally to the origin, a simple closed curve for small $k > 0$. According to the general results of [4] there are four vertices and two inflexions on this closed curve. (The situation for hyperbolic cusps of Gauss—and indeed

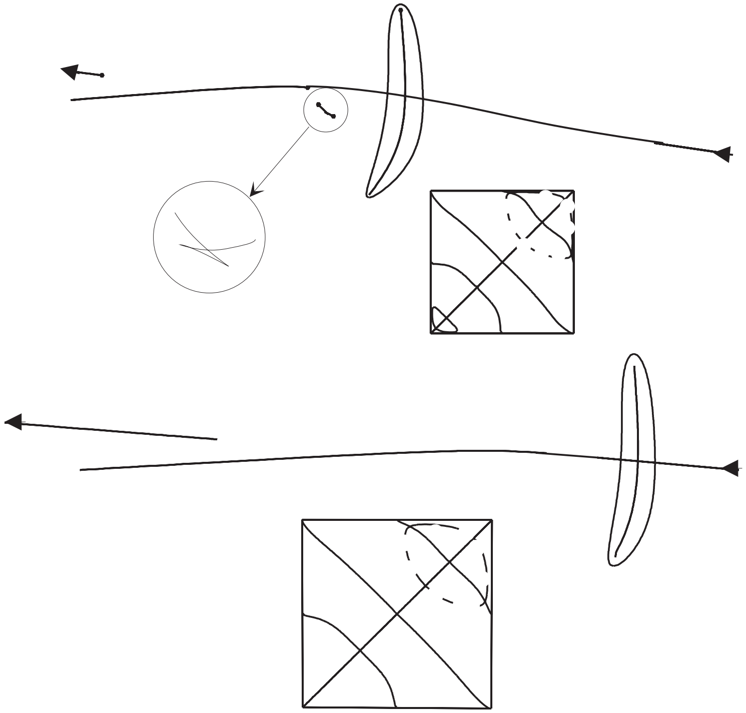


Fig. 5. Example 1 as in Figure 3, with annotated symmetry set and pre-symmetry set. The dashed section of the pre-symmetry set (boxed inserts) is due to the parallel tangents arising from the inflexions on $f(x, y) = k$; see §4. It is ignored when plotting the symmetry set (main diagram). The arrows indicate that a branch of the symmetry set goes to infinity when the bitangent circle becomes a bitangent line. The other endpoints are actual. Top: k is large enough for the level set to have six vertices. The additional branch of the pre-symmetry set is the loop towards the bottom left of the box and the corresponding piece of symmetry set is enlarged in the circle. As k decreases the number of vertices reaches its stable value of 4, as in the bottom figure. The medial axis for both values of k consists of the branch interior to the curve and the part of the other branch going to infinity to the left

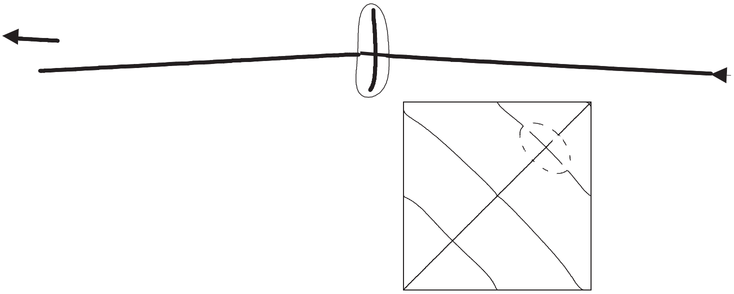


Fig. 6. Example 2 of Figure 3. Here k is taken small enough to make the level set have its stable number of vertices, namely 4. The symmetry set has very much the same structure as Example 1, that is two simple branches one of which extends to infinity

for hyperbolic points—is much more complicated, with the pattern of vertices and inflexions giving rise to several cases, which are detailed in [4].) Figure 3 shows several level sets together with the loci of vertices and inflexions, for two elliptic cusps of Gauss.

Example 1 is $f(x, y) = x^2 + 2x^2y + xy^2 + 2x^2y^2 - xy^3 + y^4$,
Example 2 is $f(x, y) = x^2 + 2x^2y + xy^2 + 2x^2y^2 - xy^3 + 6y^4$.

Figure 4, left, shows the curve of Example 1 together with approximations obtained as in §3. In this figure the horizontal scale is exaggerated to improve clarity (otherwise the approximations and the actual curve are too hard to distinguish!). We also show a hyperbolic cusp of Gauss example in Figure 4, right, though we do not go on to examine the symmetry set here.

Theoretical results on the symmetry sets of these curves appear to be much more difficult to obtain than those for elliptic, hyperbolic and ordinary parabolic points ([4, 5]). Thus evidence gathered from examples is all the more valuable. We show in Figure 5 the symmetry sets and pre-symmetry sets for the two examples of Figure 3. In both cases the stable situation as $k \to 0$ is two smooth branches, one of which extends to infinity. Thus the structure of the symmetry set for an elliptic cusp of Gauss appears to be very much simpler than for an umbilic point. In particular there are no cusps and no triple crossings. Furthermore, the only distinction between the symmetry sets of sections of a surface close to the tangent plane at an ordinary elliptic point and at an elliptic cusp of Gauss is that in the latter case one of the branches of the symmetry set extends to infinity. On the other hand, the distinction between an ordinary elliptic point and an umbilic is very striking: in the latter case there are six cusps and two triple crossings as in Figure 2.

5 Conclusion

We have given a simple method of parametrizing closed level set curves $f(x, y) = k$ to any desired degree of accuracy and used it to compute symmetry sets and medial axes of sections of a surface $z = f(x, y)$ in Monge form close to the tangent plane at an umbilic and an elliptic cusp of Gauss. The method applies also to level sets $f(x, y) = k$ which are not closed; here the patterns of vertices and inflexions on the plane sections become much more complicated [4] and there are many different cases. Furthermore there are technical problems in ensuring that the approximations capture all the necessary local information about the level sets. We shall pursue these cases elsewhere.

There are many related questions; for example, when a point on a surface moves from the hyperbolic region to the parabolic curve, how does the family of symmetry sets of the parallel plane sections behave? This is tantamount to considering a one-parameter family of *surfaces*, that is the curves will now belong to a *two*-parameter family. This again is the subject of further work.

Acknowledgements. The work of the first two named authors is a part of the DSSCV project supported by the IST Programme of the European Union (IST-2001-35443). They also acknowledge EPSRC for the provision of a graphics computer used to draw the symmetry sets.

References

1. T.F.Banchoff, T.Gaffney and C.McCrory, *Cusps of Gauss Mappings*, Pitman Research Notes in Mathematics, 55, 1982.
2. J.W. Bruce and P.J. Giblin, 'Growth, motion and one-parameter families of symmetry sets', *Proc. Royal Soc. Edinburgh* 104A (1986), 179–204.
3. J.W.Bruce, P.J.Giblin and F.Tari, 'Ridges, crests and sub-parabolic lines of evolving surfaces', *Int. J. Computer Vision.* 18 (1996), 195–210.
4. A. Diatta and P.J. Giblin, 'Vertices and inflexions of plane sections of surfaces in $\mathbb{R}^3$', to appear in *Real and Complex Singularities 2004, LMS Lecture Notes in Maths.*, Cambridge University Press. Available from http://www.liv.ac.uk/~pjgiblin
5. A.Diatta and P.J.Giblin, 'Geometry of isophote curves', *Scale Space and PDE Methods in Computer Vision, Lecture Notes in Computer Science* 3459 (2005), 50–61. Available from http://www.liv.ac.uk/~pjgiblin
6. P.J.Giblin, 'Symmetry sets and medial axes in two and three dimensions', *The Mathematics of Surfaces IX*, Roberto Cipolla and Ralph Martin (eds.), Springer-Verlag 2000, pp. 306–321.
7. J.J. Koenderink, *Solid Shape*, M.I.T. Press (1990)
8. R.J.Morris, *Liverpool Surface Modelling Package*, http://www.amsta.leeds.ac.uk/~rjm/lsmp/ See also R.J.Morris, 'The use of computer graphics for solving problems in singularity theory', in *Visualization in Mathematics*, H.-C.Hege & K.Polthier, Heidelburg: Springer-Verlag (1997), 173–187, and http://www.scs.leeds.ac.uk/pfaf/lsmp/SingSurf.html

An Heuristic Analysis of the Classification of Bivariate Subdivision Schemes

Neil A. Dodgson

University of Cambridge Computer Laboratory,
15 J. J. Thomson Avenue, Cambridge, UK CB3 0FD
nad@cl.cam.ac.uk

Abstract. Alexa [1] and Ivrissimtzis *et al.* [2] have proposed a classification mechanism for bivariate subdivision schemes. Alexa considers triangular primal schemes, Ivrissimtzis *et al.* generalise this both to quadrilateral and hexagonal meshes and to dual and mixed schemes. I summarise this classification and then proceed to analyse it in order to determine which classes of subdivision scheme are likely to contain useful members. My aim is to ascertain whether there are any potentially useful classes which have not yet been investigated or whether we can say, with reasonable confidence, that all of the useful classes have already been considered.

I apply heuristics related to the mappings of element types (vertices, face centres, and mid-edges) to one another, to the preservation of symmetries, to the alignment of meshes at different subdivision levels, and to the size of the overall subdivision mask. My conclusion is that there are only a small number of useful classes and that most of these have already been investigated in terms of linear, stationary subdivision schemes. There is some space for further work, particularly in the investigation of whether there are useful ternary linear, stationary subdivision schemes, but it appears that future advances are more likely to lie elsewhere.

1 Introduction

Alexa [1] and Ivrissimtzis *et al.* [2] propose a classification of subdivision schemes. Alexa classifies all triangular primal schemes. Ivrissimtzis *et al.* extend this both to quadrilateral and hexagonal base meshes and to dual and mixed schemes (this terminology is explained later in this section). The extension to quadrilateral meshes is based on Sloan's work on 2D lattices [3].

While this classification tells of the existence of many classes of subdivision scheme, it does not give any indication as to which classes are likely to contain useful schemes. This paper analyses Ivrissimtzis *et al.*'s classification with the intention of determining which classes are likely to contain useful (stationary, linear) subdivision schemes and which classes are unlikely to contain useful schemes. I expect that there will be an indeterminate region between those classes which clearly contain useful schemes and those classes which clearly do not. I assume that the reader is familiar with subdivision [4].

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 161–183, 2005.

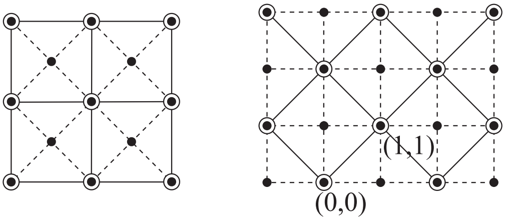


Fig. 1. Open circles are source vertices; black dots are subdivided vertices. The solid lines are the source mesh; the dashed lines are the subdivided mesh. At left is a visualisation of $QP(1, 1)$ subdivision as we usually think of it: a new vertex is introduced at the centre of each quadrilateral, the old vertices are adjusted, and the new grid is constructed as shown. At right is an equivalent visualisation, this time with the subdivided grid aligned horizontally and vertically. If the edges of the subdivided mesh are assigned unit length then this is the coordinate system used by Ivrissimtzis *et al.* for quadrilateral meshes, which is used throughout this paper

Subdivision schemes may be classified in a variety of ways. Ivrissimtzis, Sabin and I use a hierarchy of detail, where the top level classes encompass many subdivision schemes, while the lowest level precisely specifies a single scheme. The hierarchy has the following levels (this is an expanded form of the list given by Ivrissimtzis *et al.* [2]).

Base mesh type. This is the base mesh in the regular case. Most subdivision schemes are based on either a quadrilateral or a triangular mesh. It is also possible to base a scheme on an hexagonal mesh, this being the only other regular monohedral tiling of the plane [5], or on one of the semi-regular tilings of the plane.

Mapping. This concerns how vertices, face centres, and mid-edges map to one another from one level of subdivision to the next. Face centres and mid-edges refer to these points in a regular tiling of the plane. If one applies subdivision to a regular tiling of the plane, the mapping is exact. In the case of a general mesh in 3D space, we can think of the regular tiling of the plane as a parameterization of the actual mesh. Ivrissimtzis *et al.* [2] classify schemes based on whether vertices map to vertices or to face centres. In this paper I extend this to consider what elements are mapped to by face centres and to consider also the mappings of mid-edges.

Arity. This describes how the source grid maps to the subdivided grid in the regular case. It can be represented either as a scalar, representing the ratio of the lengths of edges in the source and subdivided grids, or as an ordered pair, (n, m), giving the relative position, in the coordinate system of the subdivided grid, of one source vertex with respect to an adjacent source vertex (see Fig. 1); in the case of the hexagonal grid, of the position of one source face centre with respect to an adjacent one (see Fig. 2). Without loss of generality we can take $n > 0$ and $0 \leq m \leq n$. Thus $(2, 0)$ represents binary subdivision (e.g. Catmull-Clark [6], Doo-Sabin [7], Loop [8]), while $(1, 1)$ represents the $\sqrt{2}$ class for quadrilateral grids (e.g. simplest [9], Peters-

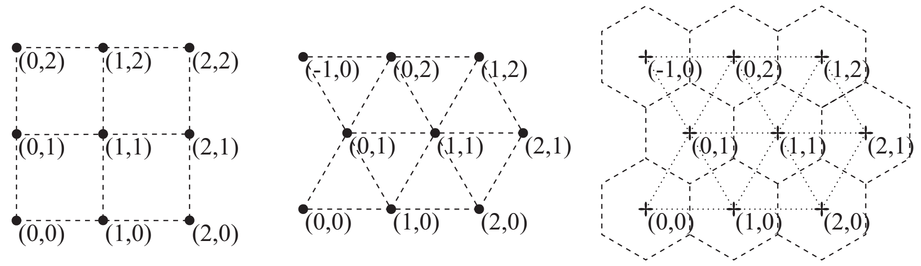


Fig. 2. The coordinate systems of the three mesh types. The quadrilateral mesh has the conventional coordinate system. Each edge is of unit length. The triangular mesh has axes at an angle $\pi/3$ to one another, with all edges of unit length. The hexagonal mesh is more complex. As with the triangular mesh, the axes are at an angle $\pi/3$ to one another, but it is the face centres which are at integer coordinates; edges are of length one-third, and vertices are at $\left(x+\frac{1}{3}, y+\frac{1}{3}\right)$, $\left(x+\frac{2}{3}, y+\frac{2}{3}\right)$, $x, y \in \mathbb{Z}$. This makes the hexagonal mesh a precise dual of the triangular mesh, as illustrated in the figure. See Appendix A for more on this coordinate system

Shiue [10], Velho [11, 12]) and the $\sqrt{3}$ class for triangular and hexagonal grids (e.g. Kobbelt's $\sqrt{3}$ [13], hexagon-by-three [14]). Examples are shown in Fig. 3.

Footprint. Having chosen values for the above three, the next level is to specify which new vertices are affected by a given source vertex in the regular case. This corresponds to specifying which coefficients in the subdivision mask are non-zero. A larger footprint gives greater freedom in choice of coefficients but also greater computation and increased difficulties in handling extraordinary points. Of the well-known published schemes, simplest [9] has the smallest footprint (4 vertices) while Catmull-Clark [6], butterfly [15], and Kobbelt [16] have the largest (25 vertices in each case). Amongst more recent schemes, ternary Loop [17] has 61 non-zero coefficients and interpolating ternary triangular [18] has up to 85. I note that the terminology is not consistent in the literature, so it is worth saying that I am using Sabin's definition of the term *mask* [19] where the mask shows the contributions made to each new vertex by a given old vertex, c.f. the *stencils* where a stencil shows the contributions made by each old vertex to a given new vertex.

Mask coefficients. The next step is to decide what values the coefficients should have. For B-spline based and box-spline based schemes, there is no freedom beyond choosing the particular spline basis, as the coefficients must be derived from the spline on which they are based. Other schemes have more freedom (e.g. butterfly [15], Kobbelt [16], interpolating ternary triangular [18]). Amongst other things, the choice of coefficients determines whether the scheme is interpolating or approximating. Interpolating schemes (e.g. butterfly [15]) are those where the limit surface is constrained to pass through the source vertices. Approximating schemes (e.g. Loop [8]) do not have this constraint.

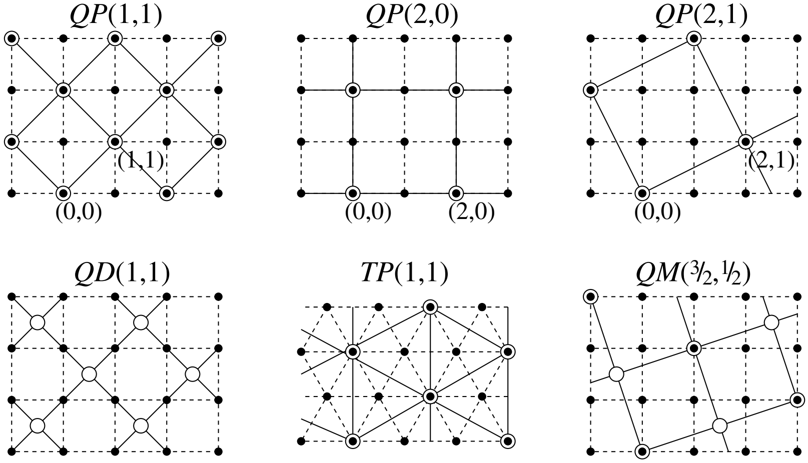


Fig. 3. Some example classes. Open circles are source vertices; black dots are subdivided vertices. The solid lines are the source mesh; the dashed lines are the subdivided mesh. Note how the (n, m) notation gives the coordinates, in the coordinate system of the subdivided grid, of an adjacent source vertex with respect to an arbitrary origin source vertex; to illustrate this, the top line of examples has an origin and the appropriate adjacent source vertex explicitly labelled with their coordinates

Extraordinary cases, boundaries, and creases. The final step is to handle the extraordinary cases. This is the step which requires a significant amount of careful thought and analysis. Some schemes have more than one proposed method for handling extraordinary cases. For example, the schemes based on the bivariate quadratic and cubic B-splines are commonly known as Doo-Sabin and Catmull-Clark subdivision respectively but, in fact, each of them has two variant mechanisms for handling extraordinary cases: one proposed by Catmull and Clark [6] and one proposed by Doo and Sabin [7]. Boundaries of the mesh must also be handled as special cases as must creases [4] in the mesh which are internal edges along which a designer wants reduced continuity.

Alexa [1] and Ivrissimtzis *et al.* [2] consider the top three levels of this hierarchy. This paper analyses that classification in order to ascertain which classes are likely to contain useful schemes.

2 Summary of the Classification Notation

Ivrissimtzis *et al.* [2] use notation of the form $AB(n, m)$, where A is the base mesh type, B the mapping, and (n, m) the arity. Occasionally it is convenient to use $A(n, m)$ as a shorthand for all classes with the same base mesh type and arity. The coordinate systems are illustrated in Fig. 2 and example classes are shown in Fig. 3.

A can be Q (quadrilateral), T (triangular) or H (hexagonal). The right-triangle based schemes (e.g. Velho's 4-8 scheme [11, 12]) are regarded as Q schemes, because the vertices lie on the quadrilateral grid in the regular case. The right-triangle tiling, its dual (the octagon-square semi-regular tiling), and other semi-regular tilings, could be considered as primitive base mesh types in their own right, but Ivrissimtzis *et al.* [2] limit the classification to the three regular base tilings.

B can be P (primal), D (dual) or M (mixed) where *primal* means that all vertices map to vertices, *dual* that all vertices map to face centres, and *mixed* that vertices map to a combination of vertices and face centres. This classification as 'primal' and 'dual' arises from the $(2, 0)$ classes for which the terminology is well known [20] and where it is related to the concept of the dual graph. Sabin [19] points out that the classification as 'primal' and 'dual' is not necessarily particularly satisfactory for the general case. For example, in $Q(2, 0)$ classes it is related to the concept of face-splitting (primal) or vertex-splitting (dual), but this face- or vertex-splitting relationship fails for most other arities. In particular, Oswald and Schröder note that it fails for $(1, 1)$ classes [21]. The limitations of this classification are explored further in Sect. 3.3.

(n, m) is the arity, as described in Sect. 1. There are certain quirks in the specification of arity for the TM and HM (triangular mixed and hexagonal mixed) classes, which I will gloss over here as they have no impact on the conclusions of this paper (for details see Ivrissimtzis *et al.* [2]). The term *arity* can refer to either (n, m) or to the length of the vector, which is $\sqrt{n^2 + m^2}$ for Q and $\sqrt{(n + m/2)^2 + (\sqrt{3}n/2)^2}$ for T and H.

The classes of arity $(1, 0)$ represent schemes which do not subdivide. These can be identity schemes, where the mesh does not change at all, or other point-processing schemes comparable with filters used in image processing. The simplest application of these would be mesh smoothing.

There is an interesting case with the lowest possible arity class considered by Ivrissimtzis *et al.* [2], which is the class of $QM\left(\frac{1}{2}, \frac{1}{2}\right)$ schemes. The arity (length of the (n, m) vector) is $\frac{1}{\sqrt{2}}$, which is less than unity, and therefore this class represents decimation schemes, rather than subdivision schemes.

3 Heuristic Analysis

This classification allows for a large number of potential subdivision schemes. This paper asks which of these classes are likely to contain useful schemes and thus reward further investigation and, conversely, which are likely to have unresolvable problems. To facilitate a partition into usable and unusable classes, I sequentially introduce heuristics, each providing more stringent requirements on what is meant by "usable".

An heuristic is a rule of thumb, a guideline which helps us to consider only the useful alternatives. In subdivision, one early heuristic appears to have been "only binary schemes are worth considering." This apparant heuristic has been seriously challenged by the discovery and development of $\sqrt{2}$ [9, 11, 12, 22, 23],

$\sqrt{3}$ [13, 24] and ternary [17, 25, 18] schemes. They have not, however, completely invalidated it because all commercial systems are based on binary schemes. An up-to-date version of this example heuristic would therefore seem to be something like “only schemes based on Catmull-Clark or Loop are worth considering in a commercial context.” As with all heuristics, it is possible to argue both for and against it.

This paper sets out a number of heuristics which are designed to reduce the enormous number of potential schemes which are allowed for by the classification mechanism. None of the heuristics is a hard and fast rule and not all of them have a solid mathematical justification. Nevertheless, I believe that they are rules of which all practitioners of subdivision become aware, whether consciously or not. It may well be that, as with the “only binary schemes are useful” heuristic, some of these heuristics will prove to be false guides. The commentary following each heuristic therefore incorporates discussion of those situations in which the heuristic appears to be a less than perfect guide.

3.1 Heuristics Implicit in Ivrissimtzis *et al.*’s Classification

The first two heuristics are implicit in Ivrissimtzis *et al.*’s [2] classification system. The classification is thus already making assumptions about which types of subdivision schemes are likely to prove useful. For comparison, Han has produced a much more restricted classification system for subdivision schemes [26] in which he implicitly assumes that Heuristics 1–6 are true.

Heuristic 1. *Only regular monohedral tilings of the plane are useful as base meshes.*

This limits the base mesh in the regular case to being quadrilateral, triangular, or hexagonal, with the individual polygons being regular. There are subdivision schemes which appear to be based on a right-triangle mesh [11, 12] but these can be treated as Q schemes, because the vertices lie on the quadrilateral grid in the regular case; the right-triangle concept simply serves to make the explanation and implementation of the scheme somewhat easier in practice. The right-triangle tiling, its dual (the octagon-square semi-regular tiling), and other semi-regular tilings, could be considered as primitive base mesh types in their own right. In addition to semi-regular tilings it may be possible to create a subdivision scheme based on an aperiodic tiling, such as a Penrose tiling [27]. In any semi-regular or aperiodic case there would seem to be some difficulty in specifying the base mesh for an object and in extending the subdivision scheme to handle extraordinary cases, boundaries, and creases. Nevertheless, Ivrissimtzis, Claes, and I undertook some preliminary work on octagon-square subdivision schemes in 2003. It was clear from this that some sort of octagon-square subdivision scheme is possible, although the above difficulties would have to be faced; in particular it is difficult to see how to handle extraordinary faces with an odd number of edges. It was also clear that the vertices do not lie on one of the three regular meshes, unlike the right-triangle mesh whose vertices lie on the quadrilateral mesh. There may be some advantage in investigating such schemes but they seem to pose immense difficulties. Furthermore, the classification mechanism does not admit such schemes. I do not consider them further.

Heuristic 2. *Every vertex at one level of subdivision must map to either a vertex or a face centre at the next level.*

Ivrissimtzis *et al.*'s [2] classification assumes this. The second letter in the classification indicates whether the mapping is to vertices (P), face centres (D) or a mixture (M). The initial motivation for this was from consideration of primal and dual binary schemes which have either a P or D behaviour. I conjecture that it would be possible to construct a subdivision scheme where vertices at one level map to some feature other than a vertex or face centre at the next level, but that it is likely that such a scheme would not prove useful because, as described under Heuristic 3 below, it may well produce an infinite number of possible limit surfaces for the same base mesh and, as described under Heuristic 4 below, it would definitely not maintain the rotational symmetries of the mesh. This conjecture has not been tested but, as with Heuristic 1, there seem to be great difficulties with such schemes and, furthermore, the classification mechanism does not admit such schemes. I do not consider them further.

3.2 Heuristics from the Need for a Single Limit Surface

The next two heuristics are based on the desire for a subdivision schemes to produce a single deterministic limit surface, rather than an infinite number of possible limit surfaces. This requires that the limit surface depend solely on the positions and connectivity of the initial base mesh, not on any arbitrary labelling of vertices. These two heuristics exclude those classes which require such an arbitrary labelling.

Heuristic 3. *All vertices at one level of subdivision must map to the same new element type at the next level.*

The term *element* refers to a vertex, face centre, or mid-edge. It is reasonable to require all vertices to be treated identically under refinement because failure to adhere to this heuristic can lead to there being multiple possible limit surfaces for a single base mesh. In these cases, the limit surface will, in general, depend on which particular vertices map to vertices and which do not. This decision must be made at every subdivision step (see Fig. 4) and therefore there is a potentially infinite number of different, equally valid, limit surfaces for any base mesh. In the case of a finite base mesh, one vertex will be chosen as the origin and all the other vertices will eventually map to no element. There will thus be as many possible limit surfaces as there are vertices in the base mesh. The particular limit surface which is arrived at thus depends on something more than just the location and connectivity of the base mesh's vertices: this is undesirable. In addition, it is difficult to see how such schemes could be extended to handle extraordinary cases, boundaries, and creases.

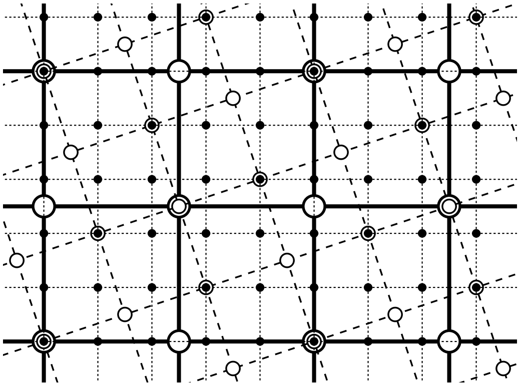


Fig. 4. An example of the arbitrary choices which have to be made in a mixed subdivision schemes. This is $QM\left(\frac{3}{2},\frac{1}{2}\right)$ with the rotation direction alternating on alternate subdivision steps. At the first level of subdivision, half of the vertices map to vertices and the other half map to face centres. At the next level of subdivision half of those vertices map to face centres, and so on. In the limit, at most one of the original vertices will map to a vertex and the choice of this original vertex is arbitrary. There are at least as many limit surfaces as there are original vertices. The mappings for this subdivision class are: $\mathsf{v} \rightarrow \mathsf{v}$ *or* f (half of the vertices will map to vertices, the other half to face centres), $\mathsf{f} \rightarrow \mathsf{e}$, $\mathsf{e} \rightarrow \mathsf{x}$. In the limit, all original vertices (but one) map to no feature at all in the limit surface as they all follow the mapping sequence $\mathsf{v} \rightarrow \mathsf{v} \rightarrow \cdots \rightarrow \mathsf{v} \rightarrow \mathsf{f} \rightarrow \mathsf{e} \rightarrow \mathsf{x} \rightarrow \mathsf{x} \rightarrow \cdots$

This heuristic eliminates all mixed classes because, in mixed classes, some vertices map to vertices and some map to face centres. Therefore all TM, QM and HM classes are unlikely to produce useful subdivision methods.

It might be sensible to extend this heuristic to say that all face centres must map to the same element type and that all mid-edges must map to the same element type. This would eliminate most of the TD classes (all except those for which $n + 2m \bmod 6 = 0$, see Table 3 and Appendix A for the detailed calculations of these restrictions). However, while I am convinced that this extension to face centres and mid-edges is sensible, I find it difficult to see how the above argument regarding multiple limit surfaces can be extended to these cases and, furthermore, all classes which would be excluded by such an extension are excluded by the next heuristic anyway.

Heuristic 4. *All rotational symmetries should be maintained under refinement.*

The requirement is that centres of k-fold rotational symmetry (k-centres) at one refinement level have k-fold rotational symmetry at the next level. k-centres may, of course, become centres of higher rotational symmetry provided that the higher symmetry preserves k-fold symmetry. This heuristic seems reasonable because a loss of rotational symmetry leads to multiple possible limit surfaces from the same source mesh. Consider, for example, a vertex in a triangular mesh (6-fold rotational symmetry) which maps to a face centre (3-fold rotational symmetry) under subdivision. There are two possible ways in which this could happen. In simple terms, the vertex maps either to an up-pointing triangle or to a down-pointing triangle. The decision as to which vertices map in which way must be taken at each subdivision step. Even a finite triangular base mesh, with no extraordinary vertices, will thus have infinitely many possible limit surfaces. As with Heuristic 3, the limit surface thus depends on something other than just the location of vertices and the connectivity of the mesh. This is undesirable.

Failing to preserve rotational symmetry also makes it difficult to extend a scheme to handle irregular cases. A particular example of this is considered by Dodgson *et al.* [28] where the $TD(1,1)$ class is explored and a particular $TD(1,1)$ scheme demonstrated; both the particular scheme and the class as a whole are shown to have severe problems. I conjecture that similar problems with irregular cases will arise in any scheme which fails to preserve rotational symmetry. A proof of this conjecture is beyond the scope of this paper because the "multiple limit surface" argument, above, is sufficient justification for this heuristic.

Ivrissimtzis *et al.* [2] suggest that symmetry considerations would be an alternative way to approach the classification problem and it is clear that symmetry considerations are important in subdivision. Han explicitly uses symmetry considerations in his alternative classification mechanism for QP and TP subdivision schemes [26].

The centres of rotational symmetry are the vertices, face centres, and mid-edges of the lattice. I will denote these elements as v, f, and e respectively. The rotational symmetry of each element is shown in Table 1(a).

I use → to indicate a mapping of an element from one level of refinement to the next and, in particular, $k \rightarrow k'$ to indicate a mapping from k-fold rotational symmetry to k'-fold rotational symmetry. Under this heuristic, allowable symmetry mappings between values of k and k' are, for Q, $2 \rightarrow 2$, $4 \rightarrow 4$, and $2 \rightarrow 4$; for T and H, $2 \rightarrow 2$, $3 \rightarrow 3$, $6 \rightarrow 6$, $2 \rightarrow 6$, and $3 \rightarrow 6$. Note that $2 \rightarrow 3$ is not allowed because a 3-centre is not also a 2-centre. The mappings in Table 1(b) are thus the only ones which are permitted.

Table 1. (a) The rotational symmetries of the different elements. (b) The allowable mappings under the restrictions of Heuristic 4

(a)

Element	Q	T	H
vertex (v)	4	6	3
face centre (f)	4	3	6
mid-edge (e)	2	2	2

(b)

Q	T	H
v → v	v → v	v → v
v → f		v → f
f → v	f → v	
f → f	f → f	f → f
e → v	e → v	
e → f		e → f
e → e	e → e	e → e

From this we see that triangular dual (TD) classes are not allowed because they map vertices to face centres. Alexa's concentration on the primal classes (TP, v → v) for triangular subdivision is therefore vindicated as neither dual nor mixed schemes are useful in the triangular case.

We can also see that any hexagonal scheme which maps face centres to vertices is not allowed, which excludes some of the hexagonal primal HP classes (those for which $(n - m) \bmod 3 = 0$, see Table 4). Thus, of the hexagonal classes, only HD classes and a subset of HP classes are considered useful.

For triangular classes, the only cases in which edges do not map to an appropriate element are already excluded by considering those cases where vertices or face centres do not map to an appropriate element. Therefore it is a moot point whether we need consider the mapping of the rotational symmetries of mid-edges as they are never called into play as a criterion for exclusion. For hexagonal classes, it is possible for an HP class to have v → v and f → v but to have mid-edges mapping to points with no rotational symmetry (see Table 4). I conjecture that these should also be excluded.

3.3 The Limitations of the Primal/Dual Notation

Details of how the above mappings are calculated can be found in Tables 2–4 and Appendix A. The fact that the calculations for HD and TP and for HP and TD are not exact duals of one another shows up a subtle bias in the classification. The classification is vertex-centric: it explicitly tells us whether a vertex maps to a vertex or a face centre. Arguably of equal significance is whether a face centre maps to a vertex or a face centre. Fortunately this information can be derived directly from the notation. There are four cases:

vv	vertex preserving	v → v, f → v
ff	face preserving	v → f, f → f
vf	preserves both	v → v, f → f
fv	preserves neither	v → f, f → v

The notation, ab, at left above is shorthand for v → a, f → b. This provides a more explicit representation of the mappings which occur than does the simple P and D labelling used by Ivrissimtzis *et al.* [2]. Note that fv is something of a special case because a subdivision scheme which is of type fv is of type vf if one considers two steps of subdivision.

Heuristic 4 restricts us to eight useful classes of subdivision scheme. A Q scheme can be any of vv, ff, vf, or fv; while a T scheme can only be vv or vf; and an H scheme can only be ff or vf.

In addition to vertices and face centres, the mappings of mid-edges can also be considered, for completeness. Tables 2–4 show how the mappings can be derived directly from the notation. We see that there are a limited set of valid mappings. I use the notation vfe → abc to indicate v → a, f → b, and e → c, extending the notation above to include edge mappings. Where context is clear I use just abc to indicate the same thing. Note that TD classes allow the possibility that an edge can map to a point with no rotational symmetry (indicated by x) and

Table 2. Calculation of the vfe coding for the quadrilateral QP and QD classes. Details of the derivation of these formulæ can be found in Appendix A

$QP(n, m) \Rightarrow \mathsf{v} \rightarrow \mathsf{v}$

$$\begin{array}{ll}
(n-m) \bmod 2 = 0 & \Rightarrow \mathsf{f} \rightarrow \mathsf{v} \\
(n-m) \bmod 2 = 1 & \Rightarrow \mathsf{f} \rightarrow \mathsf{f} \\
 & \\
n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \mathsf{e} \rightarrow \mathsf{v} \\
n \bmod 2 = m \bmod 2 = 1 & \Rightarrow \mathsf{e} \rightarrow \mathsf{f} \\
n \bmod 2 \neq m \bmod 2 & \Rightarrow \mathsf{e} \rightarrow \mathsf{e}
\end{array}$$

Possible scheme types are: vvv, vvf, and vfe.

$QD(n, m) \Rightarrow \mathsf{v} \rightarrow \mathsf{f}$

$$\begin{array}{ll}
(n-m) \bmod 2 = 0 & \Rightarrow \mathsf{f} \rightarrow \mathsf{f} \\
(n-m) \bmod 2 = 1 & \Rightarrow \mathsf{f} \rightarrow \mathsf{v} \\
 & \\
n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \mathsf{e} \rightarrow \mathsf{f} \\
n \bmod 2 = m \bmod 2 = 1 & \Rightarrow \mathsf{e} \rightarrow \mathsf{v} \\
n \bmod 2 \neq m \bmod 2 & \Rightarrow \mathsf{e} \rightarrow \mathsf{e}
\end{array}$$

Possible scheme types are: fff, ffv, and fve.

Table 3. Calculation of the vfe coding for the triangular TP and TD classes. Details of the derivation of these formulæ can be found in Appendix A

$TP(n, m) \Rightarrow \mathsf{v} \rightarrow \mathsf{v}$

$$\begin{array}{ll}
(n-m) \bmod 3 = 0 & \Rightarrow \mathsf{f} \rightarrow \mathsf{v} \\
\text{otherwise} & \Rightarrow \mathsf{f} \rightarrow \mathsf{f} \\
 & \\
n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \mathsf{e} \rightarrow \mathsf{v} \\
\text{otherwise} & \Rightarrow \mathsf{e} \rightarrow \mathsf{e}
\end{array}$$

Possible scheme types are: vvv, vve, vfv, and vfe.

$TD(n, m) \Rightarrow \mathsf{v} \rightarrow \mathsf{f}$

$$\begin{array}{ll}
(n-m) \bmod 3 = 0 & \Rightarrow \mathsf{f} \rightarrow \mathsf{f} \\
\text{otherwise} & \Rightarrow \mathsf{f} \rightarrow \frac{\mathsf{f}}{\mathsf{v}} \\
 & \\
n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \mathsf{e} \rightarrow \mathsf{f} \\
\text{otherwise} & \Rightarrow \mathsf{e} \rightarrow \mathsf{x}
\end{array}$$

Possible scheme types are: fff, ffx, $\mathsf{f}\frac{\mathsf{f}}{\mathsf{v}}\mathsf{f}$, and $\mathsf{f}\frac{\mathsf{f}}{\mathsf{v}}\mathsf{x}$.

that half of the face centres can map to face centres while the other half map to vertices (indicated by $\frac{\mathsf{f}}{\mathsf{v}}$). These possibilities are a consequence of allowing the $\mathsf{v} \longrightarrow \mathsf{f}$ mapping which reduces a 6-centre to a 3-centre, and provides further justification for Heuristic 4. A similar observation about edges mapping to points with no rotational symmetry can be made about some of the HP classes.

Fig. 5 illustrates the low arity classes. It shows at least one class of each of the mapping types for QP, QD and TP.

Table 4. Calculation of the vfe coding for the hexagonal HP and HD classes. Details of the derivation of these formulæ can be found in Appendix A

$HP(n,m) \Rightarrow \mathsf{v} \rightarrow \mathsf{v}$

$$\begin{aligned} \mathsf{v}_{\bigtriangledown} \rightarrow \mathsf{v}_{\bigtriangleup} &\Rightarrow c = 1 \\ \mathsf{v}_{\bigtriangledown} \rightarrow \mathsf{v}_{\bigtriangledown} &\Rightarrow c = 2 \end{aligned}$$

$$\begin{array}{ll} (n-m) \bmod 3 = 0 & \Rightarrow \ \mathsf{f} \rightarrow \mathsf{v} \\ (n-m) \bmod 3 = 3-c & \Rightarrow \ \mathsf{f} \rightarrow \mathsf{f} \\ (n-m) \bmod 3 = c & \Rightarrow \ HM, \text{ not } HP \end{array}$$

$$\begin{array}{ll} (n-m) \bmod 3 = 0 & \text{and} \\ n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \ \mathsf{e} \rightarrow \mathsf{v} \\ \text{otherwise} & \Rightarrow \ \mathsf{e} \rightarrow \mathsf{x} \end{array}$$

$$\begin{array}{ll} (n-m) \bmod 3 = 3-c & \text{and} \\ n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \ \mathsf{e} \rightarrow \mathsf{f} \\ \text{otherwise} & \Rightarrow \ \mathsf{e} \rightarrow \mathsf{e} \end{array}$$

Possible scheme types are: vvv, vvx, vff, and vfe.

$HD(n,m) \Rightarrow \mathsf{v} \rightarrow \mathsf{f}$

$$\begin{array}{ll} (n-m) \bmod 3 = 0 & \Rightarrow \mathsf{f} \rightarrow \mathsf{f} \\ \text{otherwise} & \Rightarrow HM, \text{ not } HD \end{array}$$

$$\begin{array}{ll} n \bmod 2 = m \bmod 2 = 0 & \Rightarrow \mathsf{e} \rightarrow \mathsf{f} \\ \text{otherwise} & \Rightarrow \mathsf{e} \rightarrow \mathsf{e} \end{array}$$

Possible scheme types are: fff and ffe.

3.4 Heuristics Based on Observation of Current Practice

While the previous two heuristics are based on a desire to have a single deterministic limit surface, the following heuristics are much less clear-cut and I therefore address their limitations as well as their merits in the discussion.

Heuristic 5. *Allow only schemes which align the mesh at one level of refinement with the mesh at some higher level of refinement.*

This heuristic was explored by Alexa [1] for the TP classes. It says that a scheme needs to produce a mesh which is in the same rotational orientation as the base mesh after a finite number of steps. For all three types of base mesh, this heuristic permits only $(n, 0)$ and (n, n) classes. Alexa [1] proves this for T classes, so it is true for H classes by geometric duality. It is also true for Q classes because it is true by inspection for $(n, 0)$ and, for (n, m), $m > 0$ it requires:

$$\tan \frac{2\pi}{p} \in \mathbb{Q},\ p \in \mathbb{Z}^{+},\ 0 < \frac{2\pi}{p} \leq \frac{\pi}{4}$$

whose only solution [29] is $p = 8$ and therefore $m = n$. In Han's classification of TP and QP schemes [26], his symmetry conditions force this heuristic to be true

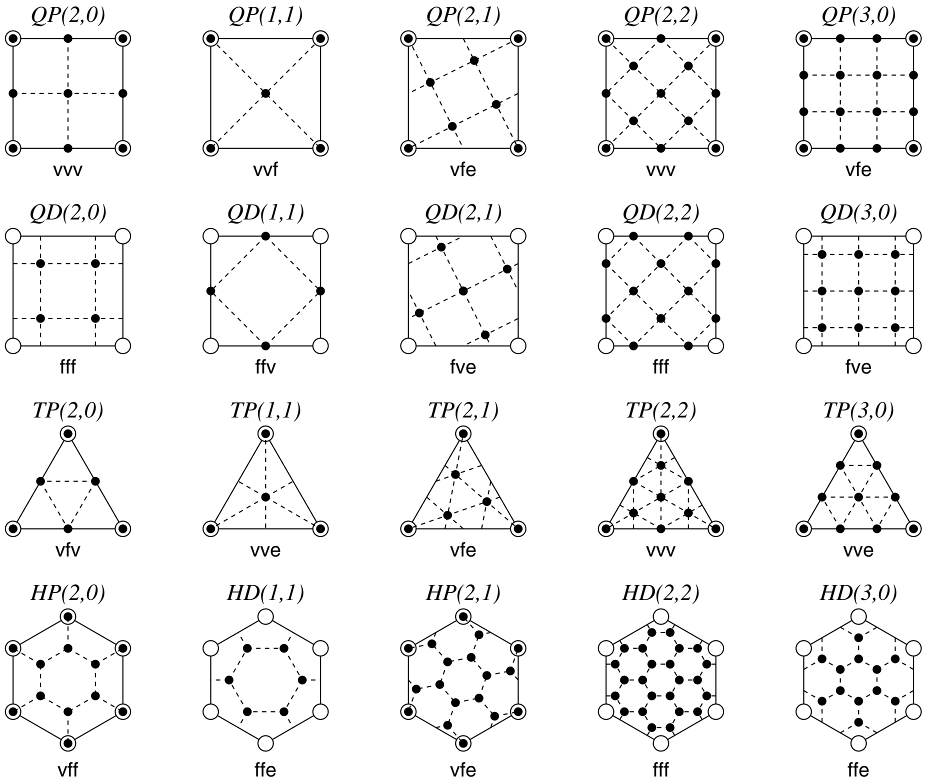


Fig. 5. Illustrations of the low arity QP, QD, TP and H classes. Open circles are source vertices; black dots are subdivided vertices. The solid lines are the source mesh; the dashed lines are the subdivided mesh. The $(2,1)$ schemes have been included for completeness, although excluded by Heuristic 5

and his Theorem 2 proves the equivalent of this restriction to $(n,0)$ and (n,n) classes.

This heuristic seems reasonable because the base mesh is often constructed with important linear features of the object aligned with the mesh, so rotating away from this alignment is a bad thing. Of course, the (n,n) classes also rotate away from the desired alignment, but they do it symmetrically and, after two subdivision steps, they are realigned.

However, it is arguable that this heuristic is not strictly necessary. In particular, it is always possible to get the subdivision meshes to realign after every two subdivision steps by performing the rotation one way on even numbered steps and the opposite way on odd numbered steps (an example can be seen in Fig. 4). One way to check the validity of the heuristic would be to perform an investigation (similar to that undertaken for $TD(1,1)$ [28]) on either of the lowest arity classes which are excluded by this heuristic: $QP(2,1)$ or $QD(2,1)$. $QP(2,1)$ is specifically mentioned by Sloan [3] as useful in the context of numerical integration and Ivrissimtzis *et al.* [30] have recently undertaken an initial investigation of $QP(2,1)$ schemes. While they do produce a valid subdivision scheme, it is unclear whether it is of practical use.

Heuristic 6. *Triangular and quadrilateral schemes are generally useful but hexagonal schemes are more limited in their applications.*

As mentioned above, it is frequently useful to have important linear features in the model, such as edges, run along an edge in the base mesh in order to preserve the linear feature from one level of subdivision to the next. Hexagonal meshes do not have any straight edges which will run between multiple polygons. This would seem to limit the applicability of hexagonal schemes because they are not useful for objects in which such linear features need to be preserved. However, Claes *et al.* [14] claim that this is one of the advantages of hexagonal schemes: that they can be used situations where one does not want linear features to be preserved. Furthermore, hexagonal dual schemes are useful as the dual of triangular primal schemes [21].

Heuristic 7. *Low arity is preferable to high arity.*

Low arity has one key advantage over high arity: it provides a smaller increase in the number of vertices, which has the desirable effect of allowing for many levels of resolution close to one another. This is one of Kobbelt's [13] justifications for the usefulness of the $\sqrt{3}$ scheme.

Low arity is therefore important. The question then arises, what is the maximum arity that is worth considering. There seems to have been no serious investigation of any class with arity higher than three. For the purposes of this paper, I consider classes of arity less than four. Four is a somewhat arbitrary cut-off point and I make only one, weak, claim for it to be the cut-off, rather than any other value, which is that any arity two (binary) scheme also describes an arity four scheme by simply taking two subdivision steps of the arity two scheme. While an arity four scheme offers greater freedom than that offered by an arity two scheme in terms of choice of coefficients, it is unclear that there would be significant advantage in providing this greater freedom as it comes at the cost of reducing the number of levels of resolution available to the users.

Between arity three and arity four lie the T and H classes of arity $(2,2)$ $(\equiv \sqrt{12})$ and the Q classes of arity $(3,1)$ $(\equiv \sqrt{10})$ and $(3,2)$ $(\equiv \sqrt{13})$. The latter two classes would be excluded by Heuristic 5 but $TP(2,2)$ and $HD(2,2)$ would not be excluded by that heuristic and may be interesting as they are the lowest arity classes with mapping types vvv (triangular) and fff (hexagonal).

It is arguable that we should consider nothing higher than arity three; this would exclude the T and H classes of arity $(2,2)$ but not the $Q(2,2)$ classes $(\equiv \sqrt{8})$. As intimated the start of Sect. 3, it has been suggested that nothing higher than arity two is worth considering, which would exclude the ternary classes (arity $(3,0)$) as well the $Q(2,2)$ classes. Recent work [17, 18, 25] appears to contradict this extreme view and ternary classes certainly allow a range of different behaviour to that permitted by binary classes.

In contradiction of this estimate that arity four is some sort of rough cut-off point, consider the work of Maillot and Stam [31], who provide subdivision of arbitrary integer arity. Their work, however, simply does a single step of subdivision, of appropriate arity, to get from the base mesh to the final mesh, which is not quite in the spirit of subdivision.

Heuristic 8. *Interpolating schemes should be primal.*

All classes can accommodate approximating schemes. Any class with the $\mathsf{v} \rightarrow \mathsf{v}$ mapping can also accommodate interpolating schemes. Classes with the $\mathsf{v} \rightarrow \mathsf{f}$ mapping are also able to produce interpolating schemes but the derivations required are complicated and it is not clear that the advantages outweigh the complications.

4 Discussion

Taking all these heuristics into account, the arites which will most reward further investigation are $(1,1)$, $(2,0)$, $(3,0)$ and $(2,2)$, producing twelve subdivision classes (eight Q, four T) or eleven if we discount the $TP(2,2)$ class with the rather high arity $\sqrt{12}$. Including the equivalent H classes would add three or four classes to be considered (depending on whether or not one includes $HD(2,2)$). Table 5 lists the low arity classes classes, along with the name of the most well-known published schemes in each class. I have included the $(2,1)$ classes (excluded by Heuristic 5) for completeness because it may be that something useful could be done with them. Fig. 5 shows the layout of a single refinement step for each. Table 5 can be considered a much extended version of Zorin and Schröder's [20] Table 1. It is worth noting that, in addition to the schemes named in Table 5, Zorin and Schröder [20] have developed a whole family of $QP(2,0)$ and $QD(2,0)$ schemes and Oswald and Schröder [21] a whole family of $TP(1,1)$ and $HD(1,1)$ schemes, all based on up-sampling followed by repeated averaging.

The classification allows description of a wide range of possible subdivision schemes ranging from those which are currently used through those which may be useful to those which are almost certainly unusable. The heuristics are a mechanism for paring away the unusable classes in order to clearly identify the useful ones. While the classification system is a clean mathematical construct, the heuristics are less well-defined. The first four heuristics have strong justifications, but the latter four are open to contradiction as demonstrated in the discussion following each heuristic. Note that Ivrissimtzis *et al.* [2] implicitely assume the first two heuristics, while Han [26] assumes the first six. This paper indicates that, in contrast to both of those assumptions, the first *four* are reasonably straightforward to justify. One useful next step would be to ascertain whether there are formal mathematical proofs which either support or shatter each heuristic.

Table 5. The low arity classes which may be useful. They are listed in order of increasing arity within the four classifications QP, QD, TP, H. I have included some which are excluded by later heuristics and the right hand column shows which heuristics would cause them to be excluded. Each class is subdivided into approximating and interpolating sub-classes. Interpolating versions of dual schemes are difficult to construct (Heuristic 8) and have therefore been omitted. Sub-classes which have been investigated in the literature are given their common names and an appropriate citation. Those which have not, to my knowledge, been investigated are given a descriptive name in square brackets

		Example Schemes		
Class	vfe →	Approximating	Interpolating	Excluded by
$QP(1,1)$	vvf	Velho [11] P&S [10]	interpolating $\sqrt{2}$ [22, 23]	
$QP(2,0)$	vvv	Catmull-Clark [6]	Kobbelt [16]	
$QP(2,1)$	vfe	$\sqrt{5}$ [30]	[*interpolating* $\sqrt{5}$]	Heuristic 5
$QP(2,2)$	vvv	[$\sqrt{8}$]	[*interpolating* $\sqrt{8}$]	
$QP(3,0)$	vfe	[*ternary*]	[*interpolating ternary*]	
$QD(1,1)$	ffv	simplest [9]	—	
$QD(2,0)$	fff	Doo-Sabin [7]	—	
$QD(2,1)$	fve	[*dual* $\sqrt{5}$]	—	Heuristic 5
$QD(2,2)$	fff	[*dual* $\sqrt{8}$]	—	
$QD(3,0)$	fve	[*dual ternary*]	—	
$TP(1,1)$	vve	$\sqrt{3}$ [13]	interpolatory $\sqrt{3}$ [24]	
$TP(2,0)$	vfv	Loop [8]	butterfly [15]	
$TP(2,1)$	vfe	[$\sqrt{7}$]	[*interpolating* $\sqrt{7}$]	Heuristic 5
$TP(3,0)$	vve	Loop ternary [17]	interpolating ternary [18]	
$TP(2,2)$	vvv	[$\sqrt{12}$]	[*interpolating* $\sqrt{12}$]	Heuristic 7(?)
$HD(1,1)$	ffe	hexagon-by-three [14]	—	Heuristic 6
$HP(2,0)$	vff	hex binary [32](?)	[*interpolating hex binary*]	Heuristic 6
$HP(2,1)$	vfe	[*hex* $\sqrt{7}$]	[*interpolating hex* $\sqrt{7}$]	Heuristics 5 and 6
$HD(3,0)$	ffe	[*hex ternary*]	—	Heuristic 6
$HD(2,2)$	fff	[*hex dual* $\sqrt{12}$]	—	Heuristics 6 and 7(?)

In addition there are open questions pertaining to classes which are identified as useful by the heuristics but which have not yet been investigated:

- Is there any advantage to be gained from using a quadrilateral ternary ($Q(3,0)$) scheme? (c.f. Hassan's [25] univariate ternary scheme and the triangular ternary schemes investigated by Loop [17] and Dodgson *et al.* [18]).
- Is there any advantage in developing a $TP(2,2)$ scheme? $TP(2,2)$ is the lowest arity triangular class where all three element types map to vertices (i.e. it is of mapping type vvv). By contrast, the simplest quadrilateral class with this mapping is the thoroughly investigated $QP(2,0)$ class. However, even the simplest, useful $TP(2,2)$ scheme would require a vertex to have influence outside its 1-ring, making it difficult to extend to extraordinary cases, boundaries, and creases, so it may have little, if any, advantage.

- Are there useful interpolating $QP(1,1)$ and $TP(3,0)$ schemes? While Ivrissimtzis *et al.* [22, 23] have calculated appropriate mask coefficients for the $QP(1,1)$ class and Dodgson *et al.* [18] have undertaken initial work on $TP(3,0)$, it remains to perform detailed analysis and to modify the schemes to handle the extraordinary cases, boundaries, and creases.

It is possible to add further heuristics to the list relating to details further down the classification hierarchy (Sect. 1). As an example, the next heuristic which I would propose is the rather obvious:

Heuristic 9. *A small footprint is preferable.*

A smaller footprint makes for more efficient calculation and is easier to modify to handle the extraordinary cases. A larger footprint gives more freedom in choice of coefficients. Loop's motivation for investigating a ternary version ($TP(3,0)$) [17] of his binary scheme ($TP(2,0)$) [8] was that the ternary version gave more degrees of freedom. As a second example, the higher degree $QP(2,0)$ and $QD(2,0)$ schemes generated by Zorin and Schröder [20] have large footprints and clearly require more calculation than the lower degree schemes which seems to be a contra-indication. However, the mechanism of repeated averaging which they use provides a straightforward way of handling the extraordinary cases at the expense of losing the extra freedoms gained by having a larger footprint and at the expense of severe distortion around extraordinary points.

5 Conclusion

By applying heuristics to the classification, I conclude that the most useful linear, stationary subdivision classes have been investigated and schemes developed for them. There is some scope for further work, principally in looking at ternary subdivision [17, 18]. However the future development of new subdivision schemes seem to lie elsewhere, for example in the development of non-linear or non-stationary versions of schemes for classes which have already been investigated [33] or in combining schemes from more than one class into a single coherent mechanism [10, 34].

Acknowledgements

This work has been supported in part by the European Union, under the ægis of the MINGLE project (HPRN–CT–1999–00117). Thanks to Malcolm Sabin and Nira Dyn for interesting discussions, to the referees of the first Symposium on Geometry Processing who commented on an earlier version of this work and made many useful suggestions, and to the referees of this conference for their helpful comments.

References

1. Alexa, M.: Refinement operators for triangle meshes. Computer Aided Geometric Design **19** (2002) 169–172
2. Ivrissimtzis, I.P., Dodgson, N.A., Sabin, M.A.: A generative classification of mesh refinement rules with lattice transformations. Computer Aided Geometric Design **22** (2004) 99–109
3. Sloan, I.H., Joe, S.: Lattice methods for multiple integration. Oxford Science Publications. Oxford (1994)
4. Warren, J., Weimer, H.: Subdivision Methods for Geometric Design. Morgan Kaufmann (2001)
5. Martin, G.E.: Transformation Geometry. Springer-Verlag, New York (1982)
6. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design **10** (1978) 350–355
7. Doo, D., Sabin, M.: Behaviour of recursive division surfaces near extraordinary points. Computer-Aided Design **10** (1978) 356–360
8. Loop, C.T.: Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics (1987)
9. Peters, J., Reif, U.: The simplest subdivision scheme for smoothing polyhedra. ACM Transactions on Graphics **16** (1997) 420–431
10. Peters, J., Shiue, L.J.: Combining 4- and 3-direction subdivision. ACM Transactions on Graphics **23** (2004) 980–1003
11. Velho, L.: Quasi 4-8 subdivision. Computer Aided Geometric Design **18** (2001) 299–379
12. Velho, L., Zorin, D.: 4-8 subdivision. Computer Aided Geometric Design **18** (2001) 397–427
13. Kobbelt, L.: $\sqrt{3}$-Subdivision. In: SIGGRAPH 2000 Conference Proceedings. (2000) 103–112
14. Claes, J., Beets, K., van Reeth, F.: A corner-cutting scheme for hexagonal subdivision surfaces. In: Proceedings of Shape Modelling International. (2002) 13–20
15. Dyn, N., Levin, D., Gregory, J.A.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Transactions on Graphics **9** (1990) 160–169
16. Kobbelt, L.: Interpolatory subdivision on open quadrilateral nets with arbitrary topology. Computer Graphics Forum **15** (1996) 409–420
17. Loop, C.T.: Smooth ternary subdivision of triangular meshes. In Cohen, A., Merrien, J.L., Schumaker, L.L., eds.: Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press (2003) 295–302
18. Dodgson, N.A., Sabin, M.A., Barthe, L., Hassan, M.F.: Towards a ternary interpolating subdivision scheme for the triangular mesh. University of Cambridge Computer Laboratory Technical Report No. 539 (2002)
19. Sabin, M.A.: Eigenanalysis and artifacts of subdivision curves and surfaces. In Iske, A., Quak, E., Floater, M.S., eds.: Tutorials on Multiresolution in Geometric Modelling. Springer (2002) 69–92
20. Zorin, D., Schröder, P.: A unified framework for primal/dual quadrilateral subdivision schemes. Computer Aided Geometric Design **18** (2001) 429–454
21. Oswald, P., Schröder, P.: Composite primal/dual $\sqrt{3}$-subdivision schemes. Computer Aided Geometric Design **20** (2003) 135–164
22. Ivrissimtzis, I.P., Dodgson, N.A., Hassan, M.F., Sabin, M.A.: On the geometry of recursive subdivision. International Journal of Shape Modeling **8** (2002) 23–42

23. Ivrissimtzis, I.P., Dodgson, N.A., Sabin, M.A.: Recursive subdivision and hypergeometric functions. In: Proceedings of SMI2002: International Conference on Shape Modelling and Applications, IEEE Press (2002) 145–153
24. Labsik, U., Greiner, G.: Interpolatory $\sqrt{3}$-subdivision. Computer Graphics Forum **19** (2000) 131–138
25. Hassan, M.F., Ivrissimtzis, I.P., Dodgson, N.A., Sabin, M.A.: An interpolating 4-point C^2 ternary stationary subdivision scheme. Computer Aided Geometric Design **19** (2002) 1–18
26. Han, B.: Classification and construction of bivariate subdivision schemes. In Cohen, A., Merrien, J.L., Schumaker, L.L., eds.: Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press (2003) 187–197
27. Ostromoukhov, V., Donohue, C., Jodoin, P.M.: Fast hierarchical importance sampling with blue noise properties. ACM Transactions on Graphics **23** (2004) 488–495 Proc. SIGGRAPH 2004.
28. Dodgson, N.A., Ivrissimtzis, I.P., Sabin, M.A.: Characteristics of dual $\sqrt{3}$ subdivision schemes. In Cohen, A., Merrien, J.L., Schumaker, L.L., eds.: Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press (2003) 119–128
29. Niven, I.M.: Irrational Numbers. Carus Mathematical Monographs. Wiley, New York (1956)
30. Ivrissimtzis, I.P., Sabin, M.A., Dodgson, N.A.: $\sqrt{5}$ subdivision. In Dodgson, N.A., Floater, M.S., Sabin, M.A., eds.: Advances in Multiresolution for Geometric Modelling, Springer-Verlag (2005) 285–299
31. Maillot, J., Stam, J.: A unified subdivision scheme for polygonal modeling. Computer Graphics Forum **20** (2001) 471–479
32. Dyn, N., Levin, D., Simoens, J.: Face value subdivision schemes on triangulations by repeated averaging. In Cohen, A., Merrien, J.L., Schumaker, L.L., eds.: Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press (2003) 129–138
33. Morin, G., Warren, J., Weimer, H.: A subdivision scheme for surfaces of revolution. Computer Aided Geometric Design **18** (2001) 483–502
34. Stam, J., Loop, C.: Quad/triangle subdivision. Computer Graphics Forum **22** (2003) 79–85

A Details of the Formulæ in Tables 2–4

A.1 Quadrilateral Mesh

In the coordinate system of the subdivided mesh, vertices are at $(x, y),\ x, y \in \mathbb{Z}$, face centres at $\left(x + \frac{1}{2}, y + \frac{1}{2}\right),\ x, y \in \mathbb{Z}$, and mid-edges at $\left(x + \frac{1}{2}, y\right)$, $\left(x, y + \frac{1}{2}\right)$, $x, y \in \mathbb{Z}$.

For the primal classes, $QP(n, m)$, the origin of the source grid is a source vertex at $(0, 0)$, with an adjacent source vertex at $(n, m),\ n, m \in \mathbb{Z},\ 0 < n,\ 0 \leq m \leq n$.

A source quadrilateral adjacent to the origin has vertices at $(0, 0)$, (n, m), $(-m, n)$, and $(n - m, n + m)$. Its face centre is at the arithmetic mean of these four points: $\left(\frac{n-m}{2}, \frac{n+m}{2}\right)$. This coincides with a vertex of the subdivided mesh if $n - m \bmod 2 = 0$. If the alternative, $n - m \bmod 2 = 1$, is true then a face centre maps to a face centre.

The source edge from $(0,0)$ to (n,m) has its midpoint at $\left(\frac{n}{2}, \frac{m}{2}\right)$. Therefore, if $n \bmod 2 = m \bmod 2 = 0$ we have $\mathsf{e} \to \mathsf{v}$, if $n \bmod 2 = m \bmod 2 = 1$, we have $\mathsf{e} \to \mathsf{f}$, and otherwise we have $\mathsf{e} \to \mathsf{e}$.

For the dual classes, $QD(n,m)$, everything shifts by $\left(\frac{1}{2}, \frac{1}{2}\right)$. The net result is that we can simply exchange the rôles of face centres and vertices in subdivided mesh in the $QP(n,m)$ case. Thus, $n-m \bmod 2 = 0 \Rightarrow \mathsf{f} \to \mathsf{f}$ and $n-m \bmod 2 = 1 \Rightarrow \mathsf{f} \to \mathsf{v}$ for the QD case and, likewise, $n \bmod 2 = m \bmod 2 = 0 \Rightarrow \mathsf{e} \to \mathsf{f}$; $n \bmod 2 = m \bmod 2 = 1 \Rightarrow \mathsf{e} \to \mathsf{v}$; otherwise $\mathsf{e} \to \mathsf{e}$.

A.2 Triangular Mesh

In the coordinate system of the subdivided mesh, vertices are at (x,y), $x,y \in \mathbb{Z}$, face centres at $\left(x+\frac{1}{3}, y+\frac{1}{3}\right)$, $\left(x+\frac{2}{3}, y+\frac{2}{3}\right)$, $x,y \in \mathbb{Z}$, and mid-edges at $\left(x+\frac{1}{2}, y\right)$, $\left(x, y+\frac{1}{2}\right)$, $\left(x+\frac{1}{2}, y+\frac{1}{2}\right)$, $x,y \in \mathbb{Z}$. Note that there are *two* types of face centre: the centres of up-pointing triangles ($\triangle$) and the centres of down-pointing triangles ($\bigtriangledown$). The ramifications of this are discussed in detail by Ivrissimtzis *et al.* [2]. We will annotate the f notation with a subscript, $\mathsf{f}_\triangle$ and $\mathsf{f}_\bigtriangledown$, where necessary.

For a $TP(n,m)$ class, without loss of generality, we will take the origin of the source grid to be a source vertex at $(0,0)$, with an adjacent source vertex at (n,m), $n,m \in \mathbb{Z}$, $0 < n$, $0 \le m \le n$, and with an up-pointing triangle to the left of the line as one moves from $(0,0)$ to (n,m).

The up-pointing source triangle to the left of this line has source vertices at $(0,0)$, (n,m) and $(-m, n+m)$. The face centre of this source triangle is at the arithmetic mean of these three points: $\left(\frac{n-m}{3}, \frac{n+2m}{3}\right)$. Thus we have three possible mappings:

$$\begin{aligned}
n+2m \bmod 3 = 0 &\Rightarrow \mathsf{f}_\triangle \to \mathsf{v} \;\; \mathsf{f}_\bigtriangledown \to \mathsf{v}\\
n+2m \bmod 3 = 1 &\Rightarrow \mathsf{f}_\triangle \to \mathsf{f}_\triangle \; \mathsf{f}_\bigtriangledown \to \mathsf{f}_\bigtriangledown\\
n+2m \bmod 3 = 2 &\Rightarrow \mathsf{f}_\triangle \to \mathsf{f}_\bigtriangledown \; \mathsf{f}_\bigtriangledown \to \mathsf{f}_\triangle
\end{aligned}$$

It is not clear that there is a need to distinguish between up- and down-pointing triangles and so, in the interests of clarity, Table 3 does not do so. The reader will note, however, that the most widely used triangular schemes (the $TP(2,0)$ schemes Loop [8] and butterfly [15]) map up-pointing triangles to down-pointing triangles and vice-versa.

The source edge from $(0,0)$ to (n,m) has its midpoint at $\left(\frac{n}{2}, \frac{m}{2}\right)$. Therefore, if $n \bmod 2 = m \bmod 2 = 0$ we have $\mathsf{e} \to \mathsf{v}$. In all other cases, $\mathsf{e} \to \mathsf{e}$.

For the $TD(n,m)$ classes, the origin of the source grid is a source vertex at the centre of a face. Its coordinates will thus be: $\left(\frac{c}{3}, \frac{c}{3}\right)$ $c \in \{1,2\}$ where $c = 1$ if the face is an up-pointing triangle and $c = 2$ if the face is a down-pointing triangle. Ivrissimtzis *et al.* [2] show that $n,m \in \mathbb{Z}$ in the TD case.

The up-pointing source triangle to the left of the line from the origin to the adjacent source vertex, $\left(n+\frac{c}{3}, m+\frac{c}{3}\right)$, has vertices at $\left(\frac{c}{3}, \frac{c}{3}\right)$, $\left(n+\frac{c}{3}, m+\frac{c}{3}\right)$ and $\left(-m+\frac{c}{3}, n+m+\frac{c}{3}\right)$. The face centre of this source triangle is at the arithmetic mean of these three points: $\left(\frac{n-m}{3}+\frac{c}{3}, \frac{n+2m}{3}+\frac{c}{3}\right)$. Thus we have three possible mappings for each of the values of c. For $c = 1$:

$$\begin{aligned} n+2m \bmod 3 = 0 &\Rightarrow \mathsf{f}_\triangle \rightarrow \mathsf{f}_\triangle \; \mathsf{f}_\bigtriangledown \rightarrow \mathsf{f}_\triangle \\ n+2m \bmod 3 = 1 &\Rightarrow \mathsf{f}_\triangle \rightarrow \mathsf{f}_\bigtriangledown \; \mathsf{f}_\bigtriangledown \rightarrow \mathsf{v} \\ n+2m \bmod 3 = 2 &\Rightarrow \mathsf{f}_\triangle \rightarrow \mathsf{v} \;\; \mathsf{f}_\bigtriangledown \rightarrow \mathsf{f}_\bigtriangledown \end{aligned}$$

For $c = 2$:

$$\begin{aligned} n+2m \bmod 3 = 0 &\Rightarrow \mathsf{f}_\triangle \rightarrow \mathsf{f}_\bigtriangledown \; \mathsf{f}_\bigtriangledown \rightarrow \mathsf{f}_\bigtriangledown \\ n+2m \bmod 3 = 1 &\Rightarrow \mathsf{f}_\triangle \rightarrow \mathsf{v} \;\; \mathsf{f}_\bigtriangledown \rightarrow \mathsf{f}_\triangle \\ n+2m \bmod 3 = 2 &\Rightarrow \mathsf{f}_\triangle \rightarrow \mathsf{f}_\triangle \; \mathsf{f}_\bigtriangledown \rightarrow \mathsf{v} \end{aligned}$$

In the TD cases, unless $n + 2m \bmod 3 = 0$, then half of the face centres map to face centres and half map to vertices, which is forbidden by Heuristic 3. However, the situation is rather messy as Heuristic 3 excludes only some of the TD classes, providing further evidence that there are deeper things going on than revealed by the simple classification into 'primal' and 'dual'.

The edge from $\left(\frac{c}{3}, \frac{c}{3}\right)$ to $\left(n + \frac{c}{3}, m + \frac{c}{3}\right)$ has its midpoint at $\left(\frac{n}{2} + \frac{c}{3}, \frac{m}{2} + \frac{c}{3}\right)$. Therefore, if $n \bmod 2 = m \bmod 2 = 0$ we have $\mathsf{e} \rightarrow \mathsf{f}$. In all other cases, $\mathsf{e} \rightarrow \mathsf{x}$, i.e. an edge maps either to a face centre or it maps to no element at all.

Only if both $n + 2m \bmod 3 = 0$ and $n \bmod 2 = m \bmod 2 = 0$ do we get a sensible mapping. Combining these two gives the condition $n + 2m \bmod 6 = 0$ which is mentioned in the discussion of Heuristic 3.

A.3 Hexagonal Mesh

The hexagonal case is somewhat more involved than the triangular case because, in the hexagonal case, we can distinguish two different types of vertex. This means that we must check that both types of vertex map to the same new element type (face or vertex) in order for the class to be either HD or HP. Otherwise, the class is HM.

In the coordinate system of the subdivided mesh, face centres are at (x, y), $x, y \in \mathbb{Z}$, vertices at $\left(x + \frac{1}{3}, y + \frac{1}{3}\right)$, $\left(x + \frac{2}{3}, y + \frac{2}{3}\right)$, $x, y \in \mathbb{Z}$, and mid-edges at $\left(x + \frac{1}{2}, y\right)$, $\left(x, y + \frac{1}{2}\right)$, $\left(x + \frac{1}{2}, y + \frac{1}{2}\right)$, $x, y \in \mathbb{Z}$. We need to annotate the v notation in order to distinguish the two types of vertex. Where necessary, vertices at $\left(x + \frac{1}{3}, y + \frac{1}{3}\right)$, $x, y \in \mathbb{Z}$ will be denoted $\mathsf{v}_\triangle$ and those at $\left(x + \frac{2}{3}, y + \frac{2}{3}\right)$, $x, y \in \mathbb{Z}$, $\mathsf{v}_\bigtriangledown$. $\mathsf{v}_\triangle$ is a Y-shaped vertex while $\mathsf{v}_\bigtriangledown$ is an inverted Y. The orientation of the triangle is the dual of the configuration of the vertex.

In the hexagonal case, the (n, m) notation does not refer to the distance between two adjacent vertices but between two vertices of the same type or, equivalently, between two face centres. This ensures that the hexagonal cases with classification (n, m) are duals of the triangular cases with classification (n, m).

For an $HD(n, m)$ class, without loss of generality, we will take the origin of the source grid to be a source vertex at $(0, 0)$, with the next source vertex of the same type at (n, m), $n, m \in \mathbb{Z}$, $0 < n$, $0 \leq m \leq n$, and with the vertex at the origin being of type $\mathsf{v}_\bigtriangledown$.

The hexagon has source vertices of type $\mathsf{v}_\bigtriangledown$ at $(0, 0)$, (n, m), and $(-m, n+m)$, with intervening vertices of type $\mathsf{v}_\triangle$ at $\left(\frac{2n+m}{3}, \frac{-n+m}{3}\right)$, $\left(\frac{2n-2m}{3}, \frac{2n+4m}{3}\right)$, and $\left(\frac{-n-2m}{3}, \frac{2n+m}{3}\right)$. The face centre of this source hexagon is at the arithmetic mean of these six points: $\left(\frac{n-m}{3}, \frac{n+2m}{3}\right)$. From these, we can determine that $\mathsf{v}_\bigtriangledown \to \mathsf{f}$ always (by definition) and that:

$$\begin{array}{l}
n+2m \bmod 3 = 0 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{f} \quad \mathsf{f} \to \mathsf{f} \\
n+2m \bmod 3 = 1 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{v}_\bigtriangledown\ \mathsf{f} \to \mathsf{v}_\bigtriangleup \\
n+2m \bmod 3 = 2 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{v}_\bigtriangleup\ \mathsf{f} \to \mathsf{v}_\bigtriangledown
\end{array}$$

Thus, if $n+2m \bmod 3 \neq 0$, we do not have an HD class because vertices of type $\mathsf{v}_\bigtriangleup$ do not map to face centres, and therefore we have an HM class. Thus, for all HD classes, $n+2m \bmod 3 = 0$, by definition, and $\mathsf{f} \to \mathsf{f}$.

Analysis of the edges show that there are only two possible edge mappings. If $n \bmod 2 = m \bmod 2 = 0$ we have $\mathsf{e} \to \mathsf{f}$. In all other cases, $\mathsf{e} \to \mathsf{e}$.

For the $HP(n,m)$ classes let us take, as the origin of the source grid, a source vertex of type $\mathsf{v}_\bigtriangledown$ at $\left(\frac{c}{3}, \frac{c}{3}\right)$ $c \in \{1,2\}$ where the value of c determines the type of destination vertex ($\mathsf{v}_\bigtriangleup$ or $\mathsf{v}_\bigtriangledown$). The next source vertex of type $\mathsf{v}_\bigtriangledown$ is at $\left(n+\frac{c}{3}, m+\frac{c}{3}\right)$.

The hexagon has source vertices of type $\mathsf{v}_\bigtriangledown$ at $\left(\frac{c}{3}, \frac{c}{3}\right)$, $\left(n+\frac{c}{3}, m+\frac{c}{3}\right)$, and $\left(-m+\frac{c}{3}, n+m+\frac{c}{3}\right)$, with intervening vertices of type $\mathsf{v}_\bigtriangleup$ at $\left(\frac{2n+m+c}{3}, \frac{-n+m+c}{3}\right)$, $\left(\frac{2n-2m+c}{3}, \frac{2n+4m+c}{3}\right)$, and $\left(\frac{-n-2m+c}{3}, \frac{2n+m+c}{3}\right)$. The face centre of this source hexagon is at the arithmetic mean of these six points: $\left(\frac{n-m+c}{3}, \frac{n+2m+c}{3}\right)$. By definition, if $c=1$ then $\mathsf{v}_\bigtriangledown \to \mathsf{v}_\bigtriangleup$ and if $c=2$ then $\mathsf{v}_\bigtriangledown \to \mathsf{v}_\bigtriangledown$. We need to consider the mappings for $\mathsf{v}_\bigtriangleup$ and f for each value of c.

$$\begin{array}{l}
\text{For } c = 1: \\
n+2m \bmod 3 = 0 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{v}_\bigtriangleup\ \mathsf{f} \to \mathsf{v}_\bigtriangleup \\
n+2m \bmod 3 = 1 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{f} \quad \mathsf{f} \to \mathsf{v}_\bigtriangledown \\
n+2m \bmod 3 = 2 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{v}_\bigtriangledown\ \mathsf{f} \to \mathsf{f}
\end{array}$$

$$\begin{array}{l}
\text{For } c = 2: \\
n+2m \bmod 3 = 0 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{v}_\bigtriangledown\ \mathsf{f} \to \mathsf{v}_\bigtriangledown \\
n+2m \bmod 3 = 1 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{v}_\bigtriangleup\ \mathsf{f} \to \mathsf{f} \\
n+2m \bmod 3 = 2 \Rightarrow \mathsf{v}_\bigtriangleup \to \mathsf{f} \quad \mathsf{f} \to \mathsf{v}_\bigtriangleup
\end{array}$$

Thus, we have HM classes if $n+2m \bmod 3 = c$ because, in these cases, $\mathsf{v}_\bigtriangledown \to \mathsf{v}$ but $\mathsf{v}_\bigtriangleup \to \mathsf{f}$. For HP schemes we can summarise our results as:

$$\begin{array}{ll}
n+2m \bmod 3 = 0 & \Rightarrow \mathsf{v} \to \mathsf{v}\ \mathsf{f} \to \mathsf{v} \\
n+2m \bmod 3 = 3-c & \Rightarrow \mathsf{v} \to \mathsf{v}\ \mathsf{f} \to \mathsf{f} \\
n+2m \bmod 3 = c & \Rightarrow \text{HM not HP}
\end{array}$$

It now remains to determine the edge mappings. There are three types of edge to consider, which can be characterised by one example of each. These are the first three edges round the source hexagon starting at the origin vertex and they are at:

$$\left(\frac{2n+m}{6}+\frac{c}{3}, \frac{-n+m}{6}+\frac{c}{3}\right), \left(\frac{5n+m}{6}+\frac{c}{3}, \frac{-n+4m}{6}+\frac{c}{3}\right), \left(\frac{5n-2m}{6}+\frac{c}{3}, \frac{2n+7m}{6}+\frac{c}{3}\right).$$

We thus need to know the values of n and m which place these coordinates at destinations vertices, face centres or edges, which means that we need to consider the values of $2n + m \bmod 6$, $5n + m \bmod 6$ and $5n + 4m \bmod 6$. Some basic analysis of these shows that the following results hold:

$$\begin{array}{lll}
(n-m) \bmod 3 = 0 & \text{and} & \\
n \bmod 2 = m \bmod 2 = 0 & \Rightarrow & \mathsf{e} \rightarrow \mathsf{v} \\
\text{otherwise} & \Rightarrow & \mathsf{e} \rightarrow \mathsf{x}
\end{array}$$

$$\begin{array}{lll}
(n-m) \bmod 3 = 3 - c & \text{and} & \\
n \bmod 2 = m \bmod 2 = 0 & \Rightarrow & \mathsf{e} \rightarrow \mathsf{f} \\
\text{otherwise} & \Rightarrow & \mathsf{e} \rightarrow \mathsf{e}
\end{array}$$

Global Curve Analysis via a Dimensionality Lifting Scheme

Gershon Elber

Technion, Israel Institute of Technology, Haifa 32000, Israel
gershon@cs.technion.ac.il

Abstract. Freeform rational parametric curves and surfaces have been playing a major role in computer aided design for several decades. The ability to analyze local (differential) properties of parametric curves is well established and extensively exploited. In this work, we explore a different lifting approach to global analysis of freeform geometry, mostly curves, in $\mathbb{R}^2$ and $\mathbb{R}^3$. In this lifting scheme, we promote the problem into a higher dimension, where we find that in the higher dimension, the solution is simplified.

1 Introduction

The differential analysis of freeform planar and 3-space parametric curves is a fundamental tool that is heavily used in computer aided geometric design applications. Numerous examples exist. Being able to handle and define curvature properties eases the understanding of singularities in offset curves, curves that are crucial to many design and manufacturing applications. Having the ability to locally define a wide variety of orientation frames along curves, from the Frenet [6] frame to orientation minimizing frames [4, 17] using local differential geometry, constitutes an immense aid in the construction of sweep surfaces.

The exploited differential analysis is indeed mostly local. Analysis of the global properties of curves (and surfaces) in $\mathbb{R}^2$ and $\mathbb{R}^3$ is far less common in geometric design. Clearly, global integrable properties of freeform geometry are much more difficult to detect. Some examples of work that has tried to derive global properties are found in [14] where moments of freeform geometry are developed. In [9, 15, 18], the area (volume) enclosed by a curve (surface) is made fixed while interactive (multi-resolution) direct manipulation is allowed. In the present work, we consider a different approach to global analysis of freeform geometry. By lifting the problem into a higher dimensional space, the hope is that the lifted geometry will, in fact, be simpler to process.

We intend to explore several problems about which very little is known. Yet and as an example, we take as our starting point the simple and already solved problem of finding all the inflection points of a freeform regular planar curve, $C(u)$. This problem is closely related to the issue at hand and could easily be reduced to finding the zeros of the univariate of $C''(u) \times C'(u)$, thereby identifying all the locations with zero curvature in the curve. Nonetheless, in Section 2,

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 184–200, 2005.

we will use this example to show a new constructor – an *orthogonality map* – that lifts the univariate curve $C(u)$ into a bivariate surface. With this map, the identification of all the inflection points is simple. Further, other (global) properties of planar curves such as visibility properties and their winding numbers will also be revealed with the aid of the orthogonality map.

Visibility and moldability are two other examples where global analysis must play a role. Visibility queries typically ask questions regarding lines-of-sight between points and objects whereas the moldability question looks at partitioning the given geometry into parts of a mold that can be assembled and disassembled without the parts colliding. Interestingly enough, these two problems are closely related. Consider the planar n-moldability problem of $C(u)$. That is, the computation of the decomposition of $C(u)$ into an n-piece mold, if possible, minimizing n. While some work on piecewise linear representations (i.e., polygons) can be found, virtually nothing is known about this problem and its solutions in the context of freeform geometry. In [1], the two-piece mold separability problem for polygonal surfaces in $I\!R^3$ is considered, with a heuristic implemented solution. View and inspection planning is another problem that was investigated in this piecewise linear context [19].

In the freeform domain, little information exists about visibility and moldability. A previous paper by Elber [12] considered the problem of determining the existence of a valid two-piece mold for a designed solid model whose boundary is represented as a NURBS surface with C^3 continuity, and finding such a mold if it exists. For more than two pieces, nothing is known. Moreover, related visibility problems such as the art gallery query, a well studied problem in the computational geometry community, again in the discrete polygonal domain [3], is another open question. In Section 3, we will address these questions using a different dimensionality lifting scheme.

Another problem we will be exploring in Section 4 is, given a simple 3-space curve, $C(u)$, find all directions V, if any, in $I\!R^3$ from which $C(u)$'s orthographic projection onto the plane orthogonal to V is simple. Finally, in Section 5, we conclude.

2 The Orthogonality Map

Consider a C^1 regular planar parametric curve $C(u)$:

Definition 1. *The* Orthogonality Function $\mathcal{O}_{\mathcal{F}}(u,v)$ *of a* C^1 *regular planar parametric curve* $C(u) : \mathcal{D} \in I\!R \rightarrow I\!R^2$ *is defined as*

$$\mathcal{O}_{\mathcal{F}}(u,v) = \langle C'(u), C'(v) \rangle : [\mathcal{D} \times \mathcal{D}] \rightarrow I\!R.$$

and

Definition 2. *The* Orthogonality Map $\mathcal{O}_{\mathcal{M}}$ *of a* C^1 *regular planar parametric curve* $C(u)$*, denoted* $\mathcal{O}_{\mathcal{M}}[C]$*, is the zero set of the orthogonality function,* $\mathcal{O}_{\mathcal{F}}(u,v) = 0$*.*

The orthogonality function lifts curve $C(u)$ into an explicit surface whose zero set is defined to be the curve's orthogonality map. Let us explore a few properties of the orthogonality map.

Lemma 1. *$\mathcal{O}_{\mathcal{F}}(u,v)$ and $\mathcal{O}_{\mathcal{M}}[C]$ are both symmetric along the $u = v$ diagonal for a closed C^1 curve C.*

Proof. Trivial by construction, $\mathcal{O}_{\mathcal{M}}(u,v) = \mathcal{O}_{\mathcal{M}}(v,u)$. ■

The fact that the $\mathcal{O}_{\mathcal{M}}$ map is symmetric will repeat itself in the other lifting schemes we will examine. Since $\mathcal{O}_{\mathcal{F}}(u,v)$ is a continuous function, all curves in $\mathcal{O}_{\mathcal{M}}[C]$ are either closed loops or curves that start and end on the boundary of $[\mathcal{D} \times \mathcal{D}]$.

Recall that we sought, as a simple exemplary motivation, to derive all the inflection points of the planar curve $C(u)$ using $\mathcal{O}_{\mathcal{M}}$. The following result explains how this could be accomplished:

Lemma 2. *The u-extreme (v-extreme) locations in $\mathcal{O}_{\mathcal{M}}[C]$ correspond to v-inflection (u-inflection) points in C.*

Proof. The extreme locations of the subset of $\mathbb{R}^2$ of $\mathcal{O}_{\mathcal{F}}(u,v) = 0$ (the $\mathcal{O}_{\mathcal{M}}$ set) are classified by the additional constraints of $\frac{\partial \mathcal{O}_{\mathcal{F}}(u,v)}{\partial v} = 0$ $\left(\frac{\partial \mathcal{O}_{\mathcal{F}}(u,v)}{\partial u} = 0\right)$ [16]. These are the locations where the $\mathcal{O}_{\mathcal{M}}$ curve has u-extreme (v-extreme) locations.

Differentiating $\mathcal{O}_{\mathcal{M}}(u,v)$ with respect to u (respectively v),

$$\frac{\partial \mathcal{O}_{\mathcal{F}}(u,v)}{\partial u} = \frac{\partial \langle C'(u), C'(v)\rangle}{\partial u} = \langle C''(u), C'(v)\rangle = C''(u) \times C'(u),$$

due to the fact that $C'(v)$ is orthogonal to $C'(u)$, satisfying $\mathcal{O}_{\mathcal{F}}(u,v) = 0$ (similarly for the case of v).

Nevertheless, the condition $C''(u) \times C'(u) = 0$ is exactly related to the zero curvature locations, hence detecting the inflection points. ■

Figure 1 presents an example of a planar curve (Figure 1 (a)) and its orthogonality map (Figure 1 (b)). In Figure 1 (c), the orthogonality function is shown in $\mathbb{R}^3$ along with its zero set, the orthogonality map. Two inflection points are presented in Figure 1 (a) and these points are clearly reflected as extreme locations in the orthogonality map shown in Figure 1 (b). Note the two pairs of lines (one horizontal pair and one vertical pair) in Figure 1 (b) at the inflection points' parameter values go through all the extreme locations, capturing the loops in the map as well.

Definition 3. *Consider a sub-region of $C(v)$, $v_1 < v < v_2$. $C(v)$, $v_1 < v < v_2$ is considered* locally visible *from some view direction V, if no point $C(v_0)$, $v_1 < v_0 < v_2$ obscures another point in that domain, when viewed from V.*

Put differently, the local visibility only considers the possibility of a small region of $C(u)$ to obscure itself, and ignores the possibility that other parts of

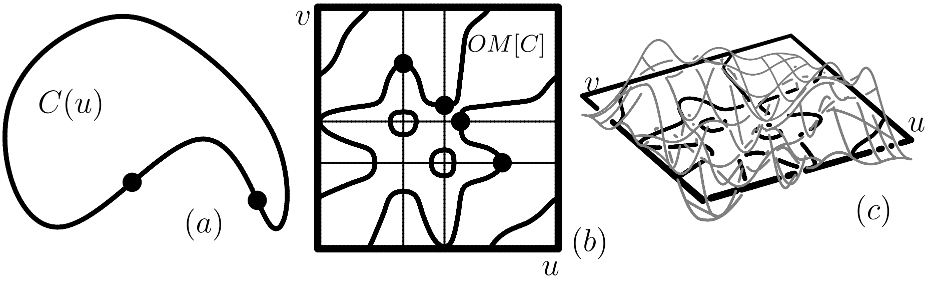


Fig. 1. An orthogonality map $OM[C]$ (b) of a planar curve $C(u)$ (a). (c) presents the orthogonality function (in gray) along with its zero set (in black), the orthogonality map

$C(u)$, outside the local domain, obscure these points. Consider now the viewing direction of $V = N(u_0)$, following the normal of C at u_0. Inspect the point (u_0, u_0) on the diagonal of $\mathcal{O}_\mathcal{M}[C]$ and shoot two opposing rays vertically, in the $-v$ and $+v$ directions, from (u_0, u_0) (see the vertical light-gray edge in Figure 2 (c)). Let the points of impact of these vertical rays with $\mathcal{O}_\mathcal{M}[C]$ be (u_0, v_1) and (u_0, v_2) and denote these points as the *vertically closest* points to (u_0, u_0).

Lemma 3. *Let (u_0, v_1) and (u_0, v_2) be the two vertically closest points on $\mathcal{O}_\mathcal{M}[C]$ to (u_0, u_0), for some u_0. Then, the region of C from $C(v_1)$ to $C(v_2)$ is locally visible from direction $V = N(u_0)$.*

Proof. By the orthogonality map, every point $C(v)$, $v_1 < v < v_2$ possesses a tangent vector that is never orthogonal to $C'(u_0)$ and hence is never parallel to V. Consequently, starting from $C(u_0)$, when we move along the curve toward $C(v_1)$ (or toward $C(v_2)$), we never locally occlude previous curve points in the domain. In other words, none of the curve points $C(v)$, $v_1 < v < v_2$, could possibly occlude each other and, therefore, they are all locally visible. ■

Figure 2 is an example of this phenomenon. In Figure 2 (a), a planar curve is presented with the point $C(u_0)$ for which the local visibility is sought when viewed from $N(u_0)$ (see Figure 2 (d)). With the aid of the orthogonality map in Figure 2 (b), the local visibility of $C(u_0)$ could be determined, as shown in Figure 2 (c) in light-gray.

One can build the lower, $L(u)$, and upper, $U(u)$, envelope of these local visibilities for all u locations in the domain of $C(u)$. These envelopes show, for all u, the maximal extent along the curve that is locally visible from $N(u)$ – the normal direction at $C(u)$. These lower and upper envelopes of the local visibility, as was determined for all u in the domain, are plotted in Figure 2 (c), in dark-gray. If the geometry is closed and C^1 continuous, the map is periodic in both the u and v axes and hence the envelopes penetrate above (below) the square domain with the semantics that these portions are warped to the bottom (top) of the domain.

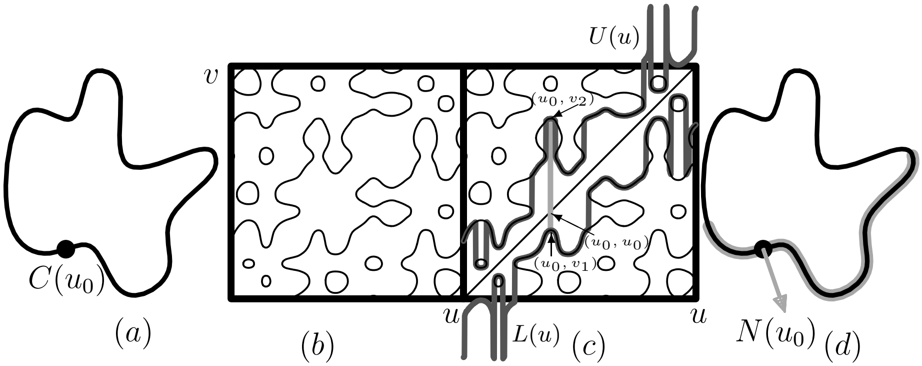


Fig. 2. The orthogonality map (b) of planar curve (a) is used to compute the local visibility (light-gray in (c) & (d)) from direction $N(u_0)$. Over all u, the lower, $L(u)$, and upper, $U(u)$, visibility envelopes are constructed and shown in (c) in dark-gray

$L(u)$ and $U(u)$ could aid in selecting a minimal set of views from which the entire curve is (locally) accessible or visible. More on this will be found in Section 3.

The orthogonality map comprises two types of entities. The first are islands that are the result of having two inflection points in the curve with tangents that are (almost) orthogonal. In Figure 1 (b), we have one such island (note we consider only one half of this symmetric function). This island corresponds to the two inflection locations marked on Figure 1 (a).

The second type of entity is composed of components that start at the bottom boundary and end at the right boundary (considering only the portion of the orthogonality map below the diagonal). Note that both islands could cross boundaries if the curve is periodic, and components that start at the bottom boundary and end at the right boundary could also traverse back and forth across these boundaries. Now, consider this periodic map, for periodic curves, as a repeated square that tiles $\mathbb{R}^2$, while denoting the original map as the *primary tile* (see Figure 3).

Let $\mathcal{W}$ denote the *winding number* of curve $C(u)$, i.e., the number of times the curve turns in the plane. $\mathcal{W}$ could be computed by integrating the curvature of the curve, yet we postulate that this global number could also be extracted from the orthogonality map. Count the number of paths from the bottom boundary through the right boundary. Then, we have the following result:

Observation 1. *The number of infinite paths from the bottom boundary to the right boundary of the orthogonality map equals* $2\mathcal{W}$.

To justify this observation, consider $C(u)$ to be a closed, simple, continuous convex curve (see Figure 4 (a) and consider an infinite line in the direction of $D = C'(u_0)$ approaching from infinity in the $N(u_0)$ direction and toward $C(u)$, $N(u_0)$ being the normal of C at u. Stop the approach at the first point of contact

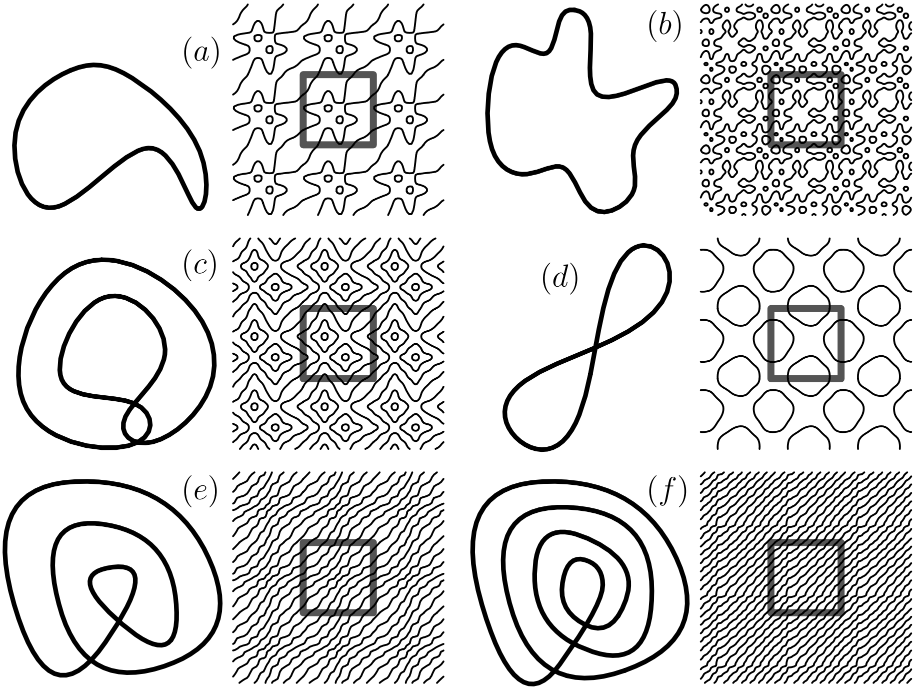


Fig. 3. The winding number of a planar curve is equal to half the number of infinite paths going through the bottom boundary to the right boundary of the primary tile (in gray) of the orthogonality map

with $C(u)$, and repeat this line-approaching process from the $-N(u_0)$ direction. The distance between these two infinite parallel lines, which are now in contact with $C(u)$ on both its sides, determines the diameter of the convex curve from that direction (which cannot be zero). Hence, a simple closed convex curve has a pair of solution points on the orthogonality map, for every location on the curve. $C(u)$ is C^1 continuous and due to this continuity these pairs will combine and form two continuous paths on the orthogonality map from the bottom boundary to the right boundary. For a convex curve, these paths will also be monotone.

The convex hull curve of every planar simple curve will, therefore, have two paths. The convex hull of a planar simple curve $C(u)$ consists of convex regions of $C(u)$ and line segments connecting these regions (see Figure 4 (b)). Nonetheless, the line segments are bitangent lines to $C(u)$ that have the same tangent direction at the two end points of each line segment, denoted $C(u_s)$ and $C(u_e)$. A path in the orthogonal map that matches another point $C(u_m)$ as an orthogonal point to $C(u_s)$ must be connected to $C(u_e)$ as well, because the curve segment $C(u)$, $u \in [u_s, u_e]$ is C^1. In other words, the two paths will still form, for a simple curve, though they will no longer be monotone.

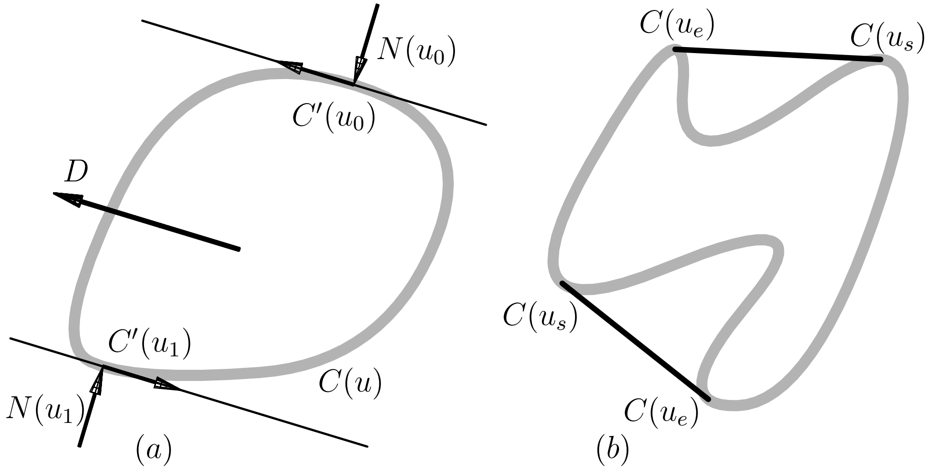


Fig. 4. Given some direction $D = C'(u_0)$ in (a), two inifinite lines in direction D approach convex planar curve $C(u)$ from $\pm N(u_0) = \mp N(u_1)$ and touch it in two locations u_0 and u_1. In (b), a general curve is complemented with bitangents, making it convex

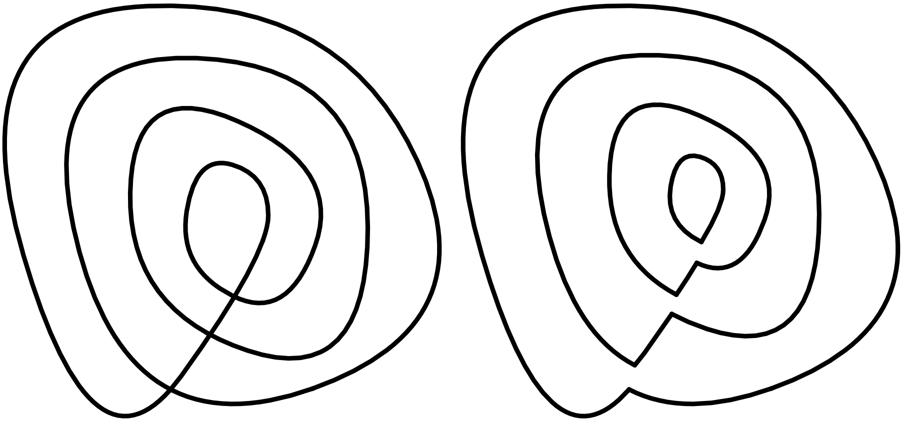


Fig. 5. A decomposition of a curve with four loops (a) into four simple nested curves (b). See also Figure 3 (f)

Now consider m nested, closed, simple C^1 continuous convex curves. Clearly, every such simple curve contributes two paths or $2m$ paths in the orthogonality map in all. Nevertheless, every closed curve could be broken into several closed and simple curves, while preserving the orientation of the loops to follow that of the original curve. For example, Figure 5 shows a decomposition of the curve in Figure 3 (f) into four simple and nested curves.

The two curves in Figures 3 (a) and (b) are simple and the primary tile indeed has two paths from the bottom boundary to the right boundary. The self intersecting shape in Figure 3 (c) is decomposed into three simple loops, but one of them is in the opposite orientation to the other two and hence the two paths from the bottom boundary to the right boundary of the primary tile prescribe the correct winding number. The self intersecting '8' shape in Figure 3 (d) is decomposed into two simple curves but in an opposite orientation, which leaves the winding number of this curve at zero. Finally, the two shapes in Figures 3 (e) and (f) are decomposed into three and four simple nested curves, all with the same orientation and as expected, six and eight paths are recognized in their primary tiles, respectively.

Given an orthogonality map, one needs to trace all the closed loops in the map and purge them away so as to find and count only the continuous paths that originate at the bottom boundary and end at the right one. This tracing process is trivial as loops are detected by tracing until one ends up at the starting location. The other tracing alternative is to start at the right boundary and then the curve, if not a loop, must end up at the bottom boundary. In recent years, the recovery of the topology of implicit forms, as is $\mathcal{O}_{\mathcal{M}}[C]$, has also been investigated. These recovery methods, for example [16], could be used to recover the winding number of $C(u)$ without explicitly tracing the entire orthogonality map.

3 The Visibility Question

Going beyond the orthogonality maps, we would like to consider global visibility as well. This means that, for every point of inspection, one needs to resolve the (view) point-curve visibility problem. The visibility could change at the silhouette curves' locations, $\mathcal{S}$ – locations where the view direction V is tangent to the curve $C(u)$ or $\mathcal{S} = \{u \mid \langle V, N(u) \rangle = 0\}$, where $N(u)$ is the normal field of C. We seek to split $C(u)$ at all locations where the quantitative invisibility [2, 7] changes. The original quantitative invisibility [2] algorithm starts by splitting all polygonal edges with inhomogeneous visibility in the projection plane into homogeneous segments. That is, if a polygonal edge is partially visible and partially hidden, it is split at the location where the visibility changes. [7] extended this work to freeform surfaces.

Reflecting on the problem at hand and seeking curve segments with homogeneous visibility from V, we need to split planar curve $C(u)$ at its silhouette points, $\mathcal{S}$. However, we also need to split $C(u)$ at all locations along the line of sight from V through $u \in \mathcal{S}$ that pierce C after the silhouette locations. These are the curve locations that become (in)visible behind silhouette points when viewed from V. Figure 6 shows an example. In (a), the silhouette locations of $C(u)$ from view location P, $\mathcal{S}$, are computed. Note here that $V = V(u) = C(u) - P$. In Figure 6 (b), these rays are extended to further pierce $C(u)$. We now have segments of C, each of which share *homogeneous visibility*, and by shooting a ray from P toward the middle of each such curve segment, its visibility can be determined.

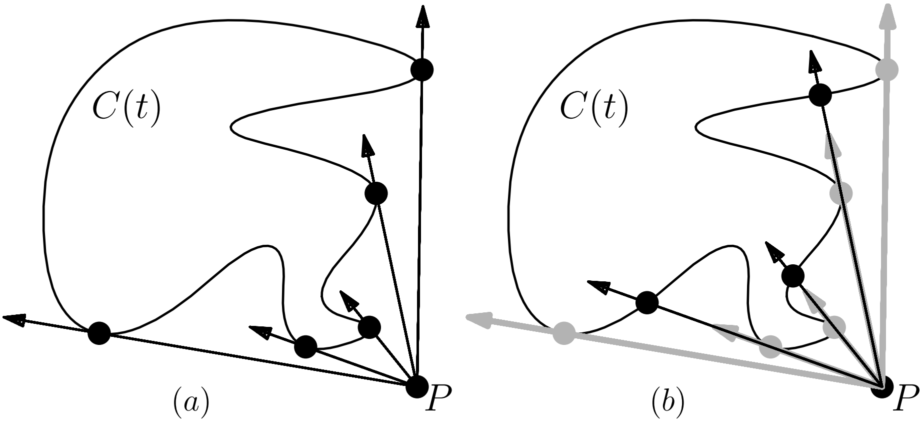


Fig. 6. A decomposition of a curve into curve segments with homogeneous visibility. In (a), the silhouette locations, $\mathcal{S}$, are computed. In (b), the silhouette rays are extended to find the locations that become visible behind $\mathcal{S}$

It should be noted we only care about visible vs. invisible segments while the level of invisibility is of no interest in this application. More on this will be found in [13].

Being able to determine the exact domain of $C(u)$ that is visible from V, one can create an *atlas of views* for a parameterized view location and/or direction. Several options could be considered to parameterize the view location/direction, following [13]:

- Use the unit circle $\mathcal{C}(\alpha)$ as a parameterization for all possible view directions, examining the geometry from infinity. Here, the visibility atlas will be a bivariate function, $V_h(\alpha, u)$.
- Use $C(v)$ as the view position's parameterization. Looking only into the interior of the curve, we are faced with the art gallery [3] problem. Looking from the inside out, we are examining, for example, the out-of-town coverage that guards on the perimeter of the town could provide. Here again, the visibility atlas is a bivariate function, $V_h(v, u)$, and this time it is also symmetric.
- Use an independent bivariate mapping $R(x, y)$ for the plane, possibly for the entire $\mathbb{R}^2$, to search for optimal placement of guards to watch curve $C(u)$, creating a visibility atlas of the form of $V_h(x, y, u)$. This map could be restricted to the interior of $C(u)$, thereby allowing the guards to be everywhere inside the gallery.

The visibility atlas answers the question of what is visible from a certain view position and/or direction. Seeking the minimal number of guards to watch a gallery or the minimal number of pieces of which a mold must be made to injection-mold the part, could be answered by discretizing the problem and reducing it to a set-covering problem [5], a problem that is NP-complete [5–p. 974] in the general case.

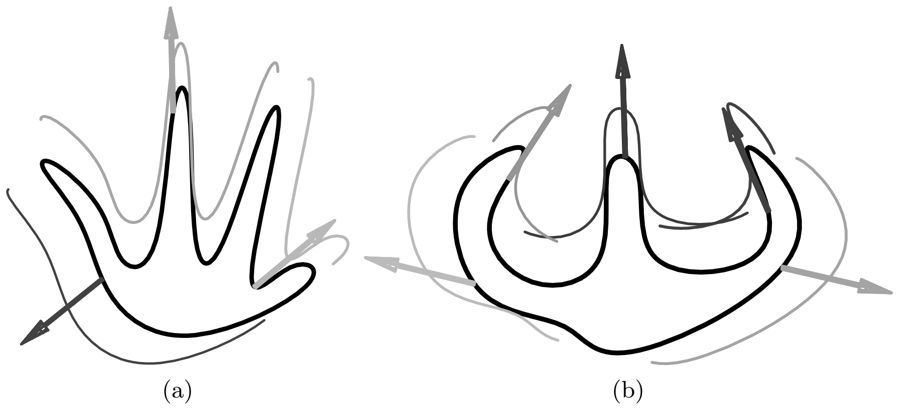


Fig. 7. Two planar curves decomposed into a three-piece mold (a) and five-piece mold (b). Note the decomposition in (b) is into non-radially monotone directions (see also Figures 8 and 9)

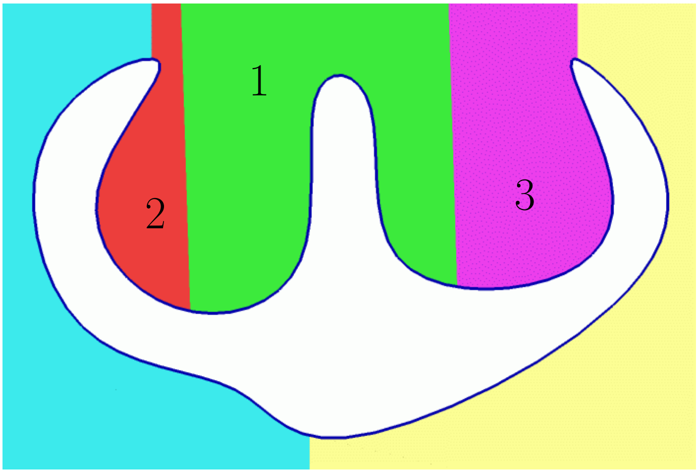


Fig. 8. The decomposition from Figure 7 yields non radially-monotone decomposition directions to form the pieces of the model. Nonetheless, a mold could be designed to realize the geometry, as is shown here

Discretize the visibility map by sampling V_h at n different locations. If for each sampled view location/direction, only one connected interval of C is visible, the solution of the set-covering problem could be reduced to polynomial complexity [13]. By sorting all the intervals along the real line and using a greedy approach to advance along the real line with the largest interval at every step, the optimum is reached. Hence, only $O(n \log n)$ is required in such a case.

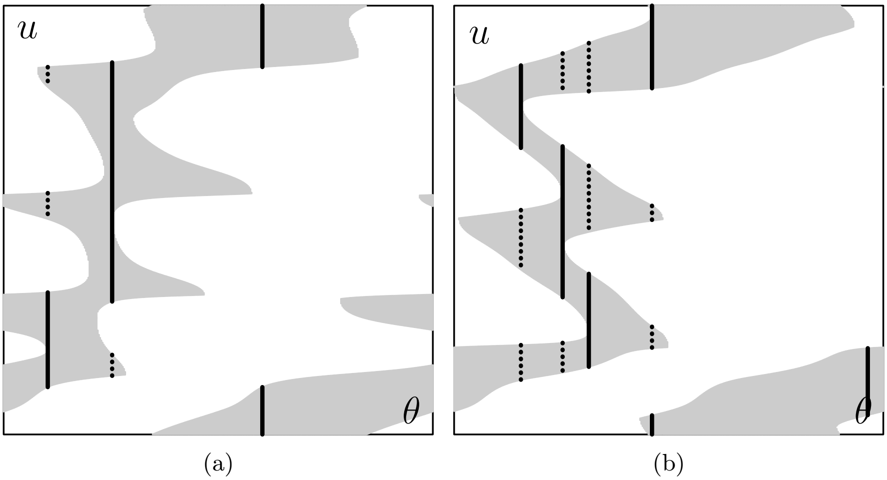


Fig. 9. The visibility atlases of the curves in Figure 7. The atlas is defined as the visible u domain (in light-gray) of $C(u)$ as a function of the view direction $\theta \in [0, 360^o]$. In (a), three views are sufficient to cover the domain while in (b) five are necessary. Only the solid intervals are considered in the coverage; the dotted intervals are ignored. Note the maps are periodic and so continuous intervals could cross boundaries

As a result, for problems where each view direction/location could hold or need hold a single interval sub-domain of C, the optimal discrete solution is tractable. This includes mold decomposition or the detection of the minimal set of parts of which a mold must be made to injection-mold a part, if possible. This is due to the fact that two disjoint intervals should be considered two different parts of the designed mold.

Figure 7 presents two examples of the mold-accessibility polynomial solution for a planar curve using the visibility atlas. In Figure 7 (a) three views are found sufficient, decomposing this curve into a three-piece mold. In Figure 7 (b), the shape must be decomposed into five pieces due to the cavities that are formed. Interestingly enough, and due to the cavities, this decomposition is made into five views that are not radially-monotone. This non-monotonicity makes the mold design a bit more complex, yet still feasible, as is shown in Figure 8. By extracting mold piece 1 in Figure 8, mold pieces 2 and 3 could be extracted as well.

Figure 9 presents the visibility atlases of the two curves in Figure 7 along with the three and five view directions selected to complete the coverage. Only one interval is used in each view and that interval is marked by a solid line in Figure 9 whereas the other intervals in that view are marked by dotted lines. 180 sampled views were used in Figure 9 (a) while 360 sampled views were used in Figure 9 (b). While the discrete sampling is not necessarily optimal, it is conservative in the sense that the offered solutions do cover the entire domain.

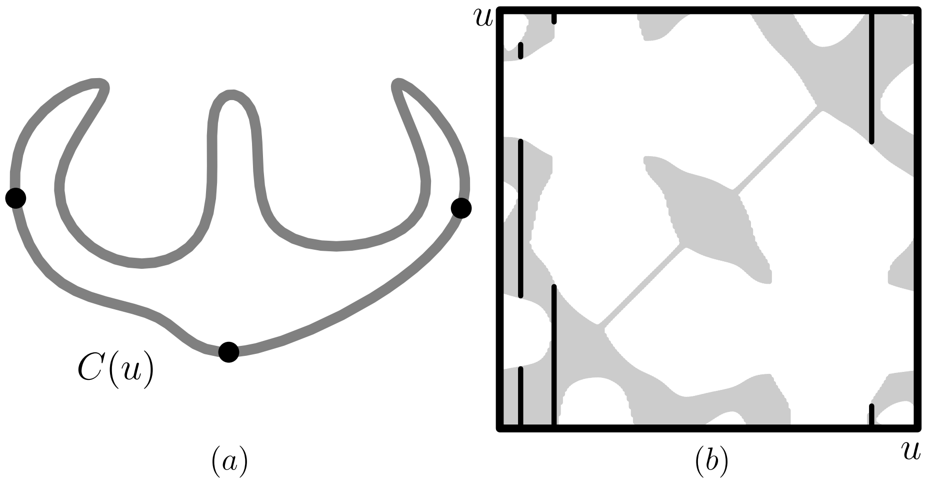


Fig. 10. Art gallery solution for the freeform shape in (a). Three guards are shown to be sufficient to cover the entire domain of this shape (as defined by the walls). In (b), the (symmetric) visibility atlas of this shape is presented along with the three selected locations whose union covers the entire parametric domain of the shape

To conclude this discussion, in Figure 10, we present one example of the art gallery problem, solving for a univariate parameterization of the view location along the curve itself, $C(u)$, and looking only into the interior domain of the curve. Three guards are found sufficient in this case, as is shown in Figure 10 (a), using the symmetric visibility atlas shown in Figure 10 (b). This set-covering case is general and hence its solution has an exponential complexity in the number of guards. Again, more on this visibility problem in the context of freeform curves, including the reduction of the freeform continuous art gallery problem into a discrete set-covering problem, can be found in [13].

4 Simple Projection of Space Curves

Consider the following simple problem: Given a simple 3-space regular parametric curve $C(u)$, find all the orthographic projection directions, if they exist, along which the image of the planar projection, $C_p(u)$, is simple or self-intersection-free. Clearly, a 3-space curve could self-intersect. Yet, a self-intersection-free 3-space curve could still intersect, once projected onto some plane. Here, we seek to find (all) directions from which $C(u)$'s projection is simple.

While this problem might seem non trivial, a simple observation could clarify the solution process. Consider some viewing direction V. If there exists two different locations on C, $C(u)$ and $C(v)$, $u \neq v$, such that $(C(u) - C(v)) \times V = 0$, V cannot serve a direction that yields a simple curve projection. In other words, we seek the directions such that for all u and v in the domain, $u \neq v$, $(C(u) - C(v)) \times V \neq 0$.

Definition 4. *The* Difference Function $\mathcal{F}_{\mathcal{M}}(u,v)$ *of a C^1 regular 3-space parametric curve $C(u) : \mathcal{D} \in I\!R \to I\!R^3$ is defined as*

$$\mathcal{F}_{\mathcal{M}}(u,v) = C(u) - C(v) : [\mathcal{D} \times \mathcal{D}] \to I\!R^3.$$

This difference map was used in the past [8, 10] to compute the distance between two different planar curves and also to find their intersection locations. The square of $||C_1(u) - C_2(v)||$ is a rational scalar field, for rationals C_i. Hence, its zeros locate the intersection locations, if any. Let $\tilde{N}_1(u)$ be an (unnormalized) normal field of planar curve $C_1(u)$ computed by rotating $C'(u)$ 90 degrees in the clockwise direction, in the plane. In [11], the scalar bivariate field of $\left\langle C_1(u) - C_2(v), \tilde{N}_1(u) \right\rangle$ for planar curves C_i was considered. Given two Bézier or B-spline planar regular curves, the control coefficients of the B-spline field of $\left\langle C_1(u) - C_2(v), \tilde{N}_1(u) \right\rangle$ is derived. Then, if all coefficients are of the same sign, the curves do not intersect. The symmetric test could be applied using $\tilde{N}_2(v)$ as $\left\langle C_1(u) - C_2(v), \tilde{N}_2(v) \right\rangle$, to provide an even better bound for intersection-free cases.

However, and going back to 3-space curves, $C(u) - C(v)$, $u \neq v$ also hints at directions that pierce the curve more than once and hence are invalid as simple curve projection directions. Centrally map all these directions of $\mathcal{F}_{\mathcal{M}}$ onto the unit sphere S^2. Every region of S^2 that is not covered by $\mathcal{F}_{\mathcal{M}}$'s central projection could then serve as a valid projection. In fact, it is sufficient to project only half of $\mathcal{F}_{\mathcal{M}}$ due to its inherent symmetry, and consequently, one should only consider the sub domain of $\mathcal{D} \times \mathcal{D}$ for which $u > v$.

Along the diagonal, $u = v$, the magnitude of $\mathcal{F}_{\mathcal{M}}$ vanishes identically. Interestingly enough, the neighborhood of the diagonal of $\mathcal{F}_{\mathcal{M}}$ represents the limit of $C(u) - C(v)$ where u and v approach the same value. In other words, the diagonal expresses the derivative of C and hence could be derived as the tangent field of C. Indeed, the tangent field, when mapped onto S^2, delineates the valid views from views where the curves starts to self-intersect. Nonetheless, the tangent field by itself is insufficient and one must consider all possible views in $C(u) - C(v)$, as will be shortly demonstrated.

A practical algorithm to detect all valid projection directions would perform the following steps:

1. Given $C(u)$, define the lifted $\mathcal{F}_{\mathcal{M}}(u,v)$ bivariate function.
2. Centrally project the sub domain of $\mathcal{F}_{\mathcal{M}}(u,v)$ for which $u > v$ onto the unit sphere S^2. This projection could, in practice, be implemented using the projection of an arbitrarily close tessellated approximation of $\mathcal{F}_{\mathcal{M}}(u,v)$.
3. Having a binary map on the unit sphere of covered vs. uncovered regions, one can either:
   (a) Extract all the uncovered regions over S^2 that contain the views with a simple projection, if any, or
   (b) Find a projection direction in the middle of the uncovered regions. These centered locations offer robust projection directions that yield a simple curve projection, even after small perturbations.

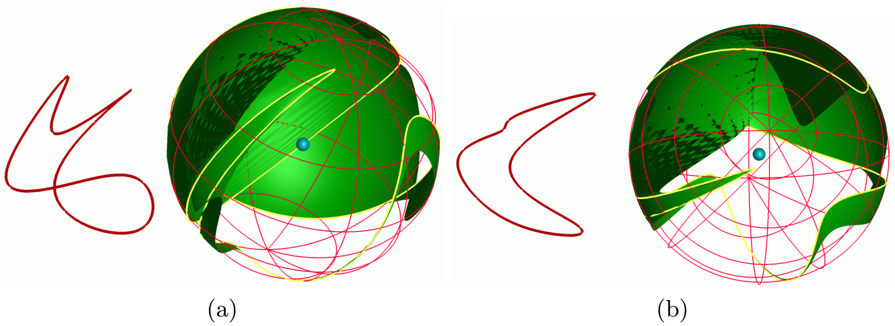


Fig. 11. Computation of directions of a simple 3-space curve that yields a simple projected curve. In (a) an invalid direction is presented where the S^2 covers that direction whereas in (b) a valid direction is shown with an uncovered direction in S^2. The projection of $\mathcal{F}_{\mathcal{M}}$ onto S^2 is shown in grey; the unit tangent field of the curve is shown as a pale line

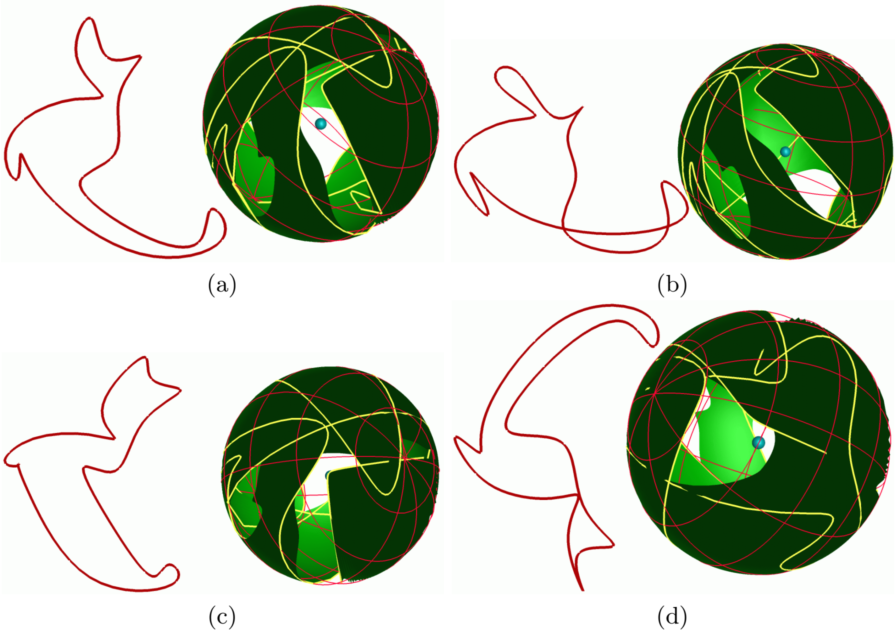


Fig. 12. Computation of the directions that yield a simple projection of a simple 3-space curve. (a) and (c) provide two valid directions whereas (b) shows an invalid one. (c) and (d) present the limiting cases of a cusp (c) and a tangency (d). The projection of $\mathcal{F}_{\mathcal{M}}$ onto S^2 is shown in grey; the unit tangent field of the curve is shown as a pale line

Stage 3b in this proposed algorithm could be accomplished by resorting to image processing techniques. Centrally project the binary map over S^2 onto the six faces of a bounding cube. By applying image dilation procedures on all six faces, one can find the valid viewing directions, if any, that are as far as possible from the covered regions. See, for example, [12] for more on such a dilation process, in a similar context.

Figure 11 shows one example where a simple periodic 3-space C^2 cubic B-spline curve with 11 control points is projected twice, once with an intersection (in Figure 11 (a)) and once as a simple projection (in Figure 11 (b)). In each case, the projected curve is plotted on the left side and the unit sphere of directions is shown on the right, with the invalid regions painted in solid light grey. Also shown, in pale grey, is the normalized tangent field of the curve, which is on S^2. This tangent field serves to delineate some portions of the light grey solid regions (the projection of $\mathcal{F}_{\mathcal{M}}$ onto S^2) but not all. The reason is that in some cases two independent regions of the curve, which are approaching, become tangent, and then intersect in the projection plane. Here the tangent field is irrelevant, yet (global) self-intersection still occurs.

Figure 12 shows a second, more complex example of a simple periodic 3-space C^2 cubic B-spline curve with 22 control points that is examined for simple projection directions. Here, the set of valid projections is narrow and yet easily identified once $\mathcal{F}_{\mathcal{M}}(u, v)$, $u > v$ is projected onto S^2. Again, appearing in yellow, the unit tangent field of the curve is also presented.

Finally, it should be noted that if $C(u_0) - C(v_0)$ is an invalid direction, so is $C(v_0) - C(u_0)$. In other words, when projecting the $\mathcal{F}_{\mathcal{M}}(u, v)$, $u > v$ field onto S^2, the antipodal projection should be considered as well. In Figures 11 and 12, only the original projection is considered for clarity, while the center of the unit sphere is drawn as a small sphere. The dual projection requirement is equivalent to finding a direction that "sees" the sphere's center over the background. In other words, in a valid view direction, the sphere center is covered by $\mathcal{F}_{\mathcal{M}}(u, v)$, $u > v$, from neither the front side nor the back side of the sphere.

5 Conclusions and Future Work

This work investigated a few lifting methods for extracting the global properties of freeform rational curves. The features of the orthogonality maps should be further investigated as well as extended into parallel and/or angular maps:

Definition 5. *The* Angular Function $\mathcal{A}_{\mathcal{F}}(u, v)$ *of a* C^1 *regular planar parametric curve* $C(u) : \mathcal{D} \in I\!R \rightarrow I\!R^2$ *is defined as*

$$\mathcal{A}_{\mathcal{F}}(u, v) = \frac{\langle C'(u), C'(v)\rangle^2}{\langle C'(u), C'(u)\rangle \langle C'(v), C'(v)\rangle} : [\mathcal{D} \times \mathcal{D}] \rightarrow I\!R.$$

and

Definition 6. *The* γ Angular Map $\mathcal{A}_{\mathcal{M}}[\gamma]$ *of a* C^1 *regular planar parametric curve* $C(u)$*, denoted* $\mathcal{A}_{\mathcal{M}}[C, \gamma]$*, is the constant set of the angular function,* $\mathcal{A}_{\mathcal{F}}(u, v) = \cos^2(\gamma)$*.*

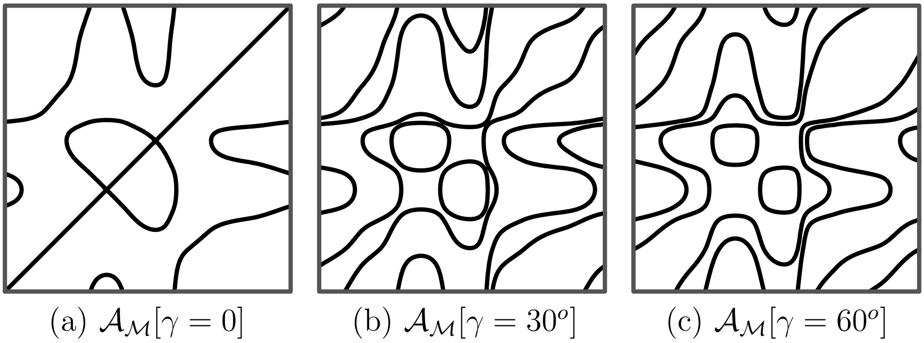

(a) $\mathcal{A}_{\mathcal{M}}[\gamma = 0]$
(b) $\mathcal{A}_{\mathcal{M}}[\gamma = 30^o]$
(c) $\mathcal{A}_{\mathcal{M}}[\gamma = 60^o]$

Fig. 13. Angular maps of the curve shown in Figure 1

Parallel and orthogonal maps are two extreme cases of angular maps, for which $\gamma = 0$ and $\gamma = \pi/2$. Being more general, the angular map lifting scheme deserves some more research. Figure 13 presents a few examples of angular maps of the curve shown in Figure 1. In (a), a parallel map ($\mathcal{A}_{\mathcal{M}}[\gamma]$ for $\gamma = 0$) is presented. The diagonal line of $u = v$ is clearly a valid solution for the parallel map. Yet, one interesting property of the parallel map, which is simple to verify, is that the number of off-diagonal branches that intersect the main diagonal exactly equal the number of inflection points in the curve, two in this case. The angular maps, for general angles, offer, for each curve location, information regarding how far one can move before the curve turns γ degrees. This information is etched onto these maps as the vertical (or horizontal) distance from the diagonal location. Figures 13 (b) and (c) show two maps of the curve in Figure 1, for γ of 30 and 60 degrees, respectively. If these maps are rebuilt using arc-length parameterizations, then they will be able to answer questions such as maximum and/or minimal turning angles of the curve per unit-length.

Open planar curves break the periodicity of the orthogonal/parallel/angular maps and hence would require special boundary conditioning. Piecewise C^1 planar curves also deserve attention, when investigating these maps. The C^0 locations would introduce discontinuities into the angular functions and their proper handling and map extraction would be more difficult.

In Section 4, a method to derive the proper projection directions of a simple space curve that would yield a simple curve has been proposed. This method could clearly be used to find directions from which surface $S(u, v)$ is completely visible. When the 4-variate vector field of $F(u, v, s, t) = S(u, v) - S(s, t)$ is projected onto S^2, its uncovered regions would yield the completely visible projection directions, if any. It should be noted that this computation is not the same as finding a direction that presents no silhouettes. A surface could possess no silhouettes from a certain view direction and yet still occlude itself.

References

1. H.-K. Ahn, M. de Berg, P. Bose, S. Cheng, D. Halperin, J. Matoušek, and O. Cheong. Separating an object from its cast, Computer-Aided Design, Vol 34, pp 547–559, 2002.
2. A. Appel. The notion of quantitative invisibility and the machine rendering of solids, Proc. ACM National Conference, Washington, DC, 387–393, 1967.
3. M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry, Algorithms, and Applications (2nd ed.), Springer-Verlag, Berlin, 2000.
4. M. Bloomenthal. Approximation of sweep surfaces by tensor product B-splines, Tech Reports UUCS-88-008, University of Utah, 1988.
5. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms, MIT Press and McGraw-Hill, 1990.
6. M. do Carmo. Differential Geometry of Curves and Surfaces. Prentice Hall, 1976.
7. G. Elber and E. Cohen. Hidden curve removal for free form surfaces, Computer Graphics (Proc. SIGGRAPH), 24 (1990), 95–104.
8. G. Elber. Symbolic and numeric computation in curve interrogation, Computer Graphics Forum, Vol 14 pp 25–34, 1995.
9. G. Elber. Multiresolution curve editing with linear constraints, The Journal of Computing & Information Science in Engineering, Vol 1, No 4, pp 347–355, Dec 2001.
10. G. Elber. Trimming local and global self-intersections in offset curves using distance maps, Proc. of the 10th IMA Conference on the Mathematics of Surfaces, Leeds, UK, pp 213–222, September 2003.
11. G. Elber. Distance separation measures between parametric curves and surfaces toward intersection and collision detection applications, Proceedings of COMPASS 2003, Schloss Weinberg, Austria, October 2003.
12. G. Elber, X. Chen, and E. Cohen. Mold accessibility via Gauss map analysis, Shape Modeling International 2004, Genova, Italy, pp 263–274, June 2004.
13. G. Elber, R. Sayegh, G. Barequet, and R. R. Martin. Two-dimensional visibility charts for continuous curves, to Appear in Shape Modeling International 2005, Boston, USA, June 2005.
14. C. Gonzales-Ochoa, S. Mccamnon, and J. Peters. Computing moments of objects enclosed by piecewise polynomial surfaces, ACM Transactions on Graphics, Vol 17, No 3, pp 143–157, July 1998.
15. S. Hahmann, G.-P. Bonneau, and B. Sauvage. Area preserving deformation of multiresolution curves, submitted.
16. J. Keyser, T. Culver, D. Manocha and S. Krishnan, Efficient and exact manipulation of algebraic points and curves, Computer-Aided Design, Vol 32, No 11, pp 649–662, Sep 2000.
17. F. Klok. Two moving coordinate frames for sweeping along a 3D trajectory, Computer Aided Geometric Design, Vol 3, No 3, pp 217–229, 1986.
18. A. Rappaport, A. Sheffer, and M. Bercovier. Volume-preserving free-form solids, IEEE Transactions on Visualization and Computer Graphics, Vol 2, No 1, pp 19–27, March 1996.
19. T. Woo. Visibility maps and spherical algorithms, Computer-Aided Design, 26 (1994), pp 6–16.

Conversion of Dupin Cyclide Patches into Rational Biquadratic Bézier Form

Sebti Foufou[1], Lionel Garnier[1], and Michael J. Pratt[2]

[1] LE2I, UMR CNRS 5158, UFR Sciences, Université de Bourgogne,
BP 47870, 21078 Dijon Cedex, France
{sfoufou, lgarnier}@u-bourgogne.fr
[2] LMR Systems, 21 High Street, Carlton, Bedford,
MK43 7LA, UK
mike@lmr.clara.co.uk

Abstract. This paper uses the symmetry properties of circles and Bernstein polynomials to establish a series of interesting *barycentric properties* of rational biquadratic Bézier patches. A robust algorithm is presented, based on these properties, for the conversion of Dupin cyclide patches into Bézier form. A set of conversion examples illustrates the use of this algorithm.

1 Introduction

Rational Biquadratic Bézier Surfaces (RBBSs) are tensor product parametric surfaces widely used in the first generation of computer graphics applications and geometric modelling systems. Good introductions to RBBSs may be found in [1, 2, 3, 4].

Dupin cyclide surfaces represent a family of ringed surfaces, i.e., surfaces generated by a circle of variable radius sweeping through space [5, 6, 7]. It is possible to formulate them either as algebraic or parametric surfaces. In recent decades, the interest of several authors in these surfaces relates to their potential value in the development of CAGD tools [8, 9]. Also, cyclide intersections and the use of cyclides as blending surfaces have been investigated [10, 11].

The primary aim of this paper is to prove a series of useful properties of RBBSs, called *barycentric properties*, and to show how they can be used to convert cyclide patches into RBBSs. Section 2 gives background information concerning Rational Quadratic Bézier Curves (RQBCs) and RBBSs, introduces the Dupin cyclides, and discusses an algorithm for conversion of Dupin cyclide patches to RBBSs. Section 3 shows the use of RQBCs to represent circular arcs. Section 4 states and proves a set of new barycentric properties of RBBSs. Section 5 uses these properties to define a robust new algorithm for the conversion of Dupin cyclides into RBBSs, and illustrates some of the conversion results obtained. Section 6 presents our conclusions and suggests directions for future work.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 201–218, 2005.

2 Background

2.1 Rational Quadratic Bézier Curves and Surfaces

A Rational Quadratic Bézier Curve (RQBC) is a second degree parametric curve defined by:

$$\overrightarrow{OM(t)} = \frac{1}{\sum_{i=0}^{2} w_i B_i(t)} \left(\sum_{i=0}^{2} w_i B_i(t) \overrightarrow{OP_i} \right), \, t \in [0,1] \tag{1}$$

where $B_i(t)$ are quadratic Bernstein polynomials defined as: $B_0(t) = (1-t)^2$, $B_1(t) = 2t\,(1-t)$ and $B_2(t) = t^2$, and for $i \in \{0,1,2\}$, w_i are weights associated with the control points P_i. For a standard RQBC w_0 and w_2 are equal to 1, while w_1 can be used to control the type of conic defined by the curve.

Rational Biquadratic Bézier Surfaces (RBBSs) are defined by a tensor product of two RQBCs by:

$$\overrightarrow{OM\,(u,v)} = \frac{1}{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\,(u)\, B_j\,(v)} \sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\,(u)\, B_j\,(v)\, \overrightarrow{OP_{ij}} \tag{2}$$

More details on Bézier curves and surfaces can be found in [12, 4].

2.2 Dupin Cyclides

Non-degenerate Dupin cyclides can be characterized by either of the following two equivalent implicit equations:

$$\left(x^2+y^2+z^2-\mu^2+b^2\right)^2 = 4\,(ax-c\mu)^2 + 4b^2y^2 \tag{3}$$

$$\left(x^2+y^2+z^2-\mu^2-b^2\right)^2 = 4\,(cx-a\mu)^2 - 4b^2z^2 \tag{4}$$

Parameters a, b and c are related by $c^2 = a^2 - b^2$. The parameter a is always greater than or equal to c. Parameters a, c and μ determine the type of the cyclide. When $c < \mu \leq a$ it is a ring cyclide, when $0 < \mu \leq c$ it is a horned cyclide, and when $\mu > a$ it is a spindle cyclide. The parametric form is represented by equation (5), where parameters θ and ψ satisfy $(0 \leq \theta,\, \psi \leq 2\pi)$:

$$\begin{cases} x(\theta,\psi) = \dfrac{\mu(c - a\cos\theta\cos\psi) + b^2\cos\theta}{a - c\cos\theta\cos\psi} \\[2ex] y(\theta,\psi) = \dfrac{b\sin\theta\, \times (a - \mu\cos\psi)}{a - c\cos\theta\cos\psi} \\[2ex] z(\theta,\psi) = \dfrac{b\sin\psi\, \times (c\cos\theta - \mu)}{a - c\cos\theta\cos\psi} \end{cases} \tag{5}$$

These are the most important properties of Dupin cyclide surfaces:

- Simple mathematical representation in either implicit or parametric form.
- Circular lines of curvature. These lines of curvature correspond to isoparametric lines of constant θ or ψ. The angle between the surface normal and the principal normal on the lines of curvature is also constant.
- Each cyclide has two perpendicular planes of symmetry. The intersections of a cyclide with its planes of symmetry are two pairs of circles called the *principal circles.* These are the lines of curvature of the cyclide for which $\psi = 0$, $\psi = \pi$, $\theta = 0$ or $\theta = \pi$.
- In the parametric form, knowledge of the four principal circles allows the computation of the parameters a, c, and μ.
- The curvature line and tangent cone properties given above make it easy to use Dupin cyclides to create blends between other cyclide surfaces, including the important special cases of circular cylinders, circular cones, spheres and toruses.

2.3 Conversion of Dupin Cyclides to RBBSs

If a RBBS is parameterized in terms of u and v, its isoparametric curves are conics. Lines of curvature of Dupin cyclides are circles, which are particular cases of conics. It is thus possible to convert a Dupin cyclide patch into an RBBS. In the rest of this paper, the Dupin cyclide patch to be converted will be referred to as a *cyclide patch.* It is assumed to be delimited by the curvature lines corresponding to parameter values $(\theta_0, \theta_1, \psi_0, \psi_1)$.

A conversion algorithm of the type discussed was proposed in [5]. The control points and their associated weights were determined from the parametric equation of the Dupin cyclide (equation (5)) by the following method:

1. The parameter transformations
$$\forall\theta \in \mathbb{R} - (\pi + 2\pi\mathbb{Z}), \quad \cos\theta = \frac{1-\tan^2 U}{1+\tan^2 U}, \; \sin\theta = \frac{2\tan U}{1+\tan^2 U} \tag{6}$$
where $U = \tan(\theta/2)$, together with corresponding expressions for $\cos\psi$ and $\sin\psi$ in terms of $V = \tan(\psi/2)$, were used to obtain rational quadratic forms for the three coordinates in terms of U, V.
2. A linear reparametrization was now applied to obtain a representation in terms of u, v such that $u, v \in [0, 1]$.
3. The common denominator of the resulting scalar-valued rational biquadratics in u and v was reformulated in terms of the quadratic Bernstein basis functions. The resulting coefficients of those basis functions were identified with the weights w_{ij} in the denominator of equation (2).
4. Finally, with the weights known, the numerators of the three rational biquadratics were similarly reformulated. This gave the coordinates of the patch control points, expressed in terms of the defining constants of the particular cyclide, a, c, μ, together with the bounding parameter values of the original cyclide patch.

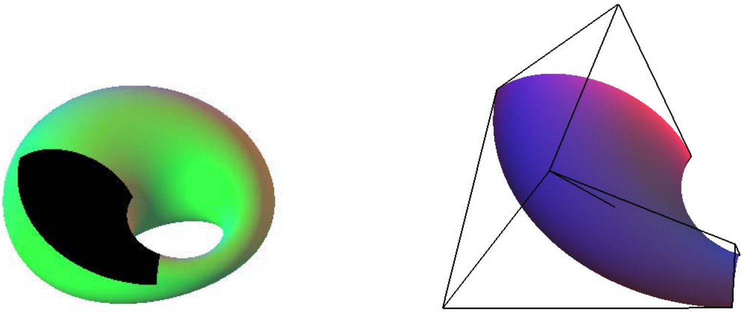


Fig. 1. Conversion of a Dupin cyclide patch (left) to RBBS (right) using Pratt's original algorithm

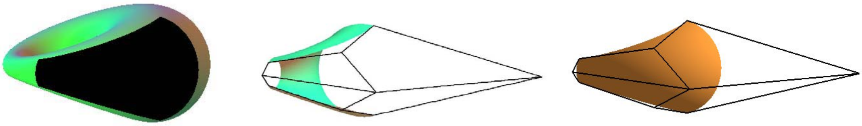


Fig. 2. Incorrect conversion. Left, the cyclide patch. Middle, the resulting RBBS representing the complementary patch, not the intended one. Right, the correct corresponding RBBS

Figure 1 shows an example of the conversion of a Dupin cyclide patch into a RBBS using this algorithm.

A problem with this algorithm is that the function tan is discontinuous, not being defined for values $\frac{\pi}{2} + \pi\mathbb{Z}$. This has the results that

1. for certain choices of patch boundary the reparametrization functions are not defined (though these special cases could be identified and appropriate limiting values used);
2. more importantly, the algorithm gives incorrect results when $\pi \in]\theta_0, \theta_1[$ or when $\pi \in]\psi_0, \psi_1[$.

Figure 2 illustrates a case of erroneous results from this algorithm. The cyclide patch (left subfigure) is defined by $a = 6$, $\mu = 3$, $c = 2$, $\theta_0 = \frac{\pi}{4}$, $\theta_1 = \frac{5\pi}{6}$, $\psi_0 = \frac{2\pi}{3}$ and $\psi_1 = \frac{4\pi}{3}$. To obtain the RBBS of the right subfigure, that correctly represents the cyclide patch of the left subfigure, we used a new variant of this algorithm obtained by taking the absolute value of the weights as calculated by Pratt's original formula, i.e.,

$$w_{ij} = |a\,(1 + G_i)\,(1 + H_j) - c\,(1 - G_i)\,(1 - H_j)| \tag{7}$$

In this formula, the variables G_i and H_j are intermediate values computed from tan of $\theta_0/2$, $\theta_1/2$, $\psi_0/2$ and $\psi_1/2$ (see [5] for further details).

As we have seen, the boundary parameter values θ_0, θ_1, ψ_0 and ψ_1, together with the values of a, c and μ, can be used to determine control points and weights

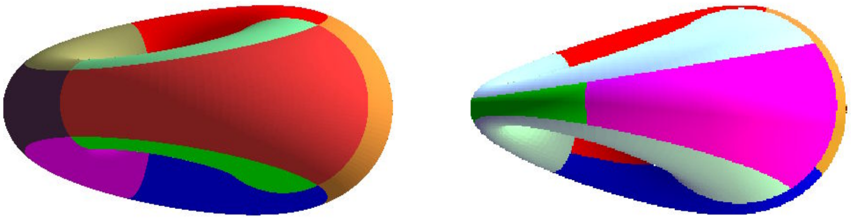


Fig. 3. Conversion of a whole Dupin cyclide into a set of RBBSs

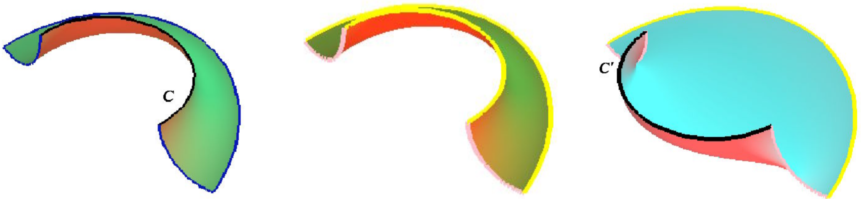


Fig. 4. Conversion of a cyclide patch. Left, the cyclide patch. Middle, the RBBS given by Pratt's algorithm. Right, the RBBS given by the variant of Pratt's algorithm

of the RBBS. If $\theta_0 = 0$ and $\theta_1 = \frac{4\pi}{3}$, the computed control points calculated by the variant algorithm incorrectly correspond to the patch delimited by $\theta_0 = -\frac{2\pi}{3}$ and $\theta_1 = 0$. We must therefore require $|\theta_0 - \theta_1| < \pi$ and $|\psi_0 - \psi_1| < \pi$. Taking this constraint into consideration gives several possibilities for conversion of a complete Dupin cyclide. Figure 3 left illustrates a Dupin cyclide converted into 9 RBBSs using this new variant of the algorithm. Combined use of both the original and the variant algorithm allows to decrease the number of RBBSs to 6, which is believed to be the minimal number of RBBSs necessary for the representation of a complete Dupin cyclide (Figure 3 right).

The new variant of this algorithm makes it possible to convert a whole Dupin cyclide into a set of RBBSs, but it is still not possible to convert a cyclide patch delimited by a curve obtained with one of the parameters equal to π. Moreover, having only positive weights is not enough to work correctly for certain cases where some control points need to have negative weights.

Figure 4 shows a case where the original algorithm works correctly (the middle subfigure) while the new variant fails (the right subfigure). The cyclide patch is defined by $a = 6$, $c = 2$, $\mu = 3$, $\theta_0 = -\frac{\pi}{3}$, $\theta_1 = \frac{\pi}{2}$, $\psi_0 = 0$ and $\psi_1 = \frac{\pi}{2}$. The output of the new variant is completely wrong: Border curve C' is the complement, on the circle, of the border curve C of the cyclide patch. This is due to the fact that w_{10} is positive while it should be negative to model the border curve C. The same problem occurs for w_{11}.

Another conversion example is given in Figure 5. Parameters defining the cyclide patch are $a = 6$, $c = 2$, $\mu = 3$, $\theta_0 = \frac{2\pi}{3}$, $\theta_1 = \frac{4\pi}{3}$, $\psi_0 = \frac{5\pi}{6}$ and $\psi_1 = \frac{3\pi}{2}$. The original algorithm gives exactly the complement of the cyclide patch: the

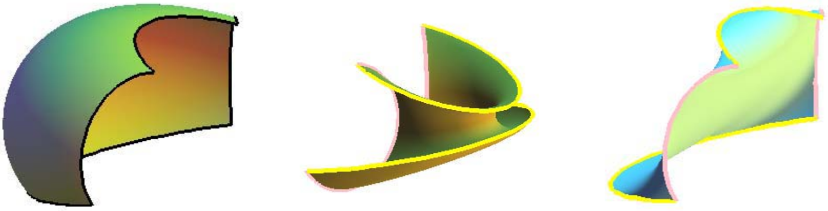


Fig. 5. A conversion example where both Pratt's algorithm and its variant fail

corner points are those of the cyclide patch while thr delimiting curves are the complements, on the circle, of the curves defining the cyclide patch. On the other hand, the RBBS obtained by the new variant of this algorithm (right subfigure) has the same corner points and border curves as the cyclide patch, but the central control point P_{11} is not correct; the weight w_{11} should be negative, while the new variant gives only positive weights.

A less problematic conversion algorithm, based on barycentric properties of RBBSs and geometric properties of Dupin cyclides is proposed in section 5.

3 Modeling Circular Arcs Using RQBCs

RQBCs can be used to model conics. Three control points and a scalar value (the weight of the middle control point) are enough to define an arc of a conic. In this section, we give some results on the expression of circular arcs using RQBC. Theorem 1 shows how to define a circle from two points and two tangents on these points. Theorem 2 presents how to compute the middle control point of the RQBC that represents a given circular arc. Theorem 3 shows how to compute the weight of the middle control point of the RQBC that represents a given circular arc. Figure 6 shows the modeling of circular arcs using RQBC. This is the geometric construction used for Theorems 1, 2 and 3. $\mathsf{C}(O_0, R)$ is a circle of centre O_0 and radius R. Segments $[P_0P_1]$ and $[P_2P_1]$ are tangents to the circle at points P_0 and P_2. I_1 is the midpoint of segment $[P_0P_2]$. $\mathcal{P}$ is the median plane of segment $[P_0P_2]$.

Theorem 1. *Circle from two points and tangents at these points.*

- *If the circle $\mathsf{C}(O_0, R)$ exists then $P_1 \in \mathcal{P}$ and $P_1 \notin [P_0P_2]$. The radius is $R = O_0P_0$ and the centre O_0 is given by formula:*

$$\overrightarrow{P_1O_0} = t_0\overrightarrow{P_1I_1} \qquad t_0 = \frac{P_0P_1^2}{\overrightarrow{I_1P_1}\bullet\overrightarrow{P_0P_1}} \tag{8}$$

- *In the plane determined by C, the geometric angle $\widehat{P_0O_0P_2}$ is less than π, this means that if we take a parametrization γ of the circle in terms of cosine and sine such as $P_0 = \gamma(\theta_0)$, $P_2 = \gamma(\theta_1)$, we have $|\theta_0 - \theta_1| < \pi$.*

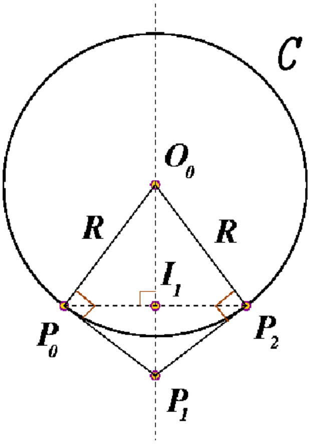


Fig. 6. Modeling circular arcs by RQBC

Theorem 2. *Computing control point P_1 when the centre of the circle is known.*

The RQBC is the arc of the circle C *passing through P_0 and P_2. In this case, the control point P_1 satisfies:*

$$\overrightarrow{I_1P_1} = t_1\overrightarrow{O_0I_1} \qquad t_1 = \frac{\overrightarrow{O_0P_0} \bullet \overrightarrow{I_1P_0}}{\overrightarrow{O_0P_0} \bullet \overrightarrow{O_0I_1}} \tag{9}$$

Theorem 3. *Computing the weight w_1.*

The RQBC defined by control points P_0, P_1 and P_2 and the weight w_1 is a circular arc if and only if the following condition hold:

$$|1 + w_1|\, R = |O_0I_1 + w_1O_0P_1| \tag{10}$$

The RQBC defines the smaller arc of the circle if:

$$w_1 = \frac{O_0I_1 - R}{R - O_0P_1} = \frac{O_0I_1 - O_0P_0}{O_0P_0 - O_0P_1} > 0 \tag{11}$$

It defines the larger arc of the circle if:

$$w_1 = -\frac{O_0I_1 + R}{R + O_0P_1} = -\frac{O_0I_1 + O_0P_0}{O_0P_0 + O_0P_1} < 0 \tag{12}$$

Proofs of these three theorems can be easily obtained by combining properties of RQBC with those of the circle and scalar product.

4 Barycentric Properties of RBBSs

Let S_0 be a Rational Biquadratic Bézier Surface (RBBS) defined according to formula (2) by control points $(P_{ij})_{0\leq i,j\leq 2}$ and weights $(w_{ij})_{0\leq i,j\leq 2}$ with $w_{00} =$

$w_{02} = w_{20} = w_{22} = 1$. In order to model surfaces with spherical curvatures by surface S_0, we should have the following constraints on control points: P_{01} belongs to the median plane of $[P_{00}P_{02}]$, P_{10} belongs to the median plane of $[P_{00}P_{20}]$, P_{21} belongs to the median plane of $[P_{20}P_{22}]$ and P_{12} belongs to the median plane of $[P_{02}P_{22}]$.

The position of control point P_{11} is less obvious than the positions of the others. The following four theorems prove that P_{11} belongs to three particular lines. A set of interesting barycentric properties of RBBSs that helps in the construction of these lines are given and proved.

Theorem 4. *Let I_0, J_0, I_2 and J_2 be respectively the midpoints of the segments $[P_{00}P_{02}]$, $[P_{00}P_{20}]$, $[P_{20}P_{22}]$ and $[P_{02}P_{22}]$. We have the following four relations:*

$$\overrightarrow{OM\left(0,\frac{1}{2}\right)} = \frac{1}{1+w_{01}}\left(\overrightarrow{OI_0} + w_{01}\overrightarrow{OP_{01}}\right) \tag{13}$$

$$\overrightarrow{OM\left(1,\frac{1}{2}\right)} = \frac{1}{1+w_{21}}\left(\overrightarrow{OI_2} + w_{21}\overrightarrow{OP_{21}}\right)$$

$$\overrightarrow{OM\left(\frac{1}{2},0\right)} = \frac{1}{1+w_{10}}\left(\overrightarrow{OJ_0} + w_{10}\overrightarrow{OP_{10}}\right) \tag{14}$$

$$\overrightarrow{OM\left(\frac{1}{2},1\right)} = \frac{1}{1+w_{12}}\left(\overrightarrow{OJ_2} + w_{12}\overrightarrow{OP_{12}}\right)$$

Proof. In order to prove the result (13) for points $\overrightarrow{OM\left(0,\frac{1}{2}\right)}$ and $\overrightarrow{OM\left(1,\frac{1}{2}\right)}$, let us recall that $B_0(0) = 1$, $B_1(0) = B_2(0) = 0$, $B_2(1) = 1$, $B_1(1) = B_0(1) = 0$, $B_1\left(\frac{1}{2}\right) = \frac{1}{2}$, $B_0\left(\frac{1}{2}\right) = B_2\left(\frac{1}{2}\right) = \frac{1}{4}$, and if I is the midpoint of a segment $[AB]$, then for every point O we have $\overrightarrow{OA} + \overrightarrow{OB} = 2\overrightarrow{OI}$. By Formula (2), the point $\overrightarrow{OM\left(0,\frac{1}{2}\right)}$ on the RBBS is:

$$\begin{aligned}
\overrightarrow{OM\left(0,\frac{1}{2}\right)} &= \frac{\displaystyle\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i(0)\,B_j\left(\frac{1}{2}\right)\overrightarrow{OP_{ij}}}{\displaystyle\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i(0)\,B_j\left(\frac{1}{2}\right)} = \frac{\displaystyle\sum_{j=0}^{2} w_{0j}B_j\left(\frac{1}{2}\right)\overrightarrow{OP_{0j}}}{\displaystyle\sum_{j=0}^{2} w_{0j}B_j\left(\frac{1}{2}\right)} \\
&= \frac{1}{\frac{w_{00}}{4}+\frac{w_{01}}{2}+\frac{w_{02}}{4}}\left(\frac{w_{00}}{4}\overrightarrow{OP_{00}} + \frac{w_{01}}{2}\overrightarrow{OP_{01}} + \frac{w_{02}}{4}\overrightarrow{OP_{02}}\right) \\
&= \frac{1}{\frac{1}{4}+\frac{w_{01}}{2}+\frac{1}{4}}\left(\frac{1}{4}\left(\overrightarrow{OP_{00}} + \overrightarrow{OP_{02}}\right) + \frac{w_{01}}{2}\overrightarrow{OP_{01}}\right) \\
&= \frac{1}{\frac{1}{2}+\frac{w_{01}}{2}}\left(\frac{1}{4}2\overrightarrow{OI_0} + \frac{w_{01}}{2}\overrightarrow{OP_{01}}\right) \\
&= \frac{2}{1+w_{01}}\left(\frac{1}{2}\overrightarrow{OI_0} + \frac{1}{2}w_{01}\overrightarrow{OP_{01}}\right) \\
&= \frac{1}{1+w_{01}}\left(\overrightarrow{OI_0} + w_{01}\overrightarrow{OP_{01}}\right)
\end{aligned}$$

On the other hand, point $\overrightarrow{OM\left(1,\frac{1}{2}\right)}$ on the RBBS is:

$$\begin{aligned}
\overrightarrow{OM\left(1,\frac{1}{2}\right)} &= \frac{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\left(1\right) B_j\left(\frac{1}{2}\right)\overrightarrow{OP_{ij}}}{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\left(1\right) B_j\left(\frac{1}{2}\right)} \\
&= \frac{1}{\sum_{j=0}^{2} w_{2j} B_j\left(\frac{1}{2}\right)} \sum_{j=0}^{2} w_{2j} B_j\left(\frac{1}{2}\right)\overrightarrow{OP_{2j}} \\
&= \frac{1}{\frac{w_{20}}{4}+\frac{w_{21}}{2}+\frac{w_{22}}{4}}\left(\frac{w_{20}}{4}\overrightarrow{OP_{20}}+\frac{w_{21}}{2}\overrightarrow{OP_{21}}+\frac{w_{22}}{4}\overrightarrow{OP_{22}}\right) \\
&= \frac{1}{\frac{1}{4}+\frac{w_{21}}{2}+\frac{1}{4}}\left(\frac{1}{4}\left(\overrightarrow{OP_{20}}+\overrightarrow{OP_{22}}\right)+\frac{w_{21}}{2}\overrightarrow{OP_{21}}\right) \\
&= \frac{1}{\frac{1}{2}+\frac{w_{21}}{2}}\left(\frac{1}{4}2\overrightarrow{OI_2}+\frac{w_{21}}{2}\overrightarrow{OP_{21}}\right) \\
&= \frac{2}{1+w_{21}}\left(\frac{1}{2}\overrightarrow{OI_2}+\frac{w_{21}}{2}\overrightarrow{OP_{21}}\right) = \frac{1}{1+w_{21}}\left(\overrightarrow{OI_2}+w_{21}\overrightarrow{OP_{21}}\right)
\end{aligned}$$

Result (14) of $\overrightarrow{OM\left(\frac{1}{2},0\right)}$ and $\overrightarrow{OM\left(\frac{1}{2},1\right)}$ can be proved in a similar way. □

Theorem 5. *Let G_0 be the isobarycentre of points P_{00}, P_{02}, P_{20}, P_{22} and G_2 the barycentre of weighted points (P_{10}, w_{10}), (P_{01}, w_{01}), (P_{12}, w_{12}), (P_{21}, w_{21}). We define the value $w = w_{01} + w_{10} + w_{12} + w_{21}$ and G_1 as the barycentre of the weighted points $(G_0, 2)$ and (G_2, w).*

The point $M\left(\frac{1}{2},\frac{1}{2}\right)$ satisfies the two following formulas:

$$\overrightarrow{OM\left(\frac{1}{2},\frac{1}{2}\right)} = \frac{1}{2+w+2w_{11}}\left((2+w)\overrightarrow{OG_1}+2w_{11}\overrightarrow{OP_{11}}\right) \tag{15}$$

$$w_{11}\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)P_{11}} = -\frac{2+w}{2}\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)G_1} \tag{16}$$

From the last formula we deduce that P_{11} belongs to the line $\left(M\left(\frac{1}{2},\frac{1}{2}\right)G_1\right)$.

Proof. Evaluating equation (2) of the RBBSs at the point $\overrightarrow{OM\left(\frac{1}{2},\frac{1}{2}\right)}$ gives:

$$\overrightarrow{OM\left(\frac{1}{2},\frac{1}{2}\right)} = \frac{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\left(\frac{1}{2}\right) B_j\left(\frac{1}{2}\right)\overrightarrow{OP_{ij}}}{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\left(\frac{1}{2}\right) B_j\left(\frac{1}{2}\right)}$$

First, let us consider the denominator of this fraction:

$$\begin{aligned}
&\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i\left(\frac{1}{2}\right) B_j\left(\frac{1}{2}\right) = \sum_{i=0}^{2} B_i\left(\frac{1}{2}\right)\left(\frac{w_{i0}}{4}+\frac{w_{i1}}{2}+\frac{w_{i2}}{4}\right)\\
&= \frac{\frac{w_{00}}{4}+\frac{w_{01}}{2}+\frac{w_{02}}{4}}{4} + \frac{\frac{w_{10}}{4}+\frac{w_{11}}{2}+\frac{w_{12}}{4}}{2} + \frac{\frac{w_{20}}{4}+\frac{w_{21}}{2}+\frac{w_{22}}{4}}{4}\\
&= \frac{\frac{w_{00}+w_{02}+w_{20}+w_{22}}{4} + \frac{w_{01}+w_{10}+w_{21}+w_{12}}{2}}{4} + \frac{\frac{w_{11}}{2}}{2}\\
&= \frac{w_{00}+w_{02}+w_{20}+w_{22}}{16} + \frac{w_{01}+w_{10}+w_{21}+w_{12}}{8} + \frac{w_{11}}{4}\\
&= \frac{2+w+2w_{11}}{8}
\end{aligned}$$

Second, the numerator can be similarly expressed as:

$$\begin{aligned}
&\sum_{i=0}^{2} B_i\left(\frac{1}{2}\right)\left(\frac{w_{i0}}{4}\overrightarrow{OP_{i0}}+\frac{w_{i1}}{2}\overrightarrow{OP_{i1}}+\frac{w_{i2}}{4}\overrightarrow{OP_{i2}}\right) =\\
&\frac{w_{00}}{16}\overrightarrow{OP_{00}}+\frac{w_{01}}{8}\overrightarrow{OP_{01}}+\frac{w_{02}}{16}\overrightarrow{OP_{02}}+\frac{w_{10}}{8}\overrightarrow{OP_{10}}+\frac{w_{11}}{4}\overrightarrow{OP_{11}}+\\
&\frac{w_{12}}{8}\overrightarrow{OP_{12}}+\frac{w_{i0}}{16}\overrightarrow{OP_{20}}+\frac{w_{21}}{8}\overrightarrow{OP_{21}}+\frac{w_{22}}{16}\overrightarrow{OP_{22}}\\
&= \left(\frac{1}{16}\overrightarrow{OP_{00}}+\frac{1}{16}\overrightarrow{OP_{02}}+\frac{1}{16}\overrightarrow{OP_{20}}+\frac{1}{16}\overrightarrow{OP_{22}}\right)+\\
&\left(\frac{w_{01}}{8}\overrightarrow{OP_{01}}+\frac{w_{10}}{8}\overrightarrow{OP_{10}}+\frac{w_{12}}{8}\overrightarrow{OP_{12}}+\frac{w_{21}}{8}\overrightarrow{OP_{21}}\right)+\frac{w_{11}}{4}\overrightarrow{OP_{11}}\\
&= \frac{1}{4}\overrightarrow{OG_0}+\frac{w}{8}\overrightarrow{OG_2}+\frac{w_{11}}{4}\overrightarrow{OP_{11}}
\end{aligned}$$

We may then write:

$$\begin{aligned}
\overrightarrow{OM\left(\frac{1}{2},\frac{1}{2}\right)} &= \frac{1}{\frac{2+w+2w_{11}}{8}}\left(\frac{1}{4}\overrightarrow{OG_0}+\frac{w}{8}\overrightarrow{OG_2}+\frac{w_{11}}{4}\overrightarrow{OP_{11}}\right)\\
&= \frac{8}{2+w+2w_{11}}\left(\frac{1}{4}\overrightarrow{OG_0}+\frac{w}{8}\overrightarrow{OG_2}+\frac{w_{11}}{4}\overrightarrow{OP_{11}}\right)\\
&= \frac{1}{2+w+2w_{11}}\left(2\overrightarrow{OG_0}+w\overrightarrow{OG_2}+2w_{11}\overrightarrow{OP_{11}}\right)\\
&= \frac{1}{2+w+2w_{11}}\left((2+w)\overrightarrow{OG_1}+2w_{11}\overrightarrow{OP_{11}}\right)
\end{aligned}$$

which proves the result (15). This latter is valid for any point O so it is valid for the point $M\left(\frac{1}{2},\frac{1}{2}\right)$. Hence, substituting O by $M\left(\frac{1}{2},\frac{1}{2}\right)$ in (15) gives:

$$\begin{aligned}
(2+w+2w_{11})\ &\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)M\left(\frac{1}{2},\frac{1}{2}\right)} =\\
&(2+w)\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)G_1}+2w_{11}\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)P_{11}}
\end{aligned}$$

Consequently, we have: $\overrightarrow{0} = (2+w)\,\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)G_1} + 2w_{11}\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)P_{11}}$. Hence:

$$w_{11}\overrightarrow{M\left(\tfrac{1}{2},\tfrac{1}{2}\right)P_{11}} = -\frac{2+w}{2}\overrightarrow{M\left(\tfrac{1}{2},\tfrac{1}{2}\right)G_1} \qquad \square$$

Theorem 6. *Let G_3 be the barycentre of the weighted points $(P_{00}, 9)$, $(P_{20}, 9)$, $(P_{02}, 1)$, $(P_{22}, 1)$, $(P_{01}, 6w_{01})$, $(P_{21}, 6w_{21})$, $(P_{10}, 18w_{10})$, $(P_{12}, 2w_{12})$, and $W_1 = 20 + 6w_{01} + 18w_{10} + 2w_{12} + 6w_{21}$.*
The point $M\left(\frac{1}{2},\frac{1}{4}\right)$ satisfies the two following formulas:

$$\overrightarrow{OM\left(\tfrac{1}{2},\tfrac{1}{4}\right)} = \tfrac{1}{W_1+12w_{11}}\left(W_1\overrightarrow{OG_3} + 12w_{11}\overrightarrow{OP_{11}}\right) \qquad (17)$$

$$(W_1 + 12w_{11})\,\overrightarrow{G_3M\left(\tfrac{1}{2},\tfrac{1}{4}\right)} = 12w_{11}\overrightarrow{G_3P_{11}} \qquad (18)$$

From the last formula we deduce that P_{11} belongs to the line $\left(G_3M\left(\frac{1}{2},\frac{1}{4}\right)\right)$.

Proof. Before starting the proof of result (17) for point $\overrightarrow{OM\left(\frac{1}{2},\frac{1}{4}\right)}$, recall that $B_0\left(\frac{1}{4}\right) = \frac{9}{16}$, $B_1\left(\frac{1}{4}\right) = \frac{3}{8}$ and $B_2\left(\frac{1}{4}\right) = \frac{1}{16}$. By formula (2) we have:

$$\overrightarrow{OM\left(\frac{1}{2},\frac{1}{4}\right)} = \frac{1}{\displaystyle\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i\left(\frac{1}{2}\right)B_j\left(\frac{1}{4}\right)}\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i\left(\frac{1}{2}\right)B_j\left(\frac{1}{4}\right)\overrightarrow{OP_{ij}}$$

The denominator can be easily reduced to:

$$\begin{aligned}
\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i\left(\frac{1}{2}\right)B_j\left(\frac{1}{4}\right) &= \sum_{i=0}^{2} B_i\left(\frac{1}{2}\right)\left(\frac{9w_{i0}}{16} + \frac{6w_{i1}}{16} + \frac{w_{i2}}{16}\right)\\
&= \frac{9w_{00} + 9w_{20} + w_{02} + w_{22} + 6w_{01} + 6w_{21} + 18w_{10} + w_{12} + 12w_{11}}{64}\\
&= \frac{20 + 6w_{01} + 6w_{21} + 18w_{10} + w_{12} + 12w_{11}}{64}\\
&= \frac{W_1 + 12w_{11}}{64}
\end{aligned}$$

Using this result and expanding the numerator gives the required proof:

$$\begin{aligned}
\overrightarrow{OM\left(\frac{1}{2},\frac{1}{4}\right)} &= \frac{64}{W_1+12w_{11}}\left(\frac{9}{64}\overrightarrow{OP_{00}} + \frac{6w_{01}}{64}\overrightarrow{OP_{01}} + \frac{w_{02}}{64}\overrightarrow{OP_{02}}\right) +\\
&\quad \frac{64}{W_1+12w_{11}}\left(\frac{18}{64}\overrightarrow{OP_{10}} + \frac{12w_{11}}{64}\overrightarrow{OP_{11}} + \frac{2w_{02}}{64}\overrightarrow{OP_{12}}+\right) +\\
&\quad \frac{64}{W_1+12w_{11}}\left(\frac{9}{64}\overrightarrow{OP_{20}} + \frac{6w_{21}}{64}\overrightarrow{OP_{21}} + \frac{1}{64}\overrightarrow{OP_{22}}\right)\\
&= \frac{1}{W_1+12w_{11}}\left(W_1\overrightarrow{OG_3} + 12w_{11}\overrightarrow{OP_{11}}\right)
\end{aligned}$$

From these results we deduce:

$$\left(W_1 + 12w_{11}\right) \overrightarrow{G_3M\left(\frac{1}{2}, \frac{1}{4}\right)} = 12w_{11}\overrightarrow{G_3P_{11}}$$

which is the result (18). □

Theorem 7. *Let G_4 be the barycentre of the weighted points $(P_{00}, 9)$, $(P_{20}, 1)$, $(P_{02}, 9)$, $(P_{22}, 1)$, $(P_{10}, 6w_{10})$, $(P_{12}, 6w_{12})$, $(P_{01}, 18w_{01})$, $(P_{21}, 2w_{21})$, and $W_2 = 20 + 6w_{10} + 18w_{01} + 2w_{21} + 6w_{12}$.*

The point $M\left(\frac{1}{4}, \frac{1}{2}\right)$ satisfies the two following formulas:

$$\overrightarrow{OM\left(\tfrac{1}{4}, \tfrac{1}{2}\right)} = \tfrac{1}{W_2+12w_{11}} \left(W_2\overrightarrow{OG_4} + 12w_{11}\overrightarrow{OP_{11}}\right) \tag{19}$$

$$\left(W_2 + 12w_{11}\right) \overrightarrow{G_4M\left(\tfrac{1}{4}, \tfrac{1}{2}\right)} = 12w_{11}\overrightarrow{G_4P_{11}} \tag{20}$$

From the last formula we deduce that P_{11} belongs to the line $\left(G_4M\left(\frac{1}{4}, \frac{1}{2}\right)\right)$.

Proof. The results (19) and (20) can be proved in the same way as (17) and (18) or (15) and (16). □

5 A New Dupin Cyclide to RBBS Conversion Algorithm

In this section we propose a new algorithm (Algorithm 1) for converting a Dupin cyclide patch or a whole Dupin cyclide into RBBS form. The algorithm is based on the barycentric properties of RBBSs given by the theorems of Sections 3 and 4.

The goal is to develop a conversion method that exploits and keeps circular symmetries along the isoparametric curves of Dupin cyclides. These latter are not surfaces of revolution, so, in step six of this algorithm, we have to be sure that lines $(G_1\Gamma(\theta_2, \psi_2))$, $(G_3\Gamma(\theta_2, \psi_3))$ and $(G_4\Gamma(\theta_3, \psi_2))$ have the same point of intersection. This is true provided that θ_0 and θ_1 (or ψ_0 and ψ_1) are symmetrical compared to 0 or π on the trigonometric circle.

A set of conversion examples is presented in the following sections.

Algorithm 1

> ***Given:*** *A Dupin cyclide defined by a parametric map Γ, and a patch on this cyclide delimited by parameter values: $\theta_0, \theta_1, \psi_0$ and ψ_1 with $|\theta_0 - \theta_1| < \pi$ and $|\psi_0 - \psi_1| < \pi$. These values also define four (circular) curvature lines on the Dupin cyclide*
>
> ***Find:*** *The representation of this patch as a RBBS S over $[0,1]\times[0,1]$, defined by nine control points P_{ij} and nine weights w_{ij}, $0 \leqslant i, j < 2$. Weights w_{ij} will be equal to one except w_{10}, w_{01}, w_{21} , w_{12} and w_{11}, which can be either positive or negative.*

Procedure:

1. *Obtain corner control points using:* $P_{00} = \Gamma(\theta_0, \psi_0)$, $P_{02} = \Gamma(\theta_1, \psi_0)$, $P_{20} = \Gamma(\theta_0, \psi_1)$, $P_{22} = \Gamma(\theta_1, \psi_1)$.
2. *Find centres of the four circles of curvature using a point on each line of curvature.*
3. *Find control points* P_{01}, P_{10}, P_{12} *and* P_{21} *on the median planes of curvature lines using equation (9) of Theorem 2.*
4. *Calculate weights* w_{10}, w_{01}, w_{21} *and* w_{12} *using equations (11) or (12) of Theorem 3.*
5. *Consider two RQBC* γ_u *and* γ_v *defined on the borders of the region to be converted. Control points and the weight of* γ_u *are* P_{00}, P_{10}, P_{20} *and* w_{10}. *Those of* γ_v *are* P_{00}, P_{01}, P_{02} *and* w_{01}.
   - *Find* θ_2 *and* θ_3, *the solutions of equations* $\gamma_u(\frac{1}{2}) = \Gamma(\theta_2, \psi_0)$ *and* $\gamma_u(\frac{1}{4}) = \Gamma(\theta_3, \psi_0)$.
   - *Find* ψ_2 *and* ψ_3, *the solutions of equations* $\gamma_v(\frac{1}{2}) = \Gamma(\theta_0, \psi_2)$ *and* $\gamma_v(\frac{1}{4}) = \Gamma(\theta_0, \psi_3)$.
6. *Compute control point* P_{11} *as the intersection of lines* $(G_1\Gamma(\theta_2, \psi_2))$ *and* $(G_3\Gamma(\theta_2, \psi_3))$. *Constructions of* G_1 *and* G_3 *are given in Theorems 5 and 6.*
7. *Compute weight* w_{11}, *using equation (16) of Theorem 7*

5.1 Conversion Examples

Figure 7 illustrates the conversions of two Dupin cyclide patches into RBBSs using the proposed algorithm. For the first example (the upper subfigures), we obtained a standard RBBS having only positive weights. For the second one (the

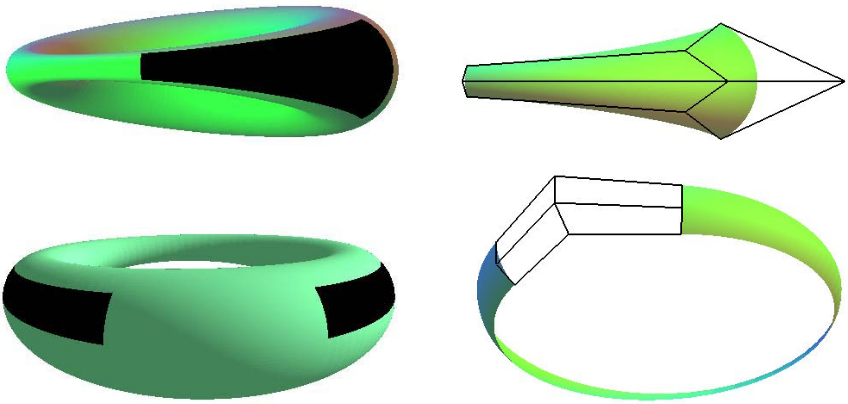


Fig. 7. Conversion of two cyclide patches into RBBSs. Upper, the result is a standard RBBS having only positive weights. Lower, the result is a RBBS with positive and negative weights

lower subfigures), in which the cyclide patch is delimited by $\psi_1 = \pi$, we obtained a RBBS with positive and negative weights. These are positive along curvature lines with θ constant, and negative along curvature lines with ψ constant.

5.2 Comparison of Conversion Algorithms

To make a significant comparison of the various algorithms for converting Dupin cyclides to RBBS form (Pratt's original algorithm, the new variant of the Pratt's algorithm and the barycentric algorithm), we have used the following criteria:

- Entire cyclide: whether it is possible or not to convert a whole Dupin cyclide into a set of RBBSs.
- Number of RBBSs: The minimum number of RBBSs needed to represent a whole Dupin cyclide.
- Parameter constraints: what are the constraints on parameters θ_0, θ_1, ψ_0 and ψ_1?
- Prohibited values: Are there particular prohibited values for parameters θ_0, θ_1, ψ_0 and ψ_1?

Table 1 summarizes this comparison. One can note that to convert a whole Dupin cyclide we need: Only four RBBSs if the new algorithm is used, nine RBBSs if the Pratt's algorithm is used, and six RBBSs if the Pratt's algorithm and its variant are combined and used. Figure 8 shows this difference in the number of resulting RBBSs.

Dupin cyclides are very useful for blending quadric surfaces [8, 13, 14, 15]. To show conversion of a blending Dupin cyclide we applied the method proposed in [8] to construct the blending cyclide of the left subfigure of Figure 9 and converted it to RBBSs by the three conversion algorithms discussed in this paper. We note here that conversion showed in the middle subfigure can be obtained either by the new algorithm or by a combination of Pratt's algorithm and its variant; two RBBSs are enough to ensure a correct conversion. On the other hand, Pratt's algorithm gives conversion showed on the right subfigure. Although in this case,

Table 1. Comparison between Pratt's algorithm, its variant and the barycentric algorithm

	Pratt's algorithm	Variant of Pratt's algorithm	Barycentric algorithm
Entire cyclide	no	yes	yes
Number of RBBSs	/	9	4
Geometric constraints	no	$\begin{cases} \lvert\theta_0 - \theta_1\rvert < \pi \\ \lvert\psi_0 - \psi_1\rvert < \pi \end{cases}$	(θ_0, θ_1) or (ψ_0, ψ_1) symmetrical compared to 0 or π.
Prohibited value	π	π	no

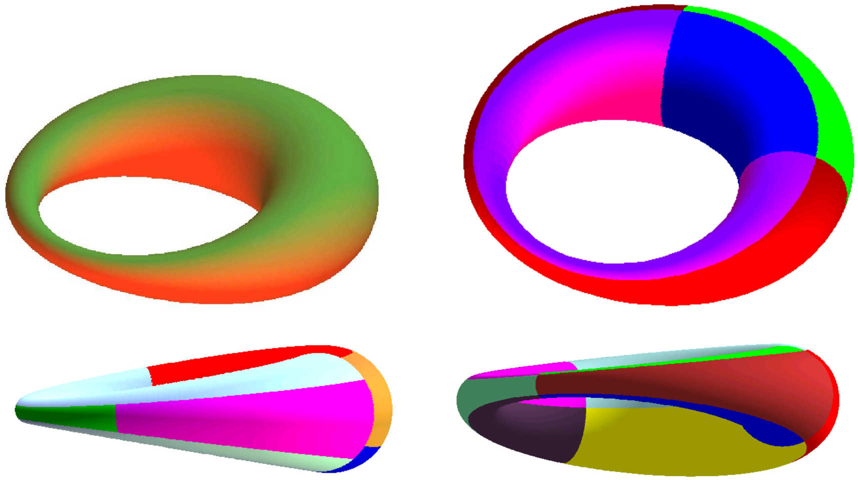


Fig. 8. Comparison of conversion algorithms. Upper left, the cyclide to be converted. Upper right, result of the new algorithm: only four RBBSs are necessary. Lower left, result of the combination of Pratt's algorithm and its variant, six RBBSs are necessary. Lower right, result of the variant of Pratt's algorithm, nine RBBSs are necessary

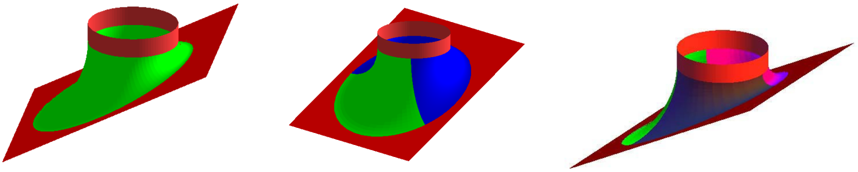


Fig. 9. Conversion of cylinder/plane blending Dupin cyclide. Left, the blending cyclide. Middle, conversion into two RBBSs. Right, conversion into three RBBSs

the minimal number of RBBSs is three, the surfaces obtained here have only positive weights so they lie completely inside their control polyhedron.

Another case where the new algorithm is better than both of others is shown by Figure 10. The cyclide patch (left subfigure) is the one of Figure 5 where Pratt's algorithm and its variant have given incorrect conversions. The new algorithm exactly reproduces the original cyclide patch (Figure 10 right).

Another advantage of the new algorithm is the possibility of converting a Dupin cyclide patch delimited by a curvature line corresponding to a parameter value of π, which is not feasible either with Pratt's algorithm or its variant. This is shown in Figure 11 where the cyclide patch to be converted is a blending surface of a cylinder and a plane. Horizontally, it is delimited, on the cylinder side by $\psi_1 = \frac{\pi}{2}$ and on the plane side by $\psi_0 = \pi$. Neither Pratt's algorithm nor its variant can be used to achieve this conversion. Two samples of possible conversions, using the barycentric algorithm, are given on subfigures 11 middle and 11 right.

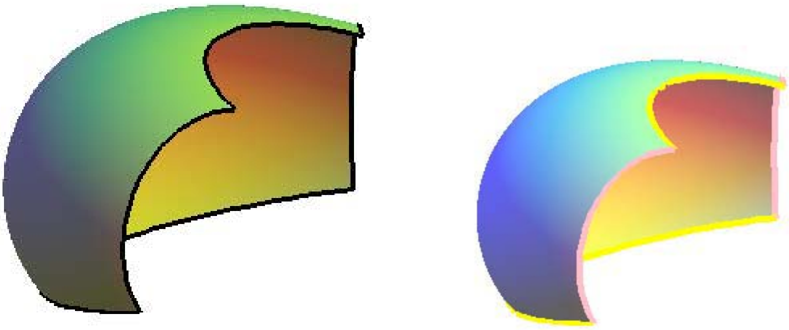


Fig. 10. Another conversion result. The new algorithm gives exact conversion (right subfigure), while other algorithms fail (see Figure 5)

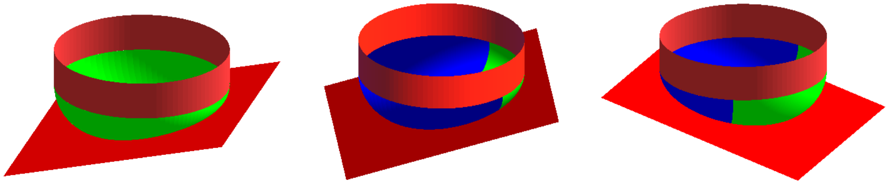


Fig. 11. Conversion of cylinder/plane blending Dupin cyclide where neither Pratt's algorithm nor its variant can be applied. Left, the blending cyclide. Middle and right, two examples of possible conversions

5.3 A Complete Case Study

As control points of RBBSs resulting from conversion may be far removed from the patches themselves, it is often difficult to draw the control polygon on figures that show these RBBSs. So, in this section we give more details concerning the conversion of the cyclide patch of Figure 5 left, and the RBBSs given by the conversion algorithms. Parameters defining the cyclide patch are $a = 6$, $c = 2$, $\mu = 3$, $\theta_0 = \frac{2\pi}{3}$, $\theta_1 = \frac{4\pi}{3}$, $\psi_0 = \frac{5\pi}{6}$ and $\psi_1 = \frac{3\pi}{2}$. Table 2 gives control points and

Table 2. Comparison of control points obtained with Pratt's algorithm and barycentric algorithm

[Points;Weights]	Pratt's algorithm	barycentric algorithm
$[P_{00}; w_{00}] \simeq$	$[(-3.466; 8.206; -2.204); 306.564]$	$[(-3.466; 8.206; -2.204), 1]$
$[P_{01}; w_{01}] \simeq$	$[(-5.995; 12.852; 5.302); -46.641]$	$[(-5.995; 12.852; 5.302); 0.384]$
$[P_{02}; w_{02}] \simeq$	$[(-1.667; 4.899; 3.771); 48.000]$	$[(-1.667; 4.899; 3.771); 1]$
$[P_{10}; w_{10}] \simeq$	$[(-35.166; -0.191; -7.807); -75.713]$	$[(-35.166; -0.422; -7.807); 0.247]$
$[P_{11}; w_{11}] \simeq$	$[(129.663; 0.682; -42.661); -5.072]$	$[(129.663; 0; -42.661); -0.042]$
$[P_{12}; w_{12}] \simeq$	$[(-9.667; -0.566; 6.600); -24.000]$	$[(-9.667; -0.117; 6.600); 0.500]$
$[P_{20}; w_{20}] \simeq$	$[(-3.466; -8.205; -2.204); 306.564]$	$[(-3.466; -8.205; -2.204); 1]$
$[P_{21}; w_{21}] \simeq$	$[(-5.995; -12.852; 5.302); -46.641]$	$[(-5.995; -12.852; 5.302); 0.384]$
$[P_{22}; w_{22}] \simeq$	$[(-1.667; -4.899; 3.771); 48.000]$	$[(-1.667; -4.899; 3.771); 1]$

weights of RBBSs of Figure 5 middle and Figure 10 right. This table shows how computed control points differ from one algorithm to another. A value of 1 is an exact value, whereas a value of 48.000 is a rounded of value. The only difference between Pratt's algorithm and its variant is the fact that negative weights become positive. On the other hand, weights computed using our algorithm are smaller than those given by Pratt's algorithm; moreover, the new algorithm gives exact values for corner control points, unlike the other algorithms.

6 Conclusion

In this paper, we have have used the symmetry properties of circles and Bernstein polynomials to prove seven theorems concerning barycentric properties of RBBSs. These properties have been employed in a new algorithm for the conversion of Dupin cyclides to RBBSs. Several examples of the use of the algorithm have been given.

In future it is proposed to extend the algorithm for the conversion of supercyclides (affine or projective transformations of Dupin cyclides) to RBBSs.

References

1. Piegl, L., Tiller, W.: A menagerie of rational B-spline circles. IEEE Computer Graphics and Applications **9** (1989) 46–56
2. Forrest, A.R.: Curves and Surfaces for Computer-Aided Design. PhD thesis, University of Cambridge, UK (1968)
3. Demengel, G., Pouget, J.: Mathématiques des Courbes et des Surfaces: Modèles de Bézier, des B-Splines et des NURBS. Volume 1. Ellipse (1998)
4. Hoschek, J., Lasser, D.: Fundamentals of Computer Aided Geometric Design. A.K.Peters, Wellesley, MA. (1993)
5. Pratt, M.J.: Cyclides in computer aided geometric design. Computer Aided Geometric Design **7** (1990) 221–242
6. Chandru, V., Dutta, D., Hoffmann, C.M.: Variable radius blending using Dupin cyclides. In Wozny, M., Turner, J.U., Preiss, K., eds.: Geometric Modelling for Product Engineering, North-Holland Publ. Co. (1990) (Proc. IFIP/NSF Workshop on Geometric Modelling, Rensselaerville, NY, Sept 1988).
7. Degen, W.L.F.: Generalized cyclides for use in CAGD. In Bowyer, A., ed.: Computer-Aided Surface Geometry and Design, Oxford University Press (1994) 349–363 (Proc. 4th IMA Conference on the Mathematics of Surfaces, Bath, UK, Sept. 1990).
8. Pratt, M.J.: Cyclides in computer aided geometric design II. Computer Aided Geometric Design **12** (1995) 131–152
9. Dutta, D., Martin, R.R., Pratt, M.J.: Cyclides in surface and solid modeling. IEEE Computer Graphics and Applications **13** (1993) 53–59
10. Paluszny, M., Boehm, W.: General cyclides. Computer Aided Geometric Design **15** (1998) 699–710
11. Shene, C.: Blending two cones with Dupin cyclides. Computer Aided Geometric Design **15** (1998) 643–673
12. Farin, G.: Curves And Surfaces. 3 edn. Academic Press (1993)

13. Aumann, G.: Curvature continuous connections of cones and cylinders. Computer-aided Design **27** (1995) 293–301
14. Shene, C.: Do blending and offsetting commute for Dupin cyclides? Computer Aided Geometric Design **17** (2000) 891–910
15. Wallner, J., Pottmann, H.: Rational blending surfaces between quadrics. Computer Aided Geometric Design **14** (1997) 407–419

Multi-sided Attribute Based Modeling

Kun Gao and Alyn Rockwood

D*Syn Corp, Black Forest, Colorado, USA
{alynrockwood, kungao}@dsyn-soft.com

Abstract. We consider the problem of defining multi-sided patches in a system that enables G^2 continuity. The technology is based on finding the weighted least squares solution of points on given input curves where a separate parameter space with control structures determines its weights. It is a generalization of Shepard's method to a parameterized vector solution. The method generates surface patches that satisfy certain minimal energy conditions. it employs any parametric curve and points as controls for defining the surface.

1 Introduction

Multi-sided patches have been a commonly researched topic in geometric design. Because of their facility to match surface patches to the underlying topology of an object, they can more closely fit the designer's vision, thus relieving the designer of the "topology burden," and allowing him or her to focus on the important, shape-defining attributes of the object. A good survey of multi-sided patches with references is found in [5]. More recent developments in multi-sided patches include recursive subdivision methods adapted to interpolate curves where a match between the curves and a sequence of edges on the base control mesh is assumed [4]. Recently [7] and [8] describe a method called T-splines that subsumes NURBS and Catmull-Clark surfaces, and admits T-junctions, which can handle many of the topological arrangements that occur in design. Bi-cubic T-splines are C^2 piecewise rational surfaces that are backward compatible with NURBS surfaces.

We describe a technique that defines multi-sided patches that interpolate arbitrary parametric curves. Since the patches allow a designer to focus on attribute curves with less concern about the topology we call it *attribute based modeling*^{TM}.

2 Fundamental Surface Definition

Given points $\mathbf{x_i} = (x_i, y_i, z_i)$, the weighted, discrete least squares solution $\mathbf{x} = (x, y, z)$ minimizes

$$\sum_i [(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2] W_i(x, y, z) \tag{1}$$

where the solution slews toward the points with larger weights $W_i(x, y, z)$ [1].

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 219–232, 2005.

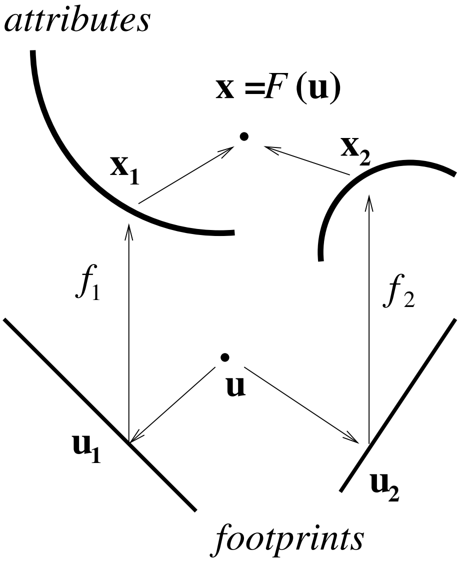


Fig. 1. Map **u** to **x** via footprints and attributes

Attribute based modeling starts with a parametric map on $\mathbb{R}^2$ that supplies the weights to interpolate points and curves in $\mathbb{R}^3$ using Formula 1, as illustrated in Fig 1. First, let U_i be simple objects in $\mathbb{R}^2$, e.g. line segments, called footprints. For any $\mathbf{u}$ in $\mathbb{R}^2$, let $P_i(\mathbf{u})$ be a projection onto a unique point $\mathbf{u_i}$ of each footprint U_i. For each footprint there is an attribute function f_i that maps from U_i to $\mathbb{R}^3$. We call the $f_i(U_i)$ attributes, e.g. the curves in attribute space. For any parameter $\mathbf{u}$ let $\mathbf{x} = F(\mathbf{u})$ in $\mathbb{R}^3$ be the weighted least squares solution for points $\mathbf{x_i}$=$f_i(\mathbf{u_i})$ where the weights are typically given as reciprocal distances from $\mathbf{u}$ to their projections $\mathbf{u_i}$. Potential singularity issue is discussed in Section 6. There are a number of ways to project to footprints and compute distances between $\mathbf{u}$ and the footprints, e.g. [6]. Some are explored later. Distance to footprint is a significant computation; thus there is strong motivation for keeping the footprints simple, such as points and line segments. The distance must also be chosen to satisfy desired continuity conditions. This is also discussed later.

The critical role that distances play is to make the weight large as $\mathbf{u}$ approaches a footprint; it forces the surface $F(\mathbf{u})$ to approach the corresponding attribute, and that guarantees interpolation. Thus in Fig 2 as $\mathbf{u}$ approaches $\mathbf{u_1}$, the corresponding points of the surface approach $\mathbf{x_1}$. Letting $\mathbf{u}$ vary yields the interpolation surface, which is displayed above the parameter space in this case. The surface mimics the behavior of the attribute curves in the nearby region.

Formula 1 characterizes the interpolant as a least-squares "minimal energy" surface with respect to weights in footprint space. This gives it a tight film-like appearance.

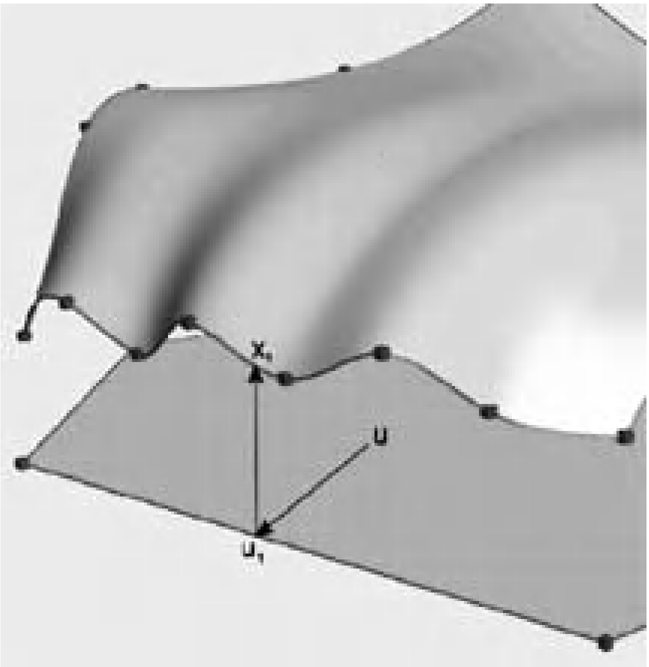


Fig. 2. Map **u** to **x** via footprints and attributes

However, a simpler algebraic form exists than in Formula 1. To see this, let the attribute functions be $f_i(\mathbf{u_i})$=$(x_i(\mathbf{u}),\ y_i(\mathbf{u}),z_i(\mathbf{u}))$. From Formula 1 we seek $\mathbf{x} = F(\mathbf{u}) = (x, y, z)$ that minimizes

$$\begin{aligned} S =& \sum_i \|f_i(\mathbf{u_i}) - F(\mathbf{u})\|^2 \, W_i(\mathbf{u}) \\ =& \sum_i [(x_i(\mathbf{u}) - x)^2 + (y_i(\mathbf{u}) - y)^2 + (z_i(\mathbf{u}) - z)^2] W_i(\mathbf{u}) \end{aligned} \tag{2}$$

with respect to (x, y, z). Without loss of generality consider the partial derivative of x as follows:

$$\partial S/\partial x = \sum_i [-2(x_i(\mathbf{u}) - x)W_i(\mathbf{u}) + (x_i(\mathbf{u}) - x)^2 \partial W_i(\mathbf{u})/\partial x] \tag{3}$$

which is equal to

$$\sum_i [-2(x_i(\mathbf{u}) - x)W_i(\mathbf{u})] \tag{4}$$

since $\partial W_i(\mathbf{u})/\partial x$=0. Minimizing Formula 2 by setting Formula 4 to 0 yields

$$\sum_i x_i(\mathbf{u})W_i(\mathbf{u}) = x \sum_i W_i(\mathbf{u}) \tag{5}$$

which implies

$$x = \sum_i x_i(\mathbf{u}) W_i(\mathbf{u}) / \sum_i W_i(\mathbf{u}) \tag{6}$$

In general we obtain

$$F(\mathbf{u}) = (x, y, z) = \sum_i f_i(\mathbf{u_i}) W_i(\mathbf{u}) / \sum_i W_i(\mathbf{u}) = \sum_i f_i(\mathbf{u_i}) \hat{W}_i(\mathbf{u}) \tag{7}$$

where $\hat{W}_i(\mathbf{u}) = W_i(\mathbf{u}) / \sum_k W_k(\mathbf{u})$, and k ranges over the number of footprints.

One will recognize Formula 7 as a normalized, linear combination seen in methods such as Shepard's formula, rational Bezier forms, NURBS and the "Wires" method ([3] and [9]). A crucial difference between Formula 7 and Shepard's formula is that in Formula 7 the attribute functions f_i and the weights W_i are computed in a separate parameter space, i.e. the footprint space, and the weights are determined by distance to freely definable footprints. The parameterization avoids the known problems of Shepard's form, e.g. flat spots at interpolating points. Hence, unlike Shepard's method, attribute based modeling has *linear precision*, that is, if interpolation points or attribute curves are restricted to a straight line, then the interpolant will be a straight line. This follows immediately from the convex combination in Formula 7. Furthermore, Shepard's method only interpolates points, while Formula 7 also interpolates curves.

The "Wires" method fosters the concept of attribute based modeling, akin in spirit to our technique. The Wires' paper has an example of underlying geometry that maps to curves in space, from which one could get geometry to conform to the curves. However, it makes no claims of minimal energy and could introduce unwanted wiggles [9].

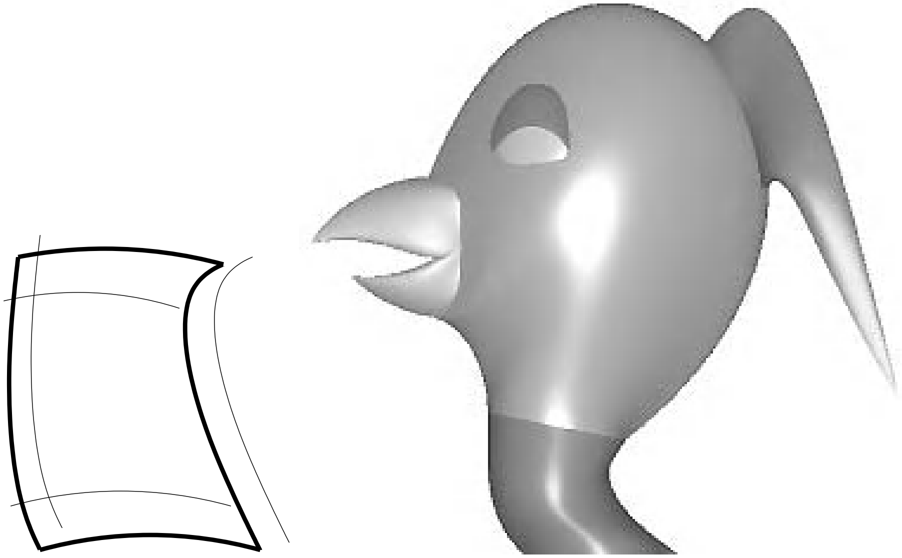
3 Adding Derivative Information

The resemblance in form to Shepard's Formula suggests useful enhancements to Formula 7. The f_i can be replaced by the first few terms of a Taylor series, for example. Consequently the interpolant passes through the attributes position and also matches derivatives in the series. Another way to extend Formula 7 that matches tangencies is to infer the partials from a lofted surface

$$R_i(s_i, t_i) = (1 - s_i) f_i(t_i) + s_i g_i(t_i) \tag{8}$$

The parameter s_i parameterizes the i^{th} loft, where $s_i = d_i(\mathbf{u})$, is some "distance" of $\mathbf{u}$ to the i^{th} footprint U_i. The $g_i(t)$ are user given parametric curves that define the boundary of the loft. The parameter t_i is the parameter value of the projection $P_i(\mathbf{u})$ on the i^{th} footprint, and is used to parameterize the footprint (see Section 5). With s_i and t_i given as functions of $\mathbf{u}$ we can give the variation of Formula 7 that is used for G^1 in this paper as

$$F(\mathbf{u}) = \sum_i R_i(s_i, t_i) \hat{W}_i(\mathbf{u})^2 \tag{9}$$



(a) Attribute curves (bold) and loft boundary curves (thin) of the neck

(b) Subsequent bird

Fig. 3. Seven-patch cartoon bird

Notice that $\hat{W}_i(\mathbf{u})$ is squared to satisfy continuity requirements, which is explained in the G^1 proof that follows.

Fig 3(a) shows both position attributes $f_i(t)$ and offset boundary curves $g_i(t)$ that defines the surface of a bird's neck. The model contains two two-sided patches for the eye and eyelid, three three-sided patches for the beaks and plume and two four-sided patches for the head and neck. The number of sides on each patch matches a natural connection topology of the object being modeled.

Rewrite Formula 9 as

$$F(\mathbf{u}) = \sum_i [(1-s_i)f_i(t_i) + s_i g_i(t_i)]\hat{W}_i(\mathbf{u})^2 \tag{10}$$

and consider the partial derivatives of Formula 10. For $\mathbf{u} = (u, v)$ and without loss of generality, let all following partials be with respect to u:

$$\begin{aligned}\partial F(u,v) = \sum_i \{&[-f_i(t_i) + g_i(t_i)]\hat{W}_i(u,v)^2 \\ &+ [(1-s_i)f_i(t_i) + s_i g_i(t_i)]2\hat{W}_i(u,v)\partial\hat{W}_i(u,v)\}\end{aligned} \tag{11}$$

We assume that s_i and t_i are affine functions of (u,v), so that their partials are constant, which can be ignored in the following arguments below. This is not a great restriction on the distance and projection calculations in the method (see Section 5).

It remains to examine $\partial \hat{W}_i(u,v)$. Recall that W_i are reciprocal functions of the distances $s_i = d_i(u,v)$, which are affine functions with respect to both variables:

$$W_i(u,v) = 1/d_i(u,v) \tag{12}$$

It can be shown by induction that for $1 < k < n+1$ footprints:

$$\begin{aligned}\partial \hat{W}_i(u,v) =& [2\prod_k d_k(u,v)]\mathrm{Pol}(d_1(u,v),...,d_k(u,v),\partial d_1(u,v),...,\partial d_k(u,v))\\ &/[\sum_j \prod_{k\neq j} d_k(u,v)^2]^2\end{aligned} \tag{13}$$

where $\mathrm{Pol}(d_1(u,v),...,d_k(u,v),\partial d_1(u,v),...,\partial d_k(u,v))$ is a multivariate polynomial in the distances and partials. Note that Formula 13 always contains a multiplicative factor of $d_i(u,v)$ in the numerator, while the denominator always contains an additive term without $d_i(u,v)$. For example, with 4 footprints and $i=1$, we obtain

$$\frac{2d_1d_2d_3d_4\mathrm{Pol}(d_1,d_2,d_3,d_4,d_1{}',d_2{}',d_3{}',d_4{}')}{(d_1{}^2d_2{}^2d_3{}^2 + d_1{}^2d_2{}^2d_4{}^2 + d_1{}^2d_3{}^2d_4{}^2 + d_2{}^2d_3{}^2d_4{}^2)^2} \tag{14}$$

Formula 13 implies that for (u_0,v_0) on the i^{th} footprint where $d_i(u_0,v_0)=0$, and for $k\neq i$, $d_i(u_0,v_0)$ is a finite, non-zero distance. Hence, the denominator is non-zero and the numerator is zero, so $\partial \hat{W}_i(u_0,v_0) = 0$. This yields:

$$\partial F(u_0,v_0) = \sum_i [-f_i(t_i(u_0,v_0)) + g_i(t_i(u_0,v_0))]\hat{W}_i(u_0,v_0)^2 \tag{15}$$

Formula 15 is the same form as Formula 7 except that the weights are squared. It is an interpolation form, which means that for any point (u_0,v_0) on the i^{th} footprint in Formula 15 becomes

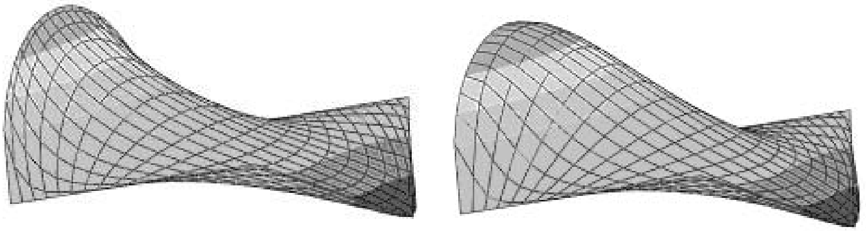
$$[g_i(t_i(u_0,v_0)) - f_i(t_i(u_0,v_0))] \tag{16}$$

which is the tangent of the loft $R_i(s_i,t_i)$ at (u_0,v_0). That is to say the surface is tangent to the loft at (u_0,v_0). This is true for the partial derivatives of both u and v by symmetry. Therefore we have:

Theorem 1. *If $F(\mathbf{u}) = \sum_i R_i(s,t)\hat{W}_i(\mathbf{u})^2$ are defined with multiple footprints, where*
$R_i(s,t) = (1-s)f_i(t) + sg_i(t)$, then

$$\partial F(u_0,v_0)/\partial u = \partial R_i(0,t_0)/\partial u, \qquad \partial F(u_0,v_0)/\partial v = \partial R_i(0,t_0)/\partial v \tag{17}$$

for any point (u_0,v_0) on the footprint and any $t_0 = t(u_0,v_0)$, which is an affine of u and v.



(a) Without lofts

(b) Add a horizontal loft to the semicircular attribute curve

Fig. 4. Five-sided, contour rendered patch

Theorem 1 guarantees that if two patches with the form in Formula 7 share a common curve $f_i(t)$ and have two lofts that share tangency at the common curve, the surface patches also share common tangency; they are G^1 at $f_i(t)$. Fig 4(a) shows a five-sided patch. In Fig 4(b) the same patch is associated with a horizontally displace lofted surface and blended to give it a fuller appearance.

4 $\mathbf{G^2}$ Continuity

The proof of G^2 continuity is very similar in form to that of the G^1 proof, except that power of the weights is cubic and the lofts are quadratic. For example one may use the Bernstein blending functions to define the loft:

$$Q_i(s,t) = (1-s)^2 f_i(t) + 2(1-s)sg_i(t) + s^2 h_i(t) \tag{18}$$

We will consider two lofts given as in Formula 18 for each i^{th} footprint, namely $QL_i(s,t)$ and $QR_i(s,t)$, such that $QL_i(0,t){=}QR_i(0,t){=}f_i(t)$. This yields two surfaces L and R defined as

$$L(\mathbf{u}) = \sum_i QL_i(s,t)\hat{W}_i(\mathbf{u})^3 \tag{19}$$

and

$$R(\mathbf{u}) = \sum_i QR_i(s,t)\hat{W}_i(\mathbf{u})^3 \tag{20}$$

Assume that the lofts are designed so that for any fixed value of t_0 the curves $QL_i(s,t_0)$ and $QR_i(s,t_0)$ are G^2 continuous at $s=0$, i.e. they share the same curvature at $s=0$. Let us consider all values of $\mathbf{u}$ that project to the same t_0 on the i^{th} footprint for $L(\mathbf{u})$, that is, the line of points perpendicular to the footprint at t_0. The parameter s is the Euclidean distance of $\mathbf{u}$ to the footprint and is therefore the distance along this perpendicular. Notice again that $s = s(\mathbf{u})$ is an affine function. We take the directional derivatives of L along s for a fixed

t_0, but since s is affine it is tantamount to a univariate derivative. Using the blending function from Formula 18 in Formula 19 we get

$$\begin{aligned} L'(s,t_0) = \textstyle\sum_i [QL_i(s,t_0) 3\hat{W}_i(s)^2 \hat{W}_i{}'(s) \\ + (-2(1-s)f_i(t_0) + (2-4s)g_i(t_0) + 2sh_i(t_0))\hat{W}_i(s)^3] \end{aligned} \tag{21}$$

For $s = 0$

$$L'(0,t_0) = \sum_i [QL_i(0,t_0) 3\hat{W}_i(0)^2 \hat{W}_i{}'(0) + (-2f_i(t_0) + 2g_i(t_0))\hat{W}_i(0)^3] \tag{22}$$

By the same steps as in the G^1 case in Formula 13 above we obtain

$$L'(0,t_0) = \sum_i [2g_i(t_0) - 2f_i(t_0))\hat{W}_i(0)^3] \tag{23}$$

which is tangent to $QL_i'(s,t_0)$. Likewise $R'(s)$ is parallel to $QR_i'(s,t_0)$ as $s \to 0$. Hence, if $QL_i'(s,t_0)$ is cotangent to $QR_i'(s,t_0)$, then $R'(s,t_0)$ is cotangent to $L'(s,t_0)$ when $s = 0$. G^1 continuity is assured in the quadratic case for the curve at t_0. Consider now the derivative of Formula 21:

$$\begin{aligned} &L''(s,t_0) = \\ &\quad \sum_i [(2fi(t_0) - 4g_i(t_0) + 2h_i(t_0))3\hat{W}_i(s)^2 \hat{W}_i{}'(s) \\ &\quad + ((1-s)^2 f_i(t) + 2(1-s)sg_i(t) + s^2 h_i(t))(6\hat{W}_i(s)\hat{W}_i{}'(s) + \hat{W}_i{}''(s)) \\ &\quad + (2f_i(t_0) - 4g_i(t_0) + 2h_i(t_0))\hat{W}_i(s)^3 \\ &\quad + (-2(1-s)f_i(t_0) + (2-4s)g_i(t_0) + 2sh_i(t_0))3\hat{W}_i(s)^2 \hat{W}_i{}'(s)] \end{aligned} \tag{24}$$

In this case, for $s = 0$

$$\begin{aligned} L''(0,t_0) = \sum_i [(2fi(t_0) - 4g_i(t_0) + 2h_i(t_0))3\hat{W}_i(0)^2 \hat{W}_i{}'(0) \\ + (2f_i(t_0) - 4g_i(t_0) + 2h_i(t_0))\hat{W}_i(0)^3] \end{aligned} \tag{25}$$

From Formula 13

$$L''(0,t_0) = \sum_i [(2fi(t_0) - 4g_i(t_0) + 2h_i(t_0))\hat{W}_i(0)^3] \tag{26}$$

Once again we see the familiar blended form of Formula 7, this time blending second order differences, which can easily be input with the right user interface. A similar form exists for $R''(0,t_0)$. If $L(s,t_0)$ and $R(s,t_0)$ are designed to have equal derivatives up to second order, then Formulae 19 and 20 will meet with curvature continuity, i.e. together they will form a G^2 curve at t_0. This allows application of a theorem of [2]

Theorem 2. *Let* $\mathbf{F}$ *and* $\mathbf{G}$ *be* G^n *surfaces sharing a common* G^1 *curve denoted by* $\mathbf{R}(t)$. *Suppose that there exists a family of* G^n *curves* $\mathbf{E}_t(s) = \mathbf{E}(t,s)$ *so that each* $\mathbf{E}_t$ *is a surface curve of* $\mathbf{F}$ *for* $s \leq 0$, *each* $\mathbf{E}_t$ *is a surface curve of* $\mathbf{G}$ *for* $s \geq 0$, *and* $\mathbf{E}_t(0) = \mathbf{R}(t)$ *and* $\mathbf{E}'_t(0)$ *is not parallel to* $\mathbf{R}'(t)$. *Then* $\mathbf{F}$ *and* $\mathbf{G}$ *have a* G^n *continuous join.*

Theorem 2 together with the continuity of the $L''(s,t_0)$ and $R''(s,t_0)$ guarantees that the surfaces L and R meet with G^2 across the common boundary curve. We have

Theorem 3. *Given*

$$L(\mathbf{u}) = \sum_i QL_i(s,t)\hat{W}_i(\mathbf{u})^3, \qquad R(\mathbf{u}) = \sum_i QR_i(s,t)\hat{W}_i(\mathbf{u})^3$$

as in Formulae 19 and 20, in which $QL_i(s,t)$ *and* $QR_i(s,t)$ *meet with* G^2 *continuity on the boundary curves of* $L(\mathbf{u})$ *and* $R(\mathbf{u})$, *then* $L(\mathbf{u})$ *and* $R(\mathbf{u})$ *are* G^2 *continuous.*

Finally we consider the mixed partials of Formula 7. First,

$$\begin{aligned}\partial L(s,t)/\partial s = &\sum_i [QL_i(s,t)3\hat{W}_i(s)^2\partial\hat{W}_i(s)/\partial s + (-2(1-s)f_i(t_0) \\ &+ (2-4s)g_i(t_0) + 2sh_i(t_0))\hat{W}_i(s)^3]\end{aligned} \tag{27}$$

As before at $s = 0$ it leads to

$$\partial L(0,t)/\partial s = (-2f_i(t_0) + 2g_i(t_0))\hat{W}_i(s)^3 \tag{28}$$

then

$$\partial^2 L(0,t)/\partial s\partial t = (-2f'_i(t_0) + 2g'_i(t_0))\hat{W}_i(s)^3 \tag{29}$$

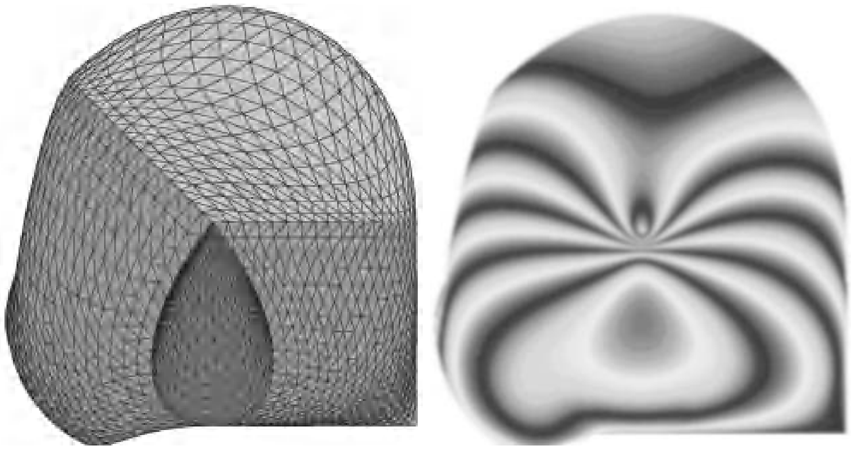
The "twist" vector of the left surface is the blend of the twist vector of the loft. Matching twists between left and right lofts results in matching twists on the boundary of the surfaces. We have

Theorem 4. *Given*

$$L(\mathbf{u}) = \sum_i QL_i(s,t)\hat{W}_i(\mathbf{u})^3, \qquad R(\mathbf{u}) = \sum_i QR_i(s,t)\hat{W}_i(\mathbf{u})^3$$

as in Formulae 19 and 20, in which $QL_i(s,t)$ *and* $QR_i(s,t)$ *meet with twist continuity on the boundary curves of* $L(\mathbf{u})$ *and* $R(\mathbf{u})$, *then* $L(\mathbf{u})$ *and* $R(\mathbf{u})$ *are twist continuous.*

Fig 5(a) shows a configuration of 2 through 5-sided patches. Fig 5(b) shows the isophote rendering of the same configuration. Isophote rendering is sensitive to G^2 and twist continuity and would kink at the boundaries where the surfaces failed to be G^2 or twist continuous.



(a) Flat rendering (b) Isophote rendering

Fig. 5. Configuration of G_2 continuous patches

5 Footprints and Distances

As mentioned in the previous proofs, the distance is assumed to an affine map of the underlying parameterization. The footprints are configured into a polygon with the same number of sides as the patch. An interesting problem is to choose the lengths and angles so as to achieve a 'fair' parameterization, but it is outside the scope of this paper. We have employed regular polygons for the footprint configuration for all figures in this paper, wnd they work fairly well.

The question is how to determine distance that is an affine map so the previous theorems hold. Note that Euclidean, i.e. perpendicular, distance from a point

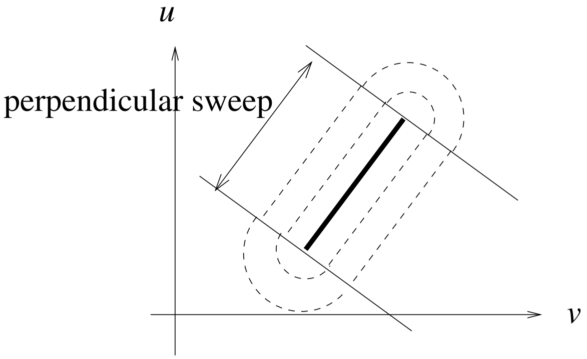
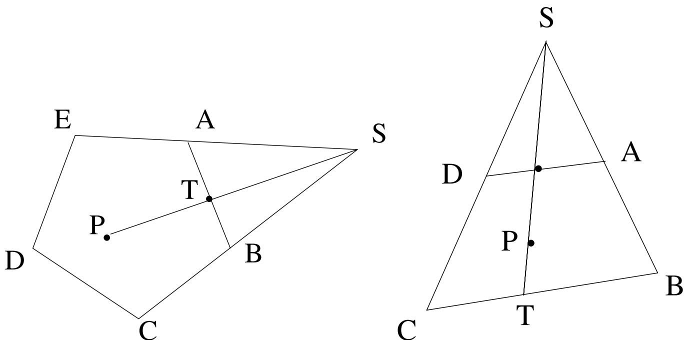


Fig. 6. Euclidean distance contours (dotted) are affine maps of u and v only in the perpendicular sweep



(a) Pentagonal footprints

(b) Quadrilateral footprint with a shared *stella* point

Fig. 7. Computing *stellated distance*

to the line segment is not affine map of the u and v parameters, except within the perpendicular sweep of the footprint, see Fig 6. Once past the endpoints of the footprint it fails to be affine; it is circular. Therefore it will only work for triangular and rectangular footprint polygons that contain points within the perpendicular sweeps of their footprints.

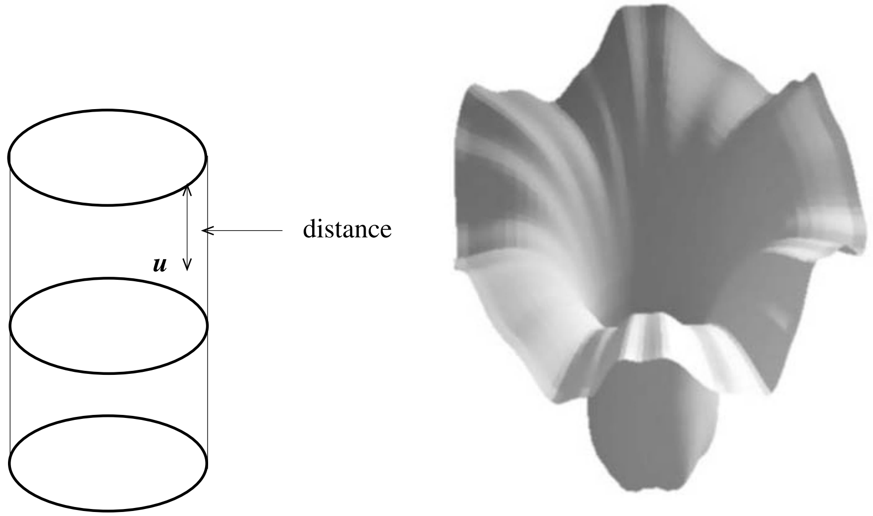
Stellated distance is a method to compute distance for all convex polygonal footprints that satisfies the affine mapping requirement. It is described as follows:

For each edge, compute its *stella* point. For example, in Fig 7(a), we have a pentagonal footprint ABCDE. For edge AB, its stella point is the intersection of the two line segments EA and BC, which are the two neighboring edges of AB. Take a point inside the polygon, P, PS forms a line segment which intersects AB at point T. In some cases, such as in Fig 7(b), two footprints can share the same stella point, and footprint BC intersects the extension of PS at T. The Euclidean distance between P and T is the stella distance for a point P to a line footprint AB.

Since PS and AB intersect within AB, the parameter t_i in T=(1-t_i)A+t_iB ranges from 0 to 1. It also gives a parameterization to the parametric attribute curve back in the attribute space.

In the case of a triangle footprint, each edge's stella point is simply its opposite vertex. Another special case is when the two neighboring edges of a footprint are parallel. In this case, we simply use the perpendicular distance from point P to the footprint AB. It is observed that *stellated distance* method achieves the affine mapping quality by restricting the Euclidean distance to a footprint within its perpendicular sweep.

While we are discussing footprints it needs to be pointed out that footprint spaces can be generalized; they do not have to be restricted to polygons in the plane. The polygons are useful because of our focus on multi-sided patches. Con-



(a) Footprint space and three attribute curves (bold)

(b) Morning glory

Fig. 8. Cylindrical footprint space

sider, however, Fig 8, which has a cylindrical footprint space where the footprints are circles on the cylinder, and distance from a point to the footprint is measured along the cylinders axis. Three footprints are sufficient to define the morning glory. Attribute curves are trigonometric functions of the circular footprints.

6 Discussion

Attribute Based Modeling enables a designer to define a model with an economy of inputs, which implies a notably concise database as shown in Table 1.

Formulae 7, 9, 19 and 20 as linear combinations suggests that for each patch, the computation is linear to the number of its attributes. This is optimal. As the distance goes to zero there exists a singularity in $W_i(\mathbf{u})$ (reciprocal of distance) in the forms, which may be resolved in a couple of ways. First, the distance is given a lower bound within a tolerable error margin, e.g. pixel resolution. All artifacts in this paper were handled this way. Second, in cases that require higher

Table 1. Inputs for various models

Figure	Patches	Curves	Max # Curves/Patch	Lofts
4	7	21	4	21
7	4	13	5	13
10	2	5	3	2

precision, e.g. CAD/CAM, the following may be used [9], which is derived from Formula 7.

$$F(\mathbf{u}) = \sum_i \prod_{j \neq i} d_j(\mathbf{u}) f_i(\mathbf{u_i}) / \sum_i \prod_{j \neq i} d_j(\mathbf{u}) \tag{30}$$

where $d_j(\mathbf{u}) = 1/W_j(\mathbf{u})$. Formula 30 guarantees an accurate computation of the surface point, but increases the computation to a quadratic time complexity.

For polynomial or rational attribute functions the surfaces defined by Formulae 7, 9, 19 and 20 are rational polynomials, although we note that input functions do not need to be polynomial or rational, only parametric. The artifacts in this paper were made with cubic Bezier and B-spline attribute functions, except Fig 8. With G^0 continuity, the attribute function is linear in s and a third degree polynomial in t. The weights with affine distances, as seen in Formula 30, contribute degree $n-1$ in the numerator and denominator in s, where n is the number of footprints. This is squared for G^1. Hence we obtain degree $1+2(n-1) = 2n-1$ over degree $2n-2$ in s and degree 3 in t. The degrees in s of the two, three, four and five-sided patches with G^1 continuity are, respectively, degrees 3/2, 5/4, 7/6 and 9/8, from which the linearity of computation with respect to number of footprints is also clear. The degrees in t are all 3.

7 Future

The methodology of this paper spawns a number of interesting research topics. For instance, the method does not require that footprints and curves be joined; it is more general than described in this paper. It allows for dangling edges and points to be used as attributes. This enables more refinement in the interior of the patches.

The method can also be extended to volume modeling by generalizing footprint space to $\mathbb{R}^3$ and allowing surfaces as footprints and attributes.

With the pre-defined boundary information, e.g. curves and slopes, a collection of patches can be dropped seamlessly onto another surface, enabling design from pre-existing parts in databases. For example, we imagine a library of noses that can be dragged onto a gap in a face and dropped in like “Mr. Potato Head.”

Operational techniques must be developed such as texture mapping, surface-ray and surface-surface intersection, and patch subdivision. The convex combination as in Formula 7 and the other forms immediately leads to convex hull and affine invariance properties. We would like to establish a variation diminishing property, which should follow from the least squares approach.

References

1. Ghostasby, A.: Image Registration by Local Approximation Methods. Image and Vision Computing, **6** (4) Nov. (1988) 255–261.
2. Hermann, T. et al.: Geometrical Criteria on the Higher Order Smoothness of Composite Surfaces. Computer Aided Geometric Design, **16** (1999) 907–911.

3. Hoschek, J., Lasser, D.: Fundamentals of Computer Aided Geometric Design, A. K. Peters (1993).
4. Levin, A.: Interpolating Nets of Curves by Smooth Subdivision Surfaces. ACM SIGGRAPH (1999) 57–64.
5. Malraison, P.: Multi-sided Surfaces: a survey. Curve and Surface Design. Editors Pierre-Jean Laurent, Paul Sablonière, Larry Schumaker, (1999) 246–256.
6. Schneider, J.: Solving the Nearest Point on the Curve Problem. Graphics Gems (1990) 607–612.
7. Sederberg, T. et al.: T-splines and T-NURCCs. ACM SIGGRAPH (2003) 477–484.
8. Sederberg, T. et al.: T-spline Simplification and Local Refinement. ACM SIGGRAPH (2004) 276–283.
9. Singh, K., Fiume, E.: Wires: a Geometric Deformation Technique. ACM SIGGRAPH (1998) 405–414.

On Normals and Control Nets

Ingo Ginkel[1], Jorg Peters[2], and Georg Umlauf[1]

[1] University of Kaiserslautern, Germany
{ginkel, umlauf}@informatik.uni-kl.de
[2] University of Florida, Gainesville, FL, USA
jorg@cise.ufl.edu

Abstract. This paper characterizes when the normals of a spline curve or spline surface lie in the more easily computed cone of the normals of the segments of the spline control net.

1 Introduction

Since, for splines, subdivision amounts to averaging, would not the limit normal of a spline curve or surface be a linear, even convex combination of the normals of the segments of the spline control net? In this introduction and in Section 2 we present some evidence that supports this conjecture for curves and functions. Section 3 discusses the case of box-spline surfaces.

Such an investigation is practically useful, since an affirmative answer to the question would allow substituting a simpler computation of bounds on the control structure for complex, exact bounds on the corresponding nonlinear curve or surface. Such bounds play a significant role in curve and surface intersection algorithms and in efficient or high-quality rendering.

For a planar, polynomial curve segment in B-spline or Bézier form, the cone spanned by the perpendiculars to the control segments encloses the cone of the curve normals. This follows from

(i) the coefficients of the hodograph (the derivative of the parametrization) are the scaled differences of the coefficients of the curve parametrization,
(ii) the hodograph is a convex combination of its coefficients and
(iii) the normal to the curve segment is perpendicular to its hodograph.

In symbols, we consider a planar curve

$$\mathbf{c}(t) := \sum_i \mathbf{c}_i \, b_i(t), \quad \mathbf{c}_i \in \mathbb{R}^2,$$

in terms of Bernstein polynomials or B-splines $b_i(t)$ of degree d. Then $\perp(x,y) = (-y,x)$ is the normal direction $\mathbf{n}(\mathbf{c},t)$ of $\mathbf{c}$ at t, and we may define

$$\Delta_i(\mathbf{c}) := \begin{cases} d(\mathbf{c}_{i+1} - \mathbf{c}_i), & \text{if } i \in \mathsf{I} := \{0,\ldots,n-1\} \\ 0, & \text{else.} \end{cases}$$

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 233–239, 2005.

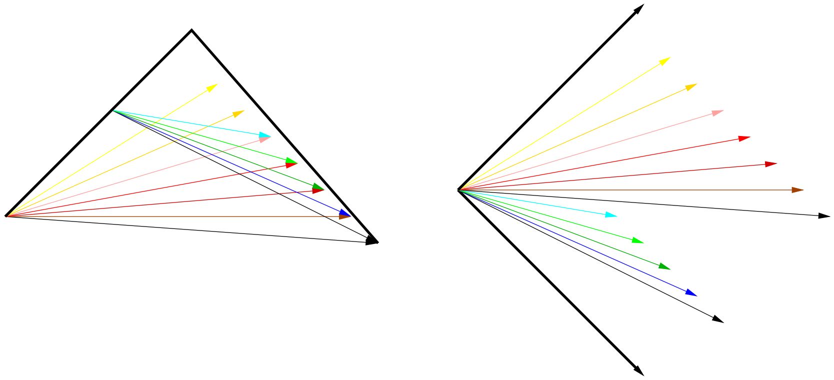


Fig. 1. The differences of intermediate points generated by de Boor's algorithm lie in the cone spanned be the differences $\Delta_i(\mathbf{r}), i \in \mathsf{I}$

The proof of the assertion is

$$\mathbf{n}(\mathbf{c},t) = \perp(\mathbf{c}'(t)) = \perp\left(\sum_i \Delta_i(\mathbf{c})\, b_i(t)\right) = \sum_i \perp(\Delta_i(\mathbf{c}))\, b_i(t)$$
$$\in \operatorname{cone}(\perp(\Delta_i(\mathbf{c})))_{i\in\mathsf{I}}.$$

Note that the basic idea generalizes to curves in higher-dimensional spaces and characterizes the normal hyperplanes perpendicular to the tangent cone spanned by the differences of the control polygon.

2 Curves and Bivariate Functions

We generalize the earlier argument to rational planar curves of the form

$$\mathbf{r}(t) := \frac{\sum_i w_i\, \mathbf{r}_i\, b_i(t)}{\sum_i w_i\, b_i(t)}, \quad \mathbf{r}_i \in \mathbb{R}^2,$$

with $w_i > 0$: the cone spanned by the perpendiculars to the control segments encloses the cone of the curve normals.

Lemma 1. *Let* $\mathbf{n}(\mathbf{r},t)$ *be the normal to the rational Bézier curve segment* $\mathbf{r}$ *and* $\rho := \sum_i \mathbf{r}_i\, b_i$. *Then*

$$\mathbf{n}(\mathbf{r},t) \in \operatorname{cone}(\perp(\Delta_i(\rho)))_{i\in\mathsf{I}}.$$

Proof. Both the rational de Casteljau and the de Boor algorithm use only pairwise convex combinations to compute intermediate points $\rho_i^j, j = 0,\ldots,d, i = 0,\ldots,j$, when written in non-homogeneous space (see e.g. [1]). For non-negative weights, the differences of intermediate points lie therefore in the cone spanned by the differences $\Delta_i(\rho), i \in \mathsf{I}$, (Figure 1) and the tangent of $\mathbf{r}$ at parameter t is

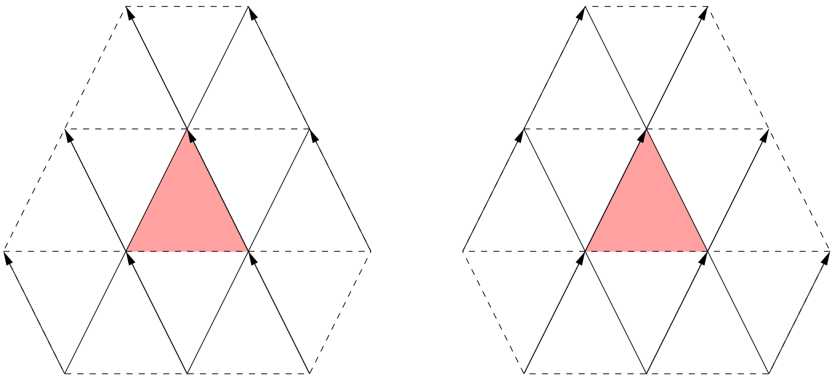


Fig. 2. Control points p_{i}, $\mathsf{i} \in \mathsf{I}$, (the set of all 12 line intersections) defining a degree 4 three-direction box-spline parametrized over the central, shaded triangle. The arrows indicate two (affinely skewed, box-spline averaging) directions e_1 and e_2 contributing to the partial derivatives of the surface piece. Note that, in each case, four of the twelve differences are zero since they do not contribute to the derivative

the scaled difference $\rho_1^{d-1} - \rho_0^{d-1}$ of the affine intermediate control points [2, 3]. Finally, the normal direction $\mathbf{n}(\mathbf{r}, t)$ of $\mathbf{r}$ at t is perpendicular to the tangent of $\mathbf{r}$ at t. □

Similarly, for a planar C^1 subdivision curve generated by a binary subdivision scheme with weights α_i

$$\mathbf{s}'_{2i} = \sum_j \alpha_{2i-2j}\mathbf{s}_j,$$
$$\mathbf{s}'_{2i+1} = \sum_j \alpha_{2i+1-2j}\mathbf{s}_j,$$

whose difference scheme uses only convex combinations ([4]), the cone spanned by the perpendiculars to the control segments encloses the cone of the curve normals

$$\mathbf{n}(\mathbf{s}, t) \in \text{cone}(\perp(\Delta_i(\mathbf{s})))_{i\in\mathsf{I}}.$$

This follows from

(i) the divided differences of control points of a refined control polygon lie in the cone of divided differences of control points of the coarse control polygon,
(ii) the divided differences converge towards the scaled tangents,
(iii) the normal direction $\mathbf{n}(\mathbf{s}, t)$ of $\mathbf{s}$ at t is perpendicular to the tangent at t.

Let us now consider a (uniform box-)spline in two variables $(\mathsf{u}_1, \mathsf{u}_2) =: \mathsf{u}$,

$$p(\mathsf{u}) := \sum_{\mathsf{i}\in\mathsf{I}} p_{\mathsf{i}} b_{\mathsf{i}}(\mathsf{u})$$

with coefficients $p_{\mathsf{i}} \in \mathbb{R}$ and $\mathsf{i} = (\mathsf{i}_1, \mathsf{i}_2) \in \mathsf{I} \subset \mathbb{Z}^2$, a multi-index, and I an appropriate index set (see [5]). One polynomial piece of such a spline is a linear combination of nodal or box-spline functions b_{i}, $\mathsf{i} \in \mathsf{I}$, of total degree d. In particular, we consider the three-direction box spline with directions $\varXi := \left[\begin{smallmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{smallmatrix}\right]$. A polynomial piece p is defined by a submesh consisting of all triangles (point- or edge-) adjacent to a central triangle (see Figure 2). Two derivatives in the generating directions, without loss of generality, $\mathsf{e}_1 := (1, 0)$ and $\mathsf{e}_2 := (0, 1)$, can be obtained by differencing control points in these directions. We define

$$\varDelta_{k,\mathsf{i}}(p) := \begin{cases} d(p_{\mathsf{i}+\mathsf{e}_k} - p_{\mathsf{i}}), & \text{if both } \mathsf{i} \in \mathsf{I} \text{ and } \mathsf{i} + \mathsf{e}_k \in \mathsf{I}, \\ 0 \text{ else} \end{cases}$$

for $k = 1, 2$. Then the same relation holds as for planar curves.

Lemma 2. *The normal of a polynomial piece* p *of a bivariate box-spline function with averaging matrix* $\varXi$ *is in the normal cone spanned by the control facet normals with vertices* p_{i}, $\mathsf{i} \in \mathsf{I}$.

Proof. By definition, the normal direction is

$$\begin{aligned}
\mathbf{n}\left(\begin{pmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \\ p \end{pmatrix}, \mathsf{u}\right) &:= \frac{\partial \begin{pmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \\ p \end{pmatrix}}{\partial \mathsf{u}_1} \times \frac{\partial \begin{pmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \\ p \end{pmatrix}}{\partial \mathsf{u}_2} \\
&= \sum_{\mathsf{i}} \begin{pmatrix} 1 \\ 0 \\ \varDelta_{1,\mathsf{i}}(p) \end{pmatrix} b_{\mathsf{i}}(\mathsf{u}) \times \sum_{\mathsf{i}} \begin{pmatrix} 0 \\ 1 \\ \varDelta_{2,\mathsf{i}}(p) \end{pmatrix} b_{\mathsf{i}}(\mathsf{u}) \\
&= \begin{pmatrix} 1 \\ 0 \\ \sum_{\mathsf{i}} \varDelta_{1,\mathsf{i}}(p) b_{\mathsf{i}}(\mathsf{u}) \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ \sum_{\mathsf{i}} \varDelta_{2,\mathsf{i}}(p) b_{\mathsf{i}}(\mathsf{u}) \end{pmatrix} \\
&= \sum_{\mathsf{i}} \begin{pmatrix} -\varDelta_{1,\mathsf{i}}(p) \\ -\varDelta_{2,\mathsf{i}}(p) \\ 1 \end{pmatrix} b_{\mathsf{i}}(\mathsf{u}) \\
&\in \operatorname{cone}\left(\begin{pmatrix} -\varDelta_{1,\mathsf{i}}(p) \\ -\varDelta_{2,\mathsf{i}}(p) \\ 1 \end{pmatrix}\right)_{\mathsf{i} \in \mathsf{I}} \\
&= \operatorname{cone}\left(\varDelta_{1,\mathsf{i}} \begin{pmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \\ p \end{pmatrix} \times \varDelta_{2,\mathsf{i}} \begin{pmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \\ p \end{pmatrix}\right)_{\mathsf{i} \in \mathsf{I}}.
\end{aligned}$$

The last expressions are the normals of the control facets. □

Here, we picked the three-direction box-spline since the control facets have a unique normal, as opposed to the control net of a tensor-product spline.

3 Box-Spline Surfaces

The examples of curves in one and functions in two variables suggest that *the cone of normal directions* $\mathbf{n}(\mathbf{p}, \mathsf{u})$, $\mathsf{u} \in U$, *of a spline surface patch* $\mathbf{p}$ *is in the cone of the normals of its spline control polyhedron of* $\mathbf{p}$. To test this hypothesis, we consider again the three-direction box splines with direction matrix $\varXi := \left[\begin{smallmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{smallmatrix}\right]$.

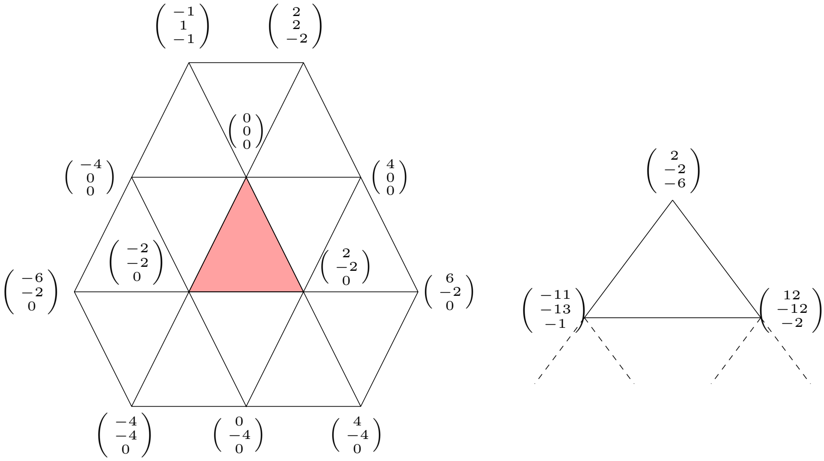

$\begin{pmatrix}-1\\1\\-1\end{pmatrix}$
$\begin{pmatrix}2\\2\\-2\end{pmatrix}$
$\begin{pmatrix}0\\0\\0\end{pmatrix}$
$\begin{pmatrix}-4\\0\\0\end{pmatrix}$
$\begin{pmatrix}4\\0\\0\end{pmatrix}$
$\begin{pmatrix}-2\\-2\\0\end{pmatrix}$
$\begin{pmatrix}2\\-2\\0\end{pmatrix}$
$\begin{pmatrix}-6\\-2\\0\end{pmatrix}$
$\begin{pmatrix}6\\-2\\0\end{pmatrix}$
$\begin{pmatrix}-4\\-4\\0\end{pmatrix}$
$\begin{pmatrix}0\\-4\\0\end{pmatrix}$
$\begin{pmatrix}4\\-4\\0\end{pmatrix}$
$\begin{pmatrix}2\\-2\\-6\end{pmatrix}$
$\begin{pmatrix}-11\\-13\\-1\end{pmatrix}$
$\begin{pmatrix}12\\-12\\-2\end{pmatrix}$

Fig. 3. (*left*) A three-direction box-spline control net. Only the top two coefficients have a non-zero z-component. The (x, y) coordinates of all coefficients, except the top left, are evenly distributed. For simplicity, the top left coefficient is the average of the top right and $(-4, 0, 0)^T$. (*right*) Bézier control points (scaled by 24) corresponding to the tip of the triangular patch near $(0, 0, 0)$

Lemma 3. *The normal of a polynomial piece* $\mathbf{p}$ *of a bivariate parametric box-spline surface with averaging matrix* Ξ *is in the normal cone spanned by the cross products of each difference in the direction* e_1 *with each difference in the direction* e_2 *of control points* $\mathbf{p}_\mathsf{i} \in \mathbb{R}^3$, $\mathsf{i} \in \mathsf{I}$.

Proof. Since the cross product results in a polynomial of degree $2d - 2$ and the box-spline functions b_i form a non-negative partition of 1, we can bound the normal by

$$\begin{aligned}\mathbf{n}(\mathbf{p}, \mathsf{u}) := \frac{\partial \mathbf{p}}{\partial \mathsf{u}_1} \times \frac{\partial \mathbf{p}}{\partial \mathsf{u}_2} &= \sum_{\mathsf{i}\in\mathsf{I}} \mathit{\Delta}_{1,\mathsf{i}}(\mathbf{p}) b_\mathsf{i}(\mathsf{u}) \times \sum_{\mathsf{j}\in\mathsf{I}} \mathit{\Delta}_{2,\mathsf{j}}(\mathbf{p}) b_\mathsf{j}(\mathsf{u}) \\ &\in \operatorname{cone}(\mathit{\Delta}_{1,\mathsf{i}}(\mathbf{p}) \times \mathit{\Delta}_{2,\mathsf{j}}(\mathbf{p}))_{\mathsf{i}\in\mathsf{I},\mathsf{j}\in\mathsf{I}}.\end{aligned}$$

□

Lemma 3 indicates that a much larger set, the set of cross products of *all* edge vectors of the control net in the direction e_1 with *all* edge vectors in the direction e_2, generates a cone that includes the limit normals of the patch. This raises the question, whether such a larger set is also necessary.

The choice of coefficients in Figure 3, *left*, shows that it is not enough to just consider the normals of the control facets.

Lemma 4. *The normal of a polynomial piece* $\mathbf{p}$ *of a box-spline surface with averaging matrix* Ξ *is in general not in the normal cone spanned by the control facet normals with vertices* $\mathbf{p}_\mathsf{i}$, $\mathsf{i} \in \mathsf{I}$.

Proof. To certify the control net as a counterexample, we use the fact that the three-direction patch **p** can alternatively be expressed in terms of Bézier coefficients (see e.g. [6, 7] for explicit formulas). Figure 3, *right*, shows the Bézier coefficients near the top point $(0, 0, 0)$ corresponding to $\mathbf{0}$. From these it is easy to compute the normal. (Alternatively, we could have used the explicit formula for normals of Loop subdivision surfaces.) Explicitly, with "$*$" indicating unimportant values,

$$\mathbf{n}(\mathbf{p}, \mathbf{0}) = \begin{pmatrix} -13 \\ -11 \\ 5 \end{pmatrix} \times \begin{pmatrix} 10 \\ -10 \\ 4 \end{pmatrix} = \begin{pmatrix} 6 \\ * \\ * \end{pmatrix},$$

i.e. the normal at the tip of the triangular patch has a nonzero x-component, while the facet normals lie in the $x = 0$ plane. □

Note that the counterexample crucially depends on non-functional data. What is the flaw in the argument that, since subdivision amounts to averaging, the limit normals are a linear combination of the initial normals? One flaw is that the normal is bilinear with respect to the action of the subdivision matrices. Hence, we should not expect to find a linear averaging scheme for the normal. Indeed the following lemma shows that, in general, no subdivision scheme exists for the normals.

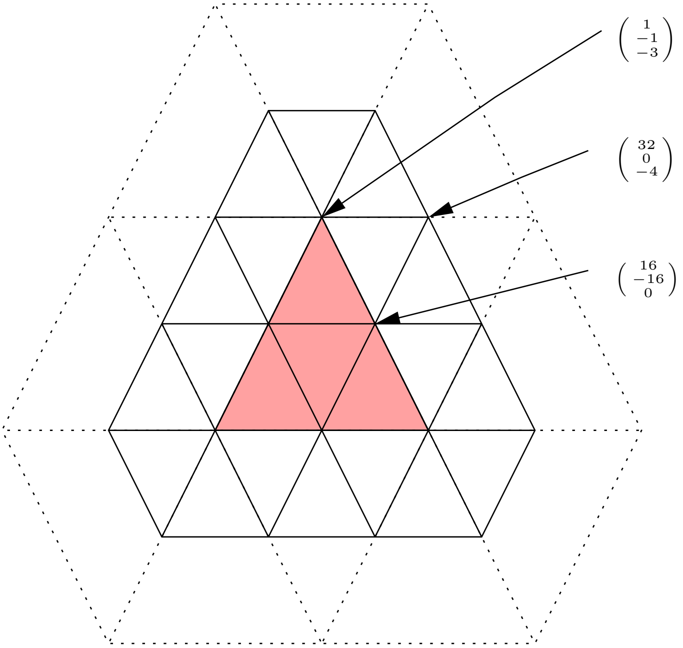

$\begin{pmatrix} 1 \\ -1 \\ -3 \end{pmatrix}$
$\begin{pmatrix} 32 \\ 0 \\ -4 \end{pmatrix}$
$\begin{pmatrix} 16 \\ -16 \\ 0 \end{pmatrix}$

Fig. 4. Coefficients (scaled by 16) of the once-refined control net

Lemma 5. *No linear subdivision scheme exists for the normals of the three-direction box-spline with direction matrix Ξ.*

Proof. We perform one subdivision step using the box-spline subdivision rules (see e.g. [8]) on the data of Figure 3, *left*, resulting in the control net shown in Figure 4. Again, we consider the facet normals at the apex of the central triangle. It has normal direction

$$\left(\begin{pmatrix}32\\0\\-4\end{pmatrix}-\begin{pmatrix}1\\-1\\-3\end{pmatrix}\right)\times\left(\begin{pmatrix}16\\-16\\0\end{pmatrix}-\begin{pmatrix}1\\-1\\-3\end{pmatrix}\right)=\begin{pmatrix}31\\1\\-1\end{pmatrix}\times\begin{pmatrix}15\\-15\\3\end{pmatrix}=\begin{pmatrix}-12\\ *\\ *\end{pmatrix}.$$

Since the normal direction has a nonzero x-component, it is not in the space spanned by the facet normals of the original control mesh. □

4 Generalizations

At least one further investigation deserves attention. Since the cross products, of all edges of the control net of one directional derivative with all edges of another, forms a cone that contains the normals of the regular part of a Loop subdivision surface [8], can a similar statement be made for the neighborhood of extraordinary points of a subdivision surface?

Acknowledgement. This research was support in part by NSF under grants DMI-0400214 and CCF-0430891.

References

1. Boehm, W.: An affine representation of de Casteljau's and de Boor's rational algorithms. Computer Aided Geometric Design **10** (1993) 175–180
2. Floater, M.: Derivatives of rational Bézier curves. Computer Aided Geometric Design **9** (1992) 161–174
3. Floater, M.: Evaluation and properties of the derivative of a NURBS curve. In Lyche, T., Schumaker, L., eds.: Mathematical Methods in CAGD, Boston, Academic Press (1992) 261–274
4. Dyn, N.: Subdivision schemes in CAGD. In Light, W., ed.: Advances in Numerical Analysis. Volume II Wavelets, Subdivision Algorithms, and Radial Basis Functions., Oxford Science Publications (1992) 37–104
5. de Boor, C., Höllig, K., Riemenschneider, S.: Box splines. Springer, New York (1993)
6. Boehm, W.: Generating the Bézier points of triangular splines. In Barnhill, R., Boehm, W., eds.: Surfaces in Computer Aided Geometric Design, North-Holland Publishing Company, (1983) 77–91
7. Peters, J., Shiue, L.: Combining 4- and 3-direction subdivision. ACM Transactions on Graphics **23** (2004) 980–1003
8. Loop, C.T.: Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah (1987)

Line Subdivision

Hiroshi Kawaharada and Kokichi Sugihara

Department of Mathematical Informatics,
Graduate School of Information Science and Technology,
University of Tokyo, Japan
kawarada@simplex.t.u-tokyo.ac.jp, sugihara@mist.i.u-tokyo.ac.jp

Abstract. This paper proposes a new subdivision scheme based on line geometry. We name the scheme 'line subdivision'. Line subdivision scheme acts on the line space, and generates two-dimensional manifolds contained in the Klein quadric. The two-dimensional manifolds are the Klein images of line congruences in P^3. So, this new subdivision scheme generates line congruences at the limit. Here, we define the line subdivision surface as an envelope surface which is made by the line congruence. Then, this paper derives basic properties of the surface. Moreover, we show that line subdivision contains ordinary subdivision and dual subdivision.

1 Introduction

Subdivision [1, 2, 3, 4, 5] is a well-known method for geometric design and for computer graphics, because the subdivision makes smooth surfaces with arbitrary topology. A subdivision scheme is defined by subdivision matrices and a rule of change of connectivity. So, many researchers study the condition of continuity of subdivision surfaces depending on subdivision matrices [4, 6, 7, 8, 9, 10, 11, 12, 13]. Moreover, multiresolution analysis [14, 15, 16, 17, 18, 19] derived by subdivision theory is extremely useful on mesh editing.

For example, the most popular subdivisions are the Catmull-Clark subdivision and the Loop subdivision [2]. These subdivisions are designed for irregular quadrilateral or triangular meshes. Most of other subdivision methods are also for triangular or quadrilateral meshes; there are only a few methods for other types of meshes. Examples are subdivisions on hexangular meshes developed by Claes [20] and Farin [21]. However, faces generated by these subdivisions are not 'flat'.

In [22], we derived a new subdivision scheme. It is the dual framework of an ordinary subdivision based on the principle of duality in projective geometry. The proposed dual subdivision can generate meshes, composed of non-triangular 'flat' faces.

In an ordinary subdivision, we compute new positions of the vertices using old positions of vertices and matrices called subdivision matrices. On the other hand, in the dual subdivision, we compute new equations of faces using old equations of faces and subdivision matrices. Moreover, a mesh which has the subdivision

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 240–254, 2005.

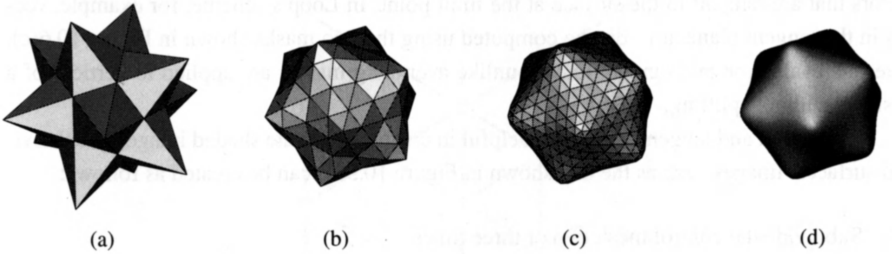


Fig. 1. Loop subdivision [16]

connectivity is the dual of a mesh which has the dual subdivision connectivity. In short, a mesh made by the dual subdivision is the dual mesh of a mesh made by ordinary subdivision. In this sense, the dual subdivision method can be made by an ordinary subdivision method. Conversely, an ordinary subdivision method can be made by the dual subdivision method.

The ordinary subdivision generates vertices, whereas the dual subdivision generates the equations of faces. Thus, we got subdivisions based on vertices and faces. Then, our next question is: is it possible to construct a subdivision scheme based on lines?

In this paper, we answer this question affirmatively. That is, we introduce a new subdivision scheme called 'line subdivision'. We define the line subdivision using line geometry, and we derive properties of line subdivision surfaces. In particular, we show that the line subdivision includes the ordinary and dual subdivisions.

2 Ordinary Subdivision

In this section, we review ordinary subdivisions in general.

2.1 Subdivision Matrix

A subdivision scheme is defined by subdivision matrices and a rule of connectivity change. The subdivision scheme, when it is applied to 2-manifold irregular meshes, generates smooth surfaces at the limit. Fig. 1 is an example of the Loop subdivision. In this figure, (a) is an original mesh; subdividing (a), we get (b); subdividing (b) once more, we get (c); subdividing infinite times, we get the smooth surface (d). We call (d) the subdivision surface. Here, a face is divided into four new faces. This is a change of connectivity. In this paper, the change of connectivity is fixed to this type, but other types of connectivity change can be discussed similarly.

Next, let us consider how to change the positions of the old vertices, and how to decide the positions of the new vertices. They are specified by matrices called 'subdivision matrices'. The subdivision matrices are defined at vertices and they depend on the degree, say k, of the vertex (the degree is the number of edges connected to the vertex). For example, Fig. 2 denotes a vertex v_0^j which has five

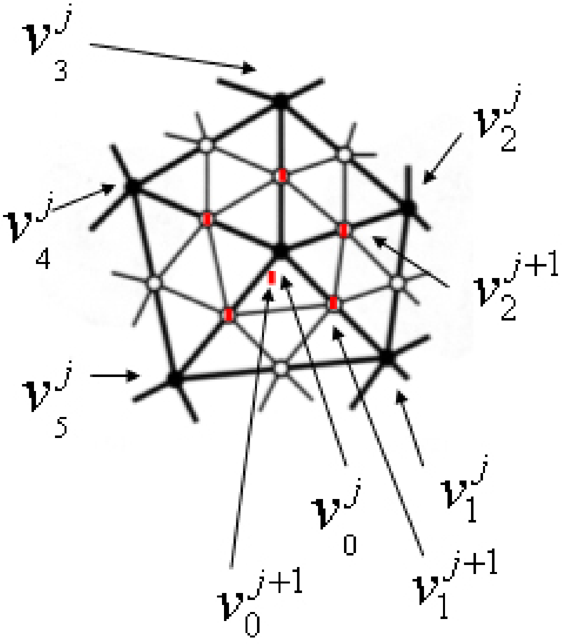


Fig. 2. Subdivision matrix

edges. Let $v_1^j, v_2^j, \cdots, v_5^j$ be the vertices at the other terminal of the five edges. Then, subdivision matrix S_5 is defined as follows:

$$\begin{pmatrix} v_0^{j+1} \\ v_1^{j+1} \\ \vdots \\ v_5^{j+1} \end{pmatrix} = S_5^j \begin{pmatrix} v_0^j \\ v_1^j \\ \vdots \\ v_5^j \end{pmatrix} .$$

Here, the subdivision matrix S_5^j is a square matrix. j means j-th step of the subdivision. The neighbor vertices of a vertex v are called vertices on the 1-disc of v. The subdivision matrix is generally defined not only on vertices in the 1-disc, but also on other vertices around. However, we discussed only subdivision matrices that depend on vertices in the 1-disc. We can discuss other subdivision matrices similarly. In this paper, we assume that the subdivision matrix is independent on j. A subdivision scheme of this type is called 'stationary'.

In this way, subdivision matrix is written for a vertex. However, since a newly generated vertex is computed by two subdivision matrices at the ends of the edge, the two subdivision matrices must generate the same location of the vertex. So, the subdivision matrices have this kind of restriction.

For example, subdivision matrices S_k $(k \geq 3)$ for the Loop subdivision are

$$S_k = \begin{pmatrix} 1-k\beta & \beta & \beta & \beta & \beta & \beta & \cdots & \beta \\ \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & \cdots & 0 & \frac{1}{8} \\ \frac{3}{8} & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & \cdots & 0 \\ \frac{3}{8} & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 & \cdots & 0 \\ \vdots & & & & \ddots & & & \\ \frac{3}{8} & \frac{1}{8} & 0 & 0 & \cdots & 0 & \frac{1}{8} & \frac{3}{8} \end{pmatrix},$$

where k is the degree of the associated vertex, and

$$\beta = \frac{1}{k}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{k}\right)\right)^2\right).$$

The degree k of a vertex is at least two. A vertex whose degree is two is a boundary vertex. The degree of a vertex of 2-manifold meshes is at least three. In this paper, we do not discuss boundaries of meshes. So, we assume that the degree is at least three.

As seen above, a stationary subdivision scheme is defined by subdivision matrices S_k $(k \geq 3)$. Then, by Theorem 2.1 in [9], the limit surface of subdivision $f : |K| \to \mathbf{R}^3$ is the following parametric surface:

$$f[p](y) = \sum_i v_i \phi_i(y),$$
$$v_i \in \mathbf{R}^3,\ \phi_i(y) \in \mathbf{R},\ y \in |K|, p = (v_0, v_1, \cdots),$$

where K is a complex, $|K|$ is a topological space, that is, the mesh, i is an index of a vertex, v_i is the position of the i-th vertex, $\phi_i(y)$ is the weight function with the i-th vertex. Moreover, the weight function $\phi_i(y)$ is dependent only on the subdivision matrices. If the sum of each row of the subdivision matrix is 1, vertices at each stage of subdivision is affine combinations of original vertices. Therefore,

$$\forall y \in |K|,\ \sum_i \phi_i(y) = 1.$$

So, weight functions make affine combinations, too. If the combination is not affine, it is not invariant under the translation of the coordinates systems, and hence we usually consider only affine combinations. Therefore, in what follows we assume that the sum of element in each row of the subdivision matrix is equal to 1.

Here, we denote $\phi(y) = (\phi_0(y), \phi_1(y), \cdots)$. Then, $\phi(y)$ decides a set of representable surfaces. Then, the set is spanned by $\phi(y)$. So, we call the weight functions basis functions. The limit surface of the subdivision is a point in such a functional space.

3 Dual Subdivision

In [22], we proposed a dual subdivision method. In this section we review it very briefly.

3.1 Definition of Dual Subdivision

The ordinary subdivision is specified by how the vertices are generated and located. On the other hand, the dual subdivision, which we will define here, is specified by how the faces are generated and located.

We assumed that the sum of each row of the subdivision matrix is 1.

Here, let p^j be the column vector of vertices at the j-th subdivision step. Using a subdivision matrix S, p^{j+1} is written as:

$$p^{j+1} = Sp^j,$$

where

$$p^j = \begin{pmatrix} p^j_{0x} \; p^j_{0y} \; p^j_{0z} \\ p^j_{1x} \; p^j_{1y} \; p^j_{1z} \\ \vdots \end{pmatrix}.$$

Therefore, if we denote

$$f^j = \begin{pmatrix} p^j_{0x} \; p^j_{0y} \; p^j_{0z} \; -1 \\ p^j_{1x} \; p^j_{1y} \; p^j_{1z} \; -1 \\ \vdots \end{pmatrix},$$

we get

$$f^{j+1} = Sf^j,$$

where the elements of each row of f^j are coefficients of the equation $p^j_{ix}x + p^j_{iy}y + p^j_{iz}z - 1 = 0$. Therefore, the equations of planes are subdivided. These equations are the dual of vertices $(p^j_{ix}, p^j_{iy}, p^j_{iz})$. So, this subdivision is a dual framework of ordinary subdivision. Moreover, dual subdivision can be defined by a projective duality $(p_x, p_y, p_z, p_w) \leftrightarrow p_x x + p_y y + p_z z - p_w w = 0$.

Now, for any triangular mesh M, using the duality, we get a dual mesh. Let us denote this dual mesh by $D(M)$. For a vertex v and a face f, we denote their duals by $D(v)$ and $D(f)$, respectively. Therefore, $D(v)$ is a plane and $D(f)$ is a point in the dual space. Here, if the degree of a vertex v of M is k, then v is the intersection of faces $f_i,\ i = 1, 2, \cdots, k$, so these vertices $D(f_i),\ i = 1, 2, \cdots, k$ is on the face $D(v)$. So, we get following proposition:

Proposition 1. *If the degree of the vertex v of M is k, the face $D(v)$ of $D(M)$ is a k-gon.*

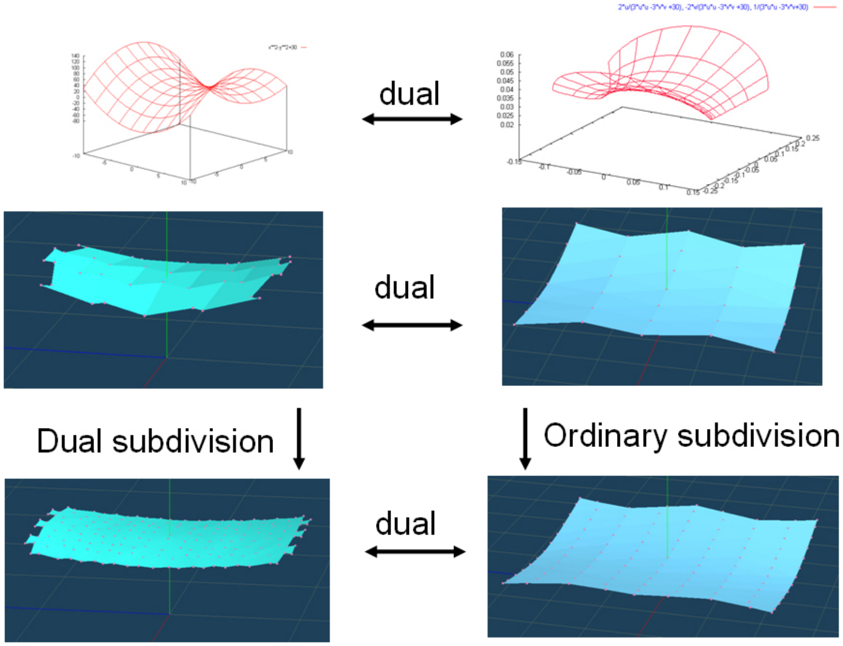


Fig. 3. The duality between ordinary subdivision and dual subdivision. The upper left drawing shows a surface which has saddle points. The upper right drawing is the dual surface. The middle right picture is a triangular mesh made by plotting points on the upper right surface. The middle left mesh is the dual mesh of the middle right mesh. We get the lower right mesh by ordinary subdivision for the middle right mesh. On the other hand, we get the lower left mesh by dual subdivision for the middle left mesh. Then, the lower left mesh is the dual mesh of the lower right mesh. Dual subdivision is defined as such. Like this, dual subdivision can represent surfaces which have saddle points

Here, we define the rule of connectivity change of dual subdivision. The connectivity change of dual subdivision is defined as dual of the connectivity change of ordinary subdivision. Fig. 3 denotes the definition of dual subdivision. In short, the mesh made by a dual subdivision in the dual space is the dual mesh of a mesh made by a corresponding ordinary subdivision in the primal space.

4 Line Geometry

In this section, we introduce some basic properties of line geometry [23, 24, 25, 26].

Let P^n be the n-dimensional real projective space. For a point $p \in P^3$, let the homogeneous coordinates of p be (p_0, p_1, p_2, p_3). Points with $p_0 = 0$ are called ideal points. For two different points $p, q \in P^3$, we can characterize the line passing through p and q using wedge product as

$$L = p \wedge q,$$

where $p \wedge q = (l_{01}, l_{02}, l_{03}, l_{23}, l_{31}, l_{12})$,$l_{ij} = p_i q_j - p_j q_i$. The elements of L are called the Plúcker coordinates, which are homogeneous coordinates of the line. In fact, let $p' = \alpha_0 p + \beta_0 q, q' = \alpha_1 p + \beta_1 q$, where $\alpha_i, \beta_i \in \mathbf{R}$, then

$$L' = p' \wedge q' = \det \begin{pmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{pmatrix} L.$$

Moreover, l_{ij} satisfies an equation, called the Plúcker identity:

$$l_{01}l_{23} + l_{02}l_{31} + l_{03}l_{12} = 0.$$

Here, let $l = (l_{01}, l_{02}, l_{03}), \bar{l} = (l_{23}, l_{31}, l_{12})$. Thus, $L = (l, \bar{l})$. Then, the Plúcker identity is $l \cdot \bar{l} = 0$. Now, let $\hat{L} = (\hat{l}, \bar{\hat{l}})$ be another line. The two lines $L, \hat{L}$ intersect if and only if $\langle L, \hat{L} \rangle = l \cdot \bar{\hat{l}} + \bar{l} \cdot \hat{l} = 0$. This is because $\langle L, \hat{L} \rangle$ is the determinant of matrix $[p, q, \hat{p}, \hat{q}]$, where $\hat{p}$ and $\hat{q}$ are two different points on the line $\hat{L}$. So, the Plúcker identity is a special case of this equation, that is, $\frac{1}{2}\langle L, L \rangle = 0$.

Direction and Moment vector. Now, without loss of generality, we can assume $q = (0, q_1, q_2, q_3), p = (1, p_1, p_2, p_3)$. Then, $L = (q_1, q_2, q_3, (q_1, q_2, q_3) \times (p_1, p_2, p_3))$. Here, $l = (q_1, q_2, q_3)$ is the direction vector of L and $\bar{l} = (q_1, q_2, q_3) \times (p_1, p_2, p_3)$ is called the moment vector of L. The Plúcker identity expresses the orthogonality of l and $\bar{l}$.

Klein mapping. The Plúcker coordinates of the line L in P^3 are the point $(l, \bar{l})$ in P^5. The point is contained in the hyperquadric $M_2^4 \subset P^5$, which is defined by the Plúcker identity. Conversely, a point in M_2^4 corresponds to a unique line in P^3. So, this mapping γ is bijective. γ is called the *Klein mapping*, and the point in M_2^4 is called the *Klein image* of the line in P^3.

Pencil of Lines. Let a be a point and L be a non-incident line in P^3. A *pencil of lines* is one-parameter family of lines, whose element is the line spanned by the point a and an arbitrary point on L. Let p, q be points which span L. Then, elements of the pencil can be written by

$$(\alpha p + \beta q) \wedge a = \alpha(p \wedge a) + \beta(q \wedge a),$$

where $\alpha, \beta \in \mathbf{R}$. Here, $(p \wedge a), (q \wedge a), \alpha(p \wedge a) + \beta(q \wedge a)$ are points in M_2^4. So, the pencil is one-dimensional subspace contained in M_2^4. Conversely, let X, Y be points in M_2^4, then X, Y are intersecting lines if and only if the one-dimensional subspace spanned by X, Y is contained in M_2^4. This is because $\langle Z, Z \rangle = 0 \Leftrightarrow \langle X, Y \rangle = 0$, where $Z = \alpha X + \beta Y$.

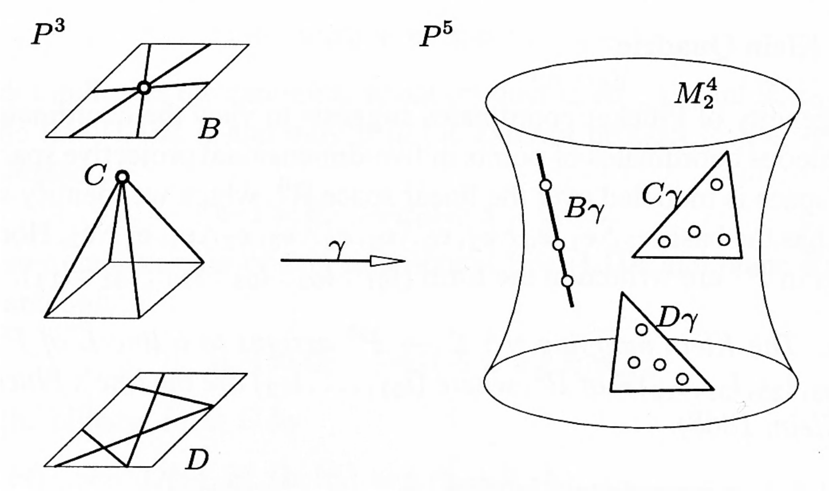


Fig. 4. Klein quadric M_2^4 [23]. B is a pencil of lines in P^3 and the Klein image $B\gamma$ is a line in M_2^4. Similarly, C is a bundle of lines and $C\gamma$ is a two-dimensional subspace. D is a field of lines and $D\gamma$ is a two-dimensional subspace

Bundle and Field of Lines. Let a be a point and A be a non-incident plane in P^3. The set of lines concurrent in a are called a *bundle of lines.* Let x, y, z be different points on A. Then, the element of the bundle can be written by

$$a \wedge (\alpha_0 x + \alpha_1 y + \alpha_2 z) = \alpha_0(a \wedge x) + \alpha_1(a \wedge y) + \alpha_2(a \wedge z),$$

where $\alpha_i \in \mathbf{R}$. So, the bundle of lines is a two-dimensional subspace contained in M_2^4. The dual of a bundle of lines is a *field of lines*, where a field of lines means the set of all lines which are contained in a plane in P^3. Let x, y, z be on the plane. Cleary, the element of the field can be written by

$$\alpha_0(x \wedge y) + \alpha_1(y \wedge z) + \alpha_2(z \wedge x).$$

So, the field of lines is a two-dimensional subspace contained in M_2^4. As is the case with the pencil of lines, there are no other two-dimensional subspace contained in M_2^4. Fig. 4 shows a pencil B and a bundle C and a field D.

Ruled Surface

Definition 1. *A family $R(u)$ of lines in P^3 is called a ruled surface, if its Klein image $R\gamma(u) = \gamma(R(u))$ is a curve in M_2^4. We say that $R(u)$ is C^r-continuous if $R\gamma(u)$ is C^r-continuous. A line of $R(u)$ is called a generator line of the ruled surface.*

Here, let $R(u) = a(u) \wedge b(u)$, where $a(u), b(u)$ are families of points in P^3. Locally, $\exists \varepsilon_i, a(u), b(u)$, can be obtained by $a(u) = R(u) \cap \varepsilon_1, b(u) = R(u) \cap \varepsilon_2$, where ε_i are two different planes. Any pair of $a(u), b(u)$ is called a pair of director curves. Then, $R\gamma(u)$ is C^r-continuous if and only if $a(u), b(u)$ are C^r-continuous. Fig. 5 shows examples of ruled surfaces.

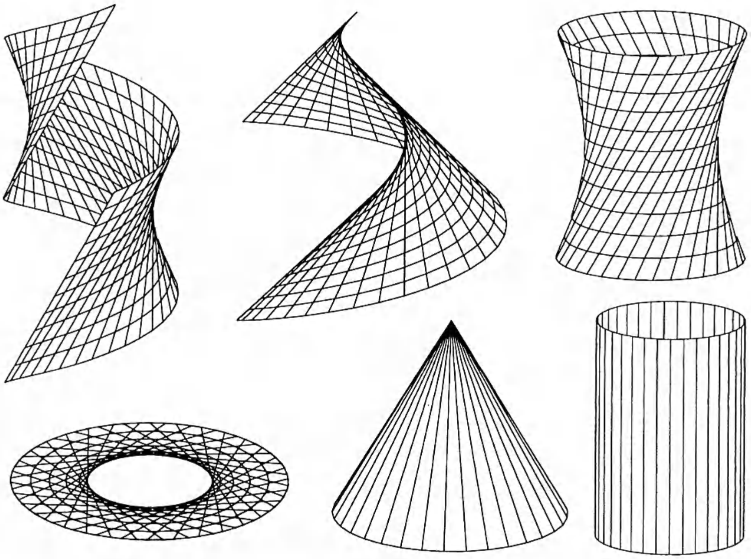


Fig. 5. Ruled surfaces [23]

5 Line Subdivision

A two-parameter family of lines in P^3 is called a *line congruence*. Klein images of line congruences are two-dimensional manifolds in M_2^4.

Remember that the ruled surface is defined as the union of one-parameter family of lines. However, at the same time, we can define the ruled surface as the envelope surface of the lines. Here, let $R(y_0)$ be the tangent line of the envelope surface at $p(y_0)$, then 'envelope surface of lines $R(y)$' is defined as the union of tangent points $p(y_0)$ such that $\lim_{||\delta|| \to 0} p(y_0 + \delta) - p(y_0) = 0$, where $p(y_0 + \delta)$ is the tangent point of the line $R(y_0 + \delta)$ (In the case of ruled surface, y and δ are one-parameters. On the other hand, in the case of the envelope surface of a line congruence, y and δ are local two-parameters.). In the case of ruled surface, any point of a line of R is the tangent point of the envelope surface.

So, similarly, we can consider the envelope surface of a line congruence. Let $E(K)$ be the envelope surface of the line congruence K. Let L be a point of a two-dimensional manifold $K\gamma$ in M_2^4. We can take a local coordinate system (y_1, y_2) on $K\gamma$ at L. Let $R(y_1)$ be the ruled surface which is a curve in M_2^4, whose direction is y_1, passing through L. Let $R(y_1, y_2)$ be the perturbation of $R(y_1)$ in the direction y_2. Then, we define $E(K)$ as the envelope of ruled surfaces $R(y_1, y_2)$.

In general, even if $K\gamma$ is a two-dimensional manifold, $E(K)$ is not necessarily a two-dimensional manifold.

5.1 Line Subdivision Surface

In what follows, we consider only stationary subdivisions. Let $|K\gamma|$ be a mesh in P^5, p_i^0 be vertices of $|K\gamma|$, p_i^0 be in M_2^4. $p_i^j = (p_{i0}^j, p_{i1}^j, p_{i2}^j, p_{i3}^j, p_{i4}^j, p_{i5}^j)$. Let p^j be the vector whose row is p_i^j, $\bar{p}^j$ be the vector whose row is $(p_{i1}^j, p_{i2}^j, p_{i3}^j, p_{i4}^j, p_{i5}^j)$, S be a subdivision matrix.

Then, we define a subdivision step as

$$\bar{p}^{j+1} = S\bar{p}^j,$$

$p_{i0}^{j+1} = -(p_{i1}^{j+1}p_{i4}^{j+1} + p_{i2}^{j+1}p_{i5}^{j+1})/p_{i3}^{j+1}$. So, any p_i^{j+1} is in M_2^4. Thus, we can see that the limit surface of this subdivision scheme is in M_2^4.

Then, the limit surface $f[p]\gamma(y)$ in M_2^4 is written:

$$f[p]\gamma(y) = \sum_i p_i^0 \phi_i(y),$$

$$\phi_i(y) \in \mathbf{R},\ y \in |K\gamma|, p = (p_0^0, p_1^0, \cdots).$$

Here, y is a local two-parameter (locally, $y = (y_1, y_2)$). Let $f[p](y)$ be the line congruence corresponding to $f[p]\gamma(y)$. Therefore, we define $E(f[p](y))$ as the *line subdivision surface.*

Degeneration of Line Subdivision Surfaces. Assume the line subdivision surface $E(f[p](y))$ is a two-dimensional manifold. We define a point of $E(f[p](y))$ as non-degenerate if the tangent line, which is a element of the line congruence $f[p](y)$, of the point is unique and if a continuous subset of $E(f[p](y))$ whose points do not share a tanget line. In short, if different points of $f[p]\gamma(y)$ have the same tangent point of $E(f[p](y))$ or if there exists a continuous subset of $E(f[p](y))$ whose point shares a tanget line, then the tangent point is degenerate. Therefore, the degenerated subset of $E(f[p](y))$ is a curve, which is a subset of ruled surface or a ruled surface.

Smoothness of Line Subdivision Surfaces. Let $f[p]\gamma(y_0)$ be a point of $f[p]\gamma(y)$, and $f[p](y_0)$ be the line corresponding to $f[p]\gamma(y_0)$. Since $f[p]\gamma(y_0)$ is in M_2^4, $\exists(a(y_0), b(y_0)), f[p](y_0) = a(y_0) \wedge b(y_0)$, where $a(y_0), b(y_0)$ are points in P^3.

Let $u(y_0)$ be a point of $E(f[p](y))$. Now, $E(f[p](y))$ is an envelope surface. So, there exists a line $f[p](y_0)$ such that $f[p](y_0)$ is the tangent line at $u(y_0)$. Thus, we can assume that $a(y_0) = u(y_0)$, $b(y)$ is in a plane A. Then, clearly, the differentiability class of $E(f[p](y))$ is equal to that of $a(y)$.

However, we want to know the relation between the differentiability class of $E(f[p](y))$ and that of $b(y)$.

Here, we get the following theorem. In this theorem, a is not necessarily the tangent point of $E(f[p](y))$.

Theorem 1 (Smoothness of line subdivision surfaces). *Assume that the line subdivision surface $E(f[p](y))$ is a two-dimensional manifold and non-degenerate. Then, $E(f[p](y))$ is tangent plane continuous if $a(y), b(y)$ are C^1-continuous. Additionally assume that the normal of tangent plane of $E(f[p](y))$ is continuous. Then, $E(f[p](y))$ is C^r-continuous if and only if $a(y), b(y)$ are C^r-continuous, that is, $f[p]\gamma(y)$ is C^r-continuous.*

Proof. Here, $\partial_1 f[p](y) = \partial_1 a(y) \wedge b(y) + a(y) \wedge \partial_1 b(y)$, where ∂_1 is differentiation on direction y_1. Let δ be a perturbation on the direction y_1. Now, there exists a direction y_1 such that the tangent line $f[p](y_0+\delta)$ at $u(y_0+\delta)$ and $f[p](y_0)$ do not span a plane which is vertical to the tangent plane at $u(y_0)$. At the limit $||\delta|| \to 0$, $\dot{a}(y_0)$ and $f[p](y_0)$ spans the tangent plane at $u(y_0)$. So, $E(f[p](y))$ is tangent plane continuous if $a(y), b(y)$ are C^1-continuous. Moreover, if the normal of tangent plane of $E(f[p](y))$ is continuous (see the chapter 'inflection plane' in [22]), the differentiability class of $E(f[p](y))$ is equal to that of $a(y), b(y)$. So, the differentiability class of $E(f[p](y))$ is equal to that of $f[p]\gamma(y)$. □

As above, the tangent plane at $u(y_0)$ depends on $f[p](y_0)$ and $\dot{a}(y_0)$. Thus, the set of the tangent planes of $E(f[p](y))$ is C^{r-1}-continuous if and only if $f[p]\gamma(y)$ is C^r-continuous.

In this proof, we do not use terms of subdivision. So, for any line congruence K, theorem 1 holds.

Boundedness of Line Subdivision Surfaces. Let $f[p]\gamma(y)$ be C^1-continuous and boundaryless, the normal of tangent plane of $E(f[p](y))$ be continuous. Then, if $E(f[p](y))$ is non-degenerate, $E(f[p](y))$ is boundaryless and $a(y), b(y)$ are C^1-continuous. So, the tangent plane $T(y_0)$ of $E(f[p](y))$ at $a(y_0)$ is spanned by $a(y_0) \wedge b(y_0)$ and $a(y_0) \wedge (a(y_0) + \dot{a}(y_0)) = a(y_0) \wedge \dot{a}(y_0)$. Let $l'(y)$ be the direction vector of $f[p](y)$.

Then, we get the following theorem.

Theorem 2 (Boundedness of line subdivision surfaces). *Assume that $f[p]\gamma(y)$ is C^1-continuous and boundaryless and $E(f[p](y))$ is a two-dimensional manifold and non-degenrate and the normal of tangent plane of $E(f[p](y))$ is continuous and $\forall y \in |K\gamma|, l'(y) \neq (0,0,0)$. Then, $E(f[p](y))$ is bounded if and only if any ideal point in P^3 is contained in at least one $T(y)$.*

Proof. $E(f([p](y))$ is C^1-continuous from Theorem 1. So, the set of $T(y)$ is C^0-continuous and the set of normals of $T(y)$ is C^0-continuous. Now, $l'(y) \neq (0,0,0)$. So, we can take $a(y), b(y)$ such that $a(y)$ or $b(y)$ is not an ideal point. Thus, $T(y)$ is not an ideal plane (Ideal planes are spanned by ideal points). Therefore, since $E(f[p](y))$ is a boundaryless two-dimensional manifold, if $E(f[p](y))$ is bounded, any ideal point in P^3 is contained in at least one $T(y)$. Moreover, since $E(f[p](y))$ is non-degenerate and C^1-continuous, $E(f[p](y))$ is bounded if any ideal point in P^3 is contained in at least one $T(y)$. □

5.2 Structure of Subdivisions

As above, vertices of a mesh in P^5 made by subdivision are in M_2^4. However, edges and faces of that are not necessarily in M_2^4. So, the lines in P^3 corresponding to the mesh do not necessarily form a mesh in P^3.

So, we consider a condition for the line congruence made by subdivision to be a mesh in P^3.

Let M be a mesh in P^3 and $M\gamma$ be the Klein image of M. Fig. 6 shows the relation between M and $M\gamma$. Vertices of M can be regarded as bundles. Then, Klein images of vertices are two-dimensional subspaces (faces of $M\gamma$). Similarly,

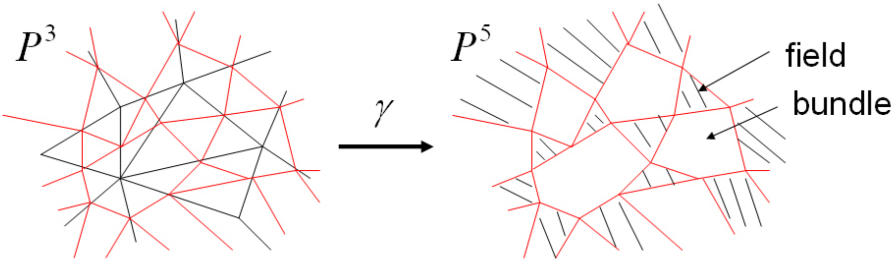


Fig. 6. $M\gamma$ is a mesh in M_2^4. The left picture shows a mesh M in P^3. The right picture shows the corresponding mesh $M\gamma$ in $M_2^4 \subset P^5$. $M\gamma$ consists of bundles and fields. Bundles corresponds vertices of M. Fields corresponds faces of M. So, the faces which are the Klein images of vertices and the faces which are that of faces appear alternately. Dached lines show the relation of M and $M\gamma$

faces of M can be regarded as fields, and hence Klein images of faces of M are faces of $M\gamma$.

Let $M\gamma^j$ be the j-th mesh made by subdividing $M\gamma$. Now, $M\gamma$ is in M_2^4. So, vertices v_{ik} of a face f_i of $M\gamma$ can be written:

$$v_{ik} = a_k X_i + b_k Y_i + c_k Z_i,$$

where $a_k, b_k, c_k \in \mathbf{R}$, X_i, Y_i, Z_i is points in M_2^4.

Here, we redefine a subdivision step as

$$p^{j+1} = Sp^j.$$

(See previous subsection. This is a more natural definition. However, in general, the linear combination of points in M_2^4 is not in M_2^4. If the linear combination is in M_2^4, lines corresponding to the points intersect in P^3. In the following case, this definition is equivalent to the previous definition.) Here, $M\gamma$ is the original mesh.

So, we consider linear combinations of the vertices $M\gamma$. If f_{i_0} and f_{i_1} are bundles, then the linear combination between v_{i_0k} and v_{i_1k} is in a bundle. This is clear, because, without loss of generality, we can assume that the bundles share the non-incident plane A (see the paragraph:Bundle and Field of Lines). Similarly, If f_{i_2} and f_{i_3} are fields, then the linear combination between v_{i_2k} and v_{i_3k} is in a field. This is because, without loss of generality, we can assume that y, z are on the line which is shared by the two faces corresponding to f_{i_2} and f_{i_3}.

However, if f_{i_4} is a bundle and f_{i_5} is a field, then the linear combination between v_{i_4k} and v_{i_5k} is neither a bundle nor a field.

Therefore, if we want $M\gamma^j$ consists of bundles and fields, we must define the subdivision based on bundles or fields.

Let M^j in P^3 be the mesh corresponding to $M\gamma^j$ in M_2^4. Then, $M\gamma^\infty$ is a line congruence made by subdivision and M^∞ is the line subdivision surface.

Here, we easily find the structure of subdivisions.

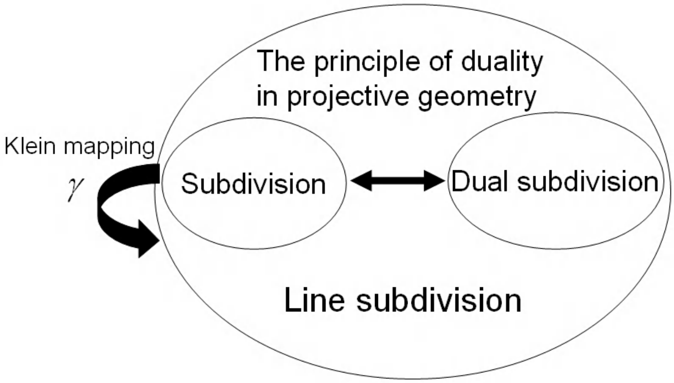


Fig. 7. The structure of subdivisions. If a line subdivision scheme makes a mesh $M\gamma$ in M_2^4 which corresponds a mesh M to in P^3 at finite step of subdivision, then the line subdivision scheme is a ordinary subdivision scheme or a dual subdivision scheme

The line subdivision based on bundles is the ordinary subdivision (here, we assume that ordinary subdivision generate 'flat' faces.). This is clear (Consider the linear combination of bundles. 'Bundle' naturally depends only the point a in P^3.). Similarly, the line subdivision based on fields is the dual subdivision (Consider the linear combination of fields. 'Field' naturally depends only the face in P^3.). Fig. 7 shows the structure of subdivisions.

Let $T(y_0)$ be the tangent plane of $M\gamma^\infty$ at $p(y_0)$ such that $T(y_0)$ is in M_2^4. Here, we easily see that if $T(y_0)$ is the Klein image of a bundle, then $\gamma^{-1}(T(y_0))$ is a bundle based on the vertex whose tangent line is $\gamma^{-1}(p(y_0))$, if $T(y_0)$ is the Klein image of a field, then $\gamma^{-1}(T(y_0))$ is a field based on the tangent plane of M^∞ which contains $\gamma^{-1}(T(y_0))$.

6 Conclusions

In this paper, we proposed a new subdivision method based on line geometry. The new subdivision scheme acts on the line space and generates two-dimensional manifolds in the Klein quadric M_2^4. Two-dimensional manifolds in M_2^4 are the Klein images of line congruences, which are local two-parameter families of lines in P^3. Then, we defined the line subdivision surface as the envelope surface of the line congruence.

Moreover, we derived theorems on smoothness and boundedness of the line subdivision surfaces.

However, since the line subdivision schemes make a mesh in P^5, only whose vertices are in M_2^4, at finite stage of line subdivision, in general, the line subdivision surfaces are not meshes. So, we want to know a condition for the line subdivision surface at finite stage to be a mesh.

In fact, there exists line subdivision schemes which satisfy this condition. Such line subdivision schemes are the line subdivision schemes based on bundles or fields. The line subdivision schemes based on bundles are equivalent to ordinary subdivision schemes. The line subdivision schemes based on fields are equivalent to dual subdivision schemes [22].

Acknowledgments. This work is supported by the 21st Century COE Program on Information Science and Technology Strategic Core at the university of Tokyo, and the Grant-in-Aid for Scientific Research of the Japanese Society for the Promotion of Science.

References

1. Lee, A., Moreton, H., Hoppe, H.: Displaced subdivision surface. In: SIGGRAPH '00 Proceedings, ACM (2000) 85–94
2. Loop, C.T.: Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics (1987)
3. Derose, T., Kass, M., Truong, T.: Subdivision surfaces in character animation. In: SIGGRAPH '98 Proceedings, ACM (1998) 85–94
4. Warren, J., Weimer, H.: Subdivision Methods for Geometric Design: A Constructive Approach. Morgan Kaufmann Publishers (1995)
5. Stam, J.: Evaluation of loop subdivision surfaces. In: SIGGRAPH 98 Conference Proceedings on CDROM, ACM (1998)
6. Zorin, D.: Smoothness of stationary subdivision on irregular meshes. Constructive Approximation **16** (2000) 359–397
7. Reif, U.: A unified approach to subdivision algorithms near extraordinary points. Computer Aided Geometric Design **12** (1995) 153–174
8. Prautzsch, H.: Analysis of c^k-subdivision surfaces at extraordinary points. Preprint. Presented at Oberwolfach (1995)
9. Cavaretta, A.S., Dahmen, W., Micchell, C.A.: Stationary subdivision. Memoirs Amer. Math. Soc. **93(453)** (1991)
10. Goodman, T.N.T., Micchell, C.A., D., W.J.: Spectral radius formulas for subdivision operators. In: L. L. Schumaker and G. Webb, editors, Recent Advances in Wavelet Analysis, Academic Press (1994) 335–360
11. Zorin, D.: Subdivision and Multiresolution Surface Representations. PhD thesis, University of California Institute of Technology (1997)
12. Reif, U.: A degree estimate for polynomial subdivision surfaces of higher regularity. Proc. Amer. Math. Soc. **124:2167–2174** (1996)
13. Doo, D., Sabin, M.A.: Behaviour of recursive subdivision surfaces near extraordinary points. Computer Aided Geometric Design **10** (1978) 356–360
14. Lounsbery, M., Derose, T., Warren, J.: Multiresolution analysis for surfaces of arbitrary topological type. ACM Transactions on Graphics **16** (1997) 34–73
15. Eck, M., DeRose, T., Duchamp, T.: Multiresolution analysis of arbitrary meshes. In: SIGGRAPH '95 Proceedings, ACM (1995) 173–182
16. Stollnitz, E.J., DeRose, T.D., Salesin, D.H.: Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann Publishers (1996)
17. Guskov, I., Sweldens, W., Schroder, P.: Multiresolution signal processing for meshes. In: SIGGRAPH '99 Proceedings, ACM (1999) 325–334
18. Guskov, I., Khodakovsky, A., Schroder, P., Sweldens, W.: Hybrid meshes: multiresolution using regular and irregular refinement. In: Proceedings of the eighteenth annual symposium on Computational geometry, ACM Press (2002) 264–272

19. Gu, X., Gortler, S.J., Hoppe, H.: Geometry images. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM (2002) 355–361
20. Claes, J., Beets, K., Reeth, F.V.: A corner-cutting scheme for hexagonal subdivision surfaces. In: Proceedings of Sape Modeling International 2002, IEEE (2002) 13–24
21. Farin, G., Hoschek, J., Kim, M.: Handbook of Computer Aided Geometric Design. Elesvier Science Publishers (2002)
22. Kawaharada, H., Sugihara, K.: Dual subdivision a new class of subdivision schemes using projective duality. METR 2005-01, The University of Tokyo (2005)
23. Pottman, H., Wallner, J.: Computational Line Geometry. Springer (2001)
24. Dietz, R., Hoschek, J., Juttler, B.: An algebraic approach to curves and surfaces on the sphere and on other quadrics. Computer Aided Geometric Design **10** (1993) 211–229
25. Dietz, R., Hoschek, J., Juttler, B.: Rational patches on quadric surfaces. Computer Aided Geometric Design **27** (1995) 27–40
26. Sun, K.A., Juttler, B., Kim, M.S., Wang, W.: Computing the distance between two surfaces via line geometry. In: the Tenth Pacific Conference on Computer Graphics and Applications, Los Alamitos (2002)

Euclidean Voronoi Diagrams of 3D Spheres: Their Construction and Related Problems from Biochemistry

Deok-Soo Kim[1], Donguk Kim[2], and Youngsong Cho[2]

[1] Department of Industrial Engineering, Hanyang University,
17 Haengdang-dong, Seongdong-gu, Seoul 133-791, South Korea
dskim@hanyang.ac.kr

[2] Voronoi Diagram Research Center, Hanyang University,
17 Haengdang-dong, Seongdong-gu, Seoul 133-791, South Korea
{donguk, ycho}@voronoi.hanyang.ac.kr

Abstract. Voronoi diagrams have several important applications in science and engineering. While the properties and algorithms for the ordinary Voronoi diagrams of point sets have been well-known, their counterparts for a set of spheres have not been sufficiently studied.

In this paper, we present properties and two algorithms for Voronoi diagrams of 3D spheres based on the Euclidean distance from the surface of spheres. Starting from a valid initial Voronoi vertex, the *edge-tracing algorithm* follows Voronoi edges until the construction is completed. The *region-expansion algorithm* constructs the desired diagram by successively expanding the Voronoi region of each sphere, one after another, via a series of topology operations, starting from the ordinary Voronoi diagram for the centres of spheres.

In the worst-case, the edge-tracing algorithm takes O(mn) time, and the region-expansion algorithm takes O($n^3 \log n$) time, where m and n are the numbers of edges and spheres, respectively. It should, however, be noted that the worst-case time complexity for both algorithms reduce to O(n^2) for proteins since the number of immediate neighbor atoms for an atom is constant. Adapting appropriate filtering techniques to reduce search space, the expected time complexities can even reduce to linear.

Then, we show how such a Voronoi diagram can be used for solving various important geometric problems in biological systems by illustrating two examples: the computation of surfaces defined on a protein, and the extraction and characterization of interaction interfaces between multiple proteins.

1 Introduction

Since its introduction in the early 20th century, the Voronoi diagram has been one of the central topics in computational geometry as well as in other disciplines in science and engineering. Due to its natural descriptive and manipulative capability, Voronoi diagrams and their variations have been known by various names

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 255–271, 2005.

such as Thiessen polygons [39], medial axis transformations (MAT) [27], symmetric axis transformations (SAT) [4,5], skeletons [25], proximity maps [17], Dirichlet tessellations, etc.

The ordinary Voronoi diagram for a point set and its construction have been studied extensively in both two and higher dimensions [33]. However, the construction of the Voronoi diagram for spheres in Euclidean distance metric, often referred to as an *additively weighted Voronoi diagram* [33], has not been explored sufficiently even though the potential impact of this Voronoi diagram on diverse applications can be significant [1,15,30,36,40]. For example, the structural analysis of protein requires an efficient computational tool to analyze the spatial structure among its atoms [15,20,36]. In the design of new materials, a similar analysis is fundamental as well [29,31,38]. However, due to the lack of appropriate algorithms and stable running codes for the Euclidean Voronoi diagram of spheres, most applications have instead adapted an ordinary Voronoi diagram of points, a power diagram [2,3], or an α-shape [11,12].

In this paper, we present properties and two algorithms for Voronoi diagrams of 3D spheres based on the Euclidean distance from the surface. Starting from a valid initial Voronoi vertex, the *edge-tracing algorithm* follows Voronoi edges until the construction is completed. The *region-expansion algorithm* constructs the desired diagram by expanding Voronoi regions for one sphere after another via a series of topology operations, starting from the ordinary Voronoi diagram for the centres of spheres. In the worst-case, the edge-tracing algorithm takes $O(mn)$ time, and the region-expansion algorithm takes $O(n^3 \log n)$ time, where m and n are the numbers of edges and spheres, respectively. It should be noted, however, that the worst-case time complexity for both algorithms reduce to $O(n^2)$ for proteins since the number of immediate neighbor atoms for an atom is constant [16]. Adapting appropriate filtering techniques to reduce the search space, the expected time complexities can even reduce to linear.

Then, we show how such a Voronoi diagram can be used for solving various important geometric problems in biological systems by illustrating two examples: the computation of surfaces defined on a protein, and the extraction and characterization of interaction interfaces between multiple proteins.

This paper is organized as follows: After reviewing related previous research in Section 2, we provide the definition and a few geometric properties of the Euclidean Voronoi diagram for spheres in Section 3. Then two algorithms, the edge-tracing and the region-expansion algorithms, are presented in Section 4. To show the powerful use, we illustrate two, among many others, important biological problems well-solved via the Voronoi diagrams of atoms. Then, we conclude this paper.

2 Related Works for the Euclidean Voronoi Diagram of 3D Spheres

Unlike other Voronoi diagrams and their variants, few reports are available for the Voronoi diagram of spheres. Aurenhammer discussed the transformation of the computation of the Euclidean Voronoi diagram of spheres in d-dimensions to that of the $(d+1)$-dimensional power diagram obtained from the convex hull in $(d+2)$-dimension [2].

Will wrote a comprehensive Ph.D. thesis dedicated to the computation of Voronoi regions in the Euclidean Voronoi diagram of spheres in 3D [42]. In his Ph.D. thesis, Will showed that the Voronoi region of a sphere has a $\Theta(n^2)$ combinatorial complexity and proposed a lower envelope algorithm which takes $\mathrm{O}(n^2 \log n)$ expected time for a single Voronoi region, where n is the number of spheres. Will also provided experimental results on various data sets from biological problems. In this work, he considered a general set of spheres meaning that there is no constraint on the size distribution of spheres.

Gavrilova, in her Ph.D. thesis, reported several properties of Euclidean Voronoi diagrams for spheres in arbitrary dimensions, including shapes of the Voronoi regions, nearest neighbors and empty-sphere properties [13, 14]. Luchnikov et al. proposed a practical idea of tracing edges which is a simple yet powerful way to obtain the desired diagram [29].

Recently, Kim et al. reported on the details of an edge-tracing algorithm and its full implementation for constructing the whole Voronoi diagram with discussions on various applications including the analysis of protein structures [20, 21, 22, 23]. They showed that the whole Voronoi diagram can be constructed in $\mathrm{O}(n^3)$ time in the worst-case. Boissonnat and Karavelas reported on an elegant algorithm based on the convex-hull of spheres transformed by inversion [6]. Kim and Kim recently are currently working on another algorithm, the region-expansion algorithm, for the problem [24].

The combinatorial complexity of the Euclidean Voronoi diagram of spheres is also important to mention. While the numbers of vertices, edges, and faces of the Voronoi diagram of general spheres are all $\mathrm{O}(n^2)$ in the worst-case, the average numbers for those are all $\mathrm{O}(n)$. Halperin even found that the upper bound of the combinatorial complexity for all of the vertices, edges, and faces of the Voronoi diagram for atoms in a protein is $\mathrm{O}(n)$ in the worst-case [16]. This nice property is due largely to two characteristics of atom distributions in a protein. According to Pauli's exclusion principle, an atom cannot contain another [32]. In addition, the differences in the atom radii are within a constant since most proteins consist of six different types of atoms, such as H, C, N, O, P, and S, with the corresponding van der Waals radii of 1.2, 1.7, 1.55, 1.52, 1.8, and 1.8 Å, respectively [43]. Under these conditions, Halperin showed that the number of neighboring atoms, which define Voronoi faces, of an atom is linear in the worst-case.

3 Definitions Related to the Euclidean Voronoi Diagrams for Spheres

Let $S = \{s_1, s_2, \cdots, s_n\}$ be a set of generators where s_i is a three dimensional spherical ball. Hence, $s_i = (c_i, r_i)$ where $c_i = (x_i, y_i, z_i)$ and r_i denote the center and radius of a ball, respectively. We assume that no ball is completely con-

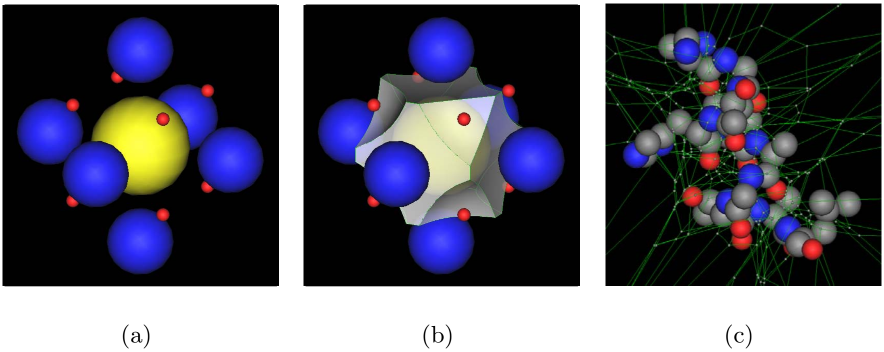


Fig. 1. Fifteen balls with three different radii and a corresponding Voronoi diagram. Voronoi edges are conics while Voronoi faces are hyperboloids: (a) fifteen generator balls, (b) the Voronoi region corresponding to the largest ball in the center, and (c) the Voronoi diagram for an α-helix with 67 atoms

tained inside another ball while other types of intersections between balls are allowed. Associated with each ball s_i, there is a corresponding region VR_i, called a *Voronoi region* for s_i, where $\mathrm{VR}_i = \{p | dist(p, c_i) - r_i \leq dist(p, c_j) - r_j, i \neq j\}$. Then, $\mathrm{VD}(S) = \{\mathrm{VR}_1, \mathrm{VR}_2, \cdots, \mathrm{VR}_n\}$ is called the *Euclidean Voronoi diagram for S*. In this paper, the ordinary L_2-distance from the surface of balls is used to define Euclidean Voronoi diagrams.

As in the ordinary Voronoi diagram, the Voronoi regions corresponding to balls on the boundary of the convex hull of S are unbounded. Other regions are bounded by *Voronoi faces*, where a Voronoi face is defined by two neighboring balls. Note that the geometry of a Voronoi face is always a hyperboloid of two sheets and a face always lies within only one sheet. Note that the face may degenerate into a plane when the balls are equi-sized. A Voronoi face intersects another face to form a *Voronoi edge*. Voronoi edges are conic and therefore planar. It can be shown that a Voronoi edge is the spine curve of a Dupin cyclide for three nearby spheres. When Voronoi edges intersect, a *Voronoi vertex* is defined. In this paper, we assume that the degree of a vertex is always four. Hence, there is a sphere centered at the vertex and simultaneously tangent to four balls. This *tangent sphere* is called *empty* since this sphere does not intersect any other balls but it is tangent to the four balls. From now on, the uses of *Euclidean* and *Voronoi* will be minimized when it does not cause any misunderstanding. In this paper, *balls* are used to denote generators and *spheres* are used to denote tangent spheres corresponding to Voronoi vertices. For further details, please refer to [20–23].

Illustrated in Fig. 1(a) and (b) are fifteen balls with three different radii and the Voronoi region corresponding to the largest ball in the center. As shown in this figure, Voronoi edges and faces are curved, and therefore Voronoi regions are not convex in general but are star-shaped with respect to the center of the corresponding ball. Note that the Voronoi region for the largest ball shares fourteen faces with its neighbors. Fig. 1(c) shows the Voronoi diagram of a subset of protein with 67 atoms, which form an alpha-helix, downloaded from PDB [35].

4 Algorithms for VD(S)

Before discussing the construction of the topology for VD(S), a discussion on computational primitives for the geometry aspect of vertices, edges, and faces is in order. A Voronoi vertex is the center of an empty sphere simultaneously tangent to four balls. An elegant algorithm to compute such a tangent sphere in a general dimension is presented by Gavrilova [14]. A Voronoi edge is defined as the locus of points equi-distant from three nearby balls, and it can be easily shown that an edge is always a conic curve and can be conveniently represented, if necessary, in a rational quadratic Bézier form [23]. Two topologically neighboring balls define a Voronoi face as a locus of points equi-distant from two nearby balls, and the face is a segment of a hyperboloid of two sheets and its equation can be easily obtained [15, 23]. Note that the topology of a face can be arbitrary such as a triangle, rectangle, pentagon, and so on. While a face is connected, a face may have a number of holes.

The data structure we have designed to store the topology of VD(S) is a variation of radial data structure [7, 41]. It should be noted that an edge-graph of VD(S) can be disconnected [23]. Hence, the dual of VD(S) is not necessarily a valid triangulation like the Delaunay triangulation for the Voronoi diagram of a point set or the regular triangulation for a power diagram. We call the dual of VD(S) a *quasi-triangulation* since it is a valid triangulation except around the tiny spheres located in-between two larger spheres. The details on the quasi-triangulation will be discussed later in another paper.

4.1 Edge-Tracing Algorithm

The idea of the edge-tracing algorithm is simple as it constructs Voronoi edges in the order of a depth-first search. The algorithm first locates a true Voronoi vertex v_0 by computing an empty tangent sphere defined by four appropriate nearby balls. Given v_0, four edges e_0, e_1, e_2, and e_3 emanating from v_0 can be easily identified and pushed into a stack called an *Edge-stack*. Hence, these edges have v_0 as their starting vertices. After popping an edge from the stack, the algorithm computes the end vertex of the popped edge. Note that the end vertex can be found by computing an empty sphere tangent to four balls: three balls which define the popped edge and one of the $(n-3)$ candidate balls.

If an empty tangent sphere is found, the center of the sphere may become the end vertex of the popped edge. (Another condition is indeed necessary.) Once the end vertex of edge is found, it is also possible to define three new edges emanating from this new vertex. Hence, these new edges starting from the new vertex just computed are created and pushed into the Edge-stack. By following this process until the Edge-stack is empty, the computation of the Voronoi diagram of a connected graph is completed.

We want to mention here that the edge-tracing algorithm, as described here, does not find disconnected subgraphs, if they exist. Such cases have to be handled separately and will not be covered in this paper since it requires a detailed discussion.

Finding Vertices. Computing an end vertex of a popped edge is equivalent to finding a ball associated with the end vertex. To find an end vertex, we compute a tangent sphere from four balls: three balls defining the popped edge and another ball from the set of balls. After a tangent sphere is computed, its emptiness should be tested against all the other balls except the four balls tangent to the sphere. Note the three balls are always correctly determined when an edge is created.

A naïve solution for this task would take $O(n^2)$ time in the worst-case. However, it can be improved as follows. Given the three surrounding balls for an edge, we first compute a tangent sphere T_i with an arbitrary ball s_i from the $(n-3)$ candidate balls. Then, we select another candidate ball s_j and construct a tangent sphere T_j with s_j and the three surrounding balls. If s_j intersects T_i, the current T_i is replaced by T_j. If not, we choose one of T_i or T_j whichever is closer to the start vertex of the edge in terms of angular distance. Since all balls in the candidate set are scanned only once, this process runs in $O(n)$ time in the worst-case.

To find a closer tangent sphere, we define an angular distance as an angle defined by the start vertex, the center of one of the surrounding balls, and the center of a tangent sphere. Then, the smaller this angle is, the closer the tangent sphere is to the start vertex.

Stitching Vertices and Edges Together. Suppose that a new vertex for an edge is computed. If the vertex has not been previously computed, then we can safely use the new vertex to complete the edge. However, if it has already been computed, meaning that the tangent sphere to those four balls has been handled before, we have to ignore this new vertex and use the previously computed (and existing somewhere) vertex to complete the definition of the edge. This is because the existing vertex already has partially determined associated topological information. Note that an existing vertex implies that a loop (or cycle) is identified in the edge-graph of $\mathrm{VD}(S)$. Hence, it is necessary to check if the new vertex has already been computed or not.

For the efficient search for an existing vertex, we have devised a table of computed vertices called a *Vertex Index Dictionary* (VIDIC). An entry in the dictionary consists of indices of four balls defining a vertex and a pointer to the vertex. However, care should be taken since there can be some pairs of entries in the VIDIC where each pair has an identical combination of spheres.

The size of the dictionary, in the worst-case, is two times the number of Voronoi vertices and therefore $O(n^2)$. However, if an appropriate ordering among the entries is used, a binary search scheme can be applied to take $O(\log n)$ time in the worst case. We believe that a tighter worst-case bound of the size of the VIDIC can be much less than $O(n^2)$. In addition, we want to mention that hashing, taking $O(1)$ time on the average, is applicable for searching in VIDIC.

Algorithm. The proposed edge-tracing algorithm can be summarized as follows:

```
Algorithm Edge-Tracing
Input: A set S of 3D balls
Output: VD($S$)
  1. Find an initial vertex $v_0$, generate four edges emanating from $v_0$, then
     push the edges into an Edge-stack.
  2. Pop an edge from the Edge-stack and find the empty tangent sphere
     closest to the start vertex of the edge to define the end vertex.
  3. Check if the vertex found in Step 2 is valid or not using VIDIC.
  4. If the vertex exists in VIDIC, finalize the popped edge appropriately. If
     the vertex is new, generate three more edges and push them into the
     Edge-stack.
  5. Repeat Step 2 through 4 until the Edge-stack is empty.
```

Time Complexity. Given an initial Voronoi vertex to start with, the edge-tracing algorithm runs in $O(mn)$ time in the worst-case, where m is the number of edges and n is the number of balls. The algorithm iterates $O(m)$ times since it traces all edges once for each edge. For each edge, it is necessary to do $O(n)$ scans through all candidate balls once to compute a valid tangent sphere. Detection of the existence of a vertex in VIDIC can be done in $O(\log n)$ time in the worst-case by a binary search. Even though m is $O(n^2)$ in the worst-case, it is $O(n)$ on the average.

While the numbers of vertices, edges, and faces are $O(n^2)$ in the worst-case, the average numbers of vertices, edges, and faces are $O(n)$. In addition, if we devise an appropriate geometric hashing, an end vertex can be found in only $O(1)$ time on the average when balls are accordingly distributed like protein data. Note that the preprocessing for an appropriate bucket takes $O(n)$ time in the worst case. Searching for a vertex in VIDIC can also be reduced to $O(1)$ time on the average if an appropriate hashing is used. Hence, we claim that the average time complexity for the whole procedure can be as low as $O(n)$. For this claim to be mathematically justified, however, the distribution of balls should be carefully examined. For further details about the algorithm, readers are referred to [23].

4.2 Region-Expansion Algorithm

The region-expansion algorithm extends its precursor in 2D [18, 19] and adapts the idea of discrete event simulation [28]. Let VD(P) be an ordinary Voronoi diagram of the centers of balls. The algorithm constructs VD(S) by expanding Voronoi regions for one ball after another via a series of edge-flips, starting from VD(P). After choosing a point generator c_i, which is the center of a spherical ball s_i, and the corresponding Voronoi region VR_i, the algorithm continuously enlarges the point generator c_i to the ball s_i. Then, the corresponding region VR_i expands according to the enlargement of the corresponding generator. Repeating the process for all generators constructs a correct Voronoi diagram if the topology is consistently and correctly maintained.

The vertices on the boundary of an expanding region are called *on-vertices*, and the others are called *off-vertices*. The edges on the expanding region are called *on-edges*, and the edges which have no on-vertex are called *off-edges*. The other edges are called *radiating-edges* and categorized into two groups: i) edges with an on-vertex and an off-vertex, and ii) edges with on-vertices at both ends. Similarly, faces are also grouped into three categories: *on-faces*, *off-faces*, and *radiating-faces*. A *vertex sphere* is the sphere simultaneously tangent to the generators defining a Voronoi vertex, and therefore its center is identical to the vertex.

Expanding Regions and Topology Changes. Given VD(P), c_i is associated with a polyhedral Voronoi region VR$_i$. Given a set of balls $S = \{s_1, s_2, \cdots, s_n\}$, we can view each ball s_i as it grows, or *expands*, to its full size starting from a point c_i at its center. While a ball expands, the corresponding region expands as well. If we can keep the topology, and the geometry as well if necessary, among vertices, edges, faces, and regions for the intermediate Voronoi diagram correctly and consistently during the expansion, the complete VD(S) can be computed by repeating the process generator by generator to the last generator. We call this process the *region-expansion*.

Suppose that we choose a generator s_i, which will expand to its full size starting from a point at its center c_i. Then, s_i and the corresponding region VR$_i$ are called *expanding* while the other generators and regions are called *staying*. It is obvious that expanding a generator always increases the volume of the corresponding region. Note that each on-vertex, during the region-expansion, moves away from the initial region by following the radiating-edge associated with the vertex. Similarly, each on-edge moves away from the initial region by following a corresponding radiating-face.

Certain topological changes occur at some point in time during the expanding process. For example, suppose that an on-vertex moving along a radiating-edge meets a corresponding off-vertex of the edge. Then, the radiating-edge degenerates to a point and disappears afterwards. We call this an *event* for such a situation causing changes in the combinatorial structure.

It can be shown that considerations about the vertices and edges are sufficient to identify all events for topological changes during the expansion process. Furthermore, it is only necessary to consider edges on radiating-faces since on-vertices and on-edges are always restricted to movement along radiating-edges and radiating-faces, respectively. Note also that on-vertices and on-edges do not have any associated state. Since an event denotes a change in the topology structure due to moving on-vertices or on-edges, the next event always occurs at edges on the radiating-faces.

Note, however, that any edge except an on-edge can be associated with events if the size of the expanding generator is sufficiently large. Since generator balls have prescribed sizes, only a subset of the events can be realized.

Event Types. Events can be classified based on the conditions of an edge: i) *end-event*, ii) *mid-event*, and iii) *split-event*. An end-event denotes the case when an edge disappears at the end of the edge during the process of region expansion.

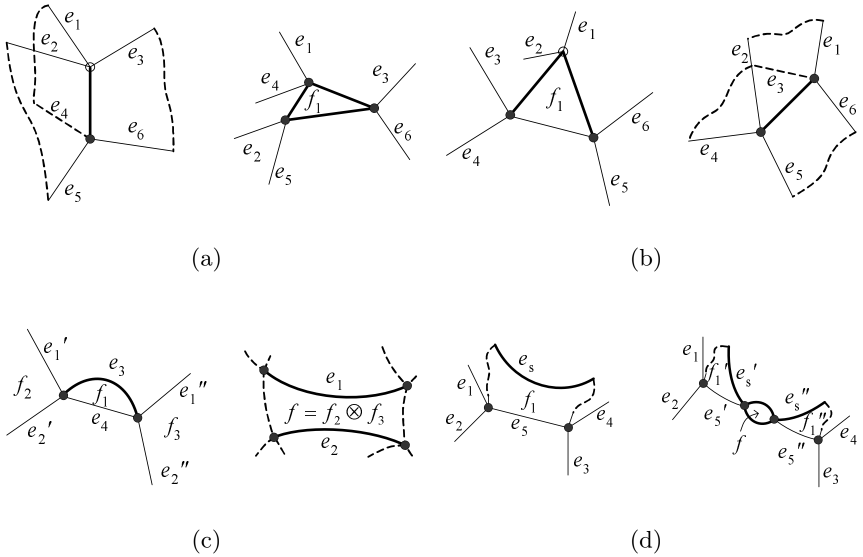


Fig. 2. Handling events: (a) one-end-event, (b) two-end-event, (c) mid-event, and (d) split-event

A mid-event denotes the case that an edge disappears in the middle of the edge. This case occurs when both end vertices (which are on-vertices) of an edge move toward the interior of the edge to meet at a point during the region expansion. Similarly, a split-event denotes the case that a new vertex is created at a point in the middle of an edge so that the edge splits into two edges.

Handling Events. Depending on the events, different actions should be performed to keep the topology correct and consistent. Fig. 2 shows examples of events. In the figure, white and black solid circles denote off-vertices and on-vertices, respectively. An end-event is further classified into two groups: *one-end-event* and *two-end-event*. In the case of an one-end-event in Fig. 2(a), the event edge (a radiating-edge), shown as the thick edge, disappears after the event is properly processed and three new on-edges are born to form a new on-face. In addition, both end vertices of the event edge disappear, and three new on-vertices for the new on-face are born.

A two-end-event, Fig. 2(b), removes two radiating-edges having the same type of events simultaneously and incident to an identical off-vertex. Then, the face, a radiating-face, which is bounded by both event edges also disappears. Therefore, three edges constituting the face disappear as well, but a new on-edge is born. Checking whether the end-event for an edge is indeed one-end-event or two-end-event can be done simply by looking at the event type of the next edge in an event queue Q.

Shown in Fig. 2(c) is the case of a *mid-event*. In the figure, e_3 denotes the edge with a mid-event and e_4 denotes an on-edge defining and bounding a radiating-face f_1 with e_3. When this event is processed, the radiating-face f_1 disappears, and two on-faces f_2 and f_3 are merged into one on-face f. Therefore, two edges e'_1 and e''_1 are merged into one edge e_1. The edges e'_2 and e''_2 are similarly handled. Therefore, this event removes one on-face and two on-edges from the topology under construction.

A *split-event* differs from the others as follows. As shown in Fig. 2(d), the event edge e_s does not disappear but is divided into two edges e'_s and e''_s. After processing the current split-event, e'_s and e''_s become the new radiating edges. Then, the future event types of e'_s and e''_s are tested. Similarly, the corresponding on-edge e_5 is divided into two edges e'_5 and e''_5.

In the case of a split-event, two new on-vertices, and two new on-edges are born as shown in Fig. 2(d). In addition, a new on-face f bounded by two new on-edges is also born. Therefore, processing a split-event creates two faces (one on-face and one radiating-face) and two on-vertices. The cardinality change for edges is, however, a little different from the previous events. After this event is appropriately handled, three on-edges and two radiating-edges are created, but one off-edge which is the current event edge is removed.

It is interesting to observe that the handlers (a) and (b) are dual to each other, and likewise (c) and (d). Each event is associated with an event time that the event may occur. The details of this region-expansion algorithm will be soon available [24].

Algorithm. The region-expansion algorithm for a single Voronoi region can be described as follows.

```
Algorithm Expand Region of $s_i$
Input: A set S of 3D balls
Output: A Voronoi region for a 3D ball $s_i$
  1. Find all edges defining radiating-faces and insert them into a set $E$.
  2. For each edge $e \in E \setminus E_{on}$, determine its event type and time $t_e$. If $t_e < r_i$,
     insert $e$ into an event queue $Q$.
  3. Pop an edge from $Q$ and check its event type. If the event type is an end-
     event, determine if it is an one-end-event or two-end-event by checking
     the next edge in $Q$.
  4. Perform appropriate actions for the detected event. After the treatment,
     find new edges bounding new radiating-faces and insert them into another
     edge set $E^*$. Perform Step 2 for all edges in $E^*$.
  5. Repeat Step 3 and 4 until $Q$ is empty.
```

Time Complexity. The computation necessary to expand a Voronoi region, starting from a center point to a complete ball, takes $O(n^2 \log n)$ time in the worst-case since there can be $O(n^2)$ number of edges and sorting is necessary according to the event time of each edge. Therefore, the whole Voronoi diagram can be constructed by the region-expansion algorithm in $O(n^3 \log n)$ time in the worst-case. We believe, however, that there should be a tighter bound for the worst-case time complexity and the expected time complexity can be much lower. We want to mention that reducing the size of all generators by the smallest generator at the very beginning enables the desired Voronoi diagram to be computed much faster. However, the computation reduction is a constant factor.

5 Applications of VD(S) in Biological Problems

To demonstrate the applicability of VD(S), we show how two important biology problems, among many others, in proteins can be solved efficiently via VD(S). Proteins consist of atoms. Given the atomic structures of molecules, analyzing their inherent spatial properties and interactions between molecules is important for understanding their biological functions. For example, the interaction between a protein and a small molecule is the basis of designing new drugs.

5.1 Computation of Molecular Surfaces

A protein is usually modelled as a set of hard spheres in order to represent atoms in a *space-filling model* where their radii are the van der Waals radii [10, 26]. Given such a model, there are usually two kinds of surfaces involved: a *solvent accessible surface(SAS)* and a *molecular surface(MS).* A SAS, first defined by Lee and Richards [26], is the set of centers of a spherical probe rolling around the protein. A probe is used for the computational convenience of a small molecule which interacts with the protein.

A MS, also known as a *Connolly surface* [8, 9], consists of the most inward points on the probe toward the interior of a protein when the probe is in contact with two or more atoms in the protein [8, 10, 37]. It is well-known that atoms located at the boundary of a protein determine the function of the protein [8, 9]. Hence, knowledge of a molecular surface is important in the study of protein functions since the surface has a direct relation with other atoms.

The molecular surface consists of two groups: a *solvent contact surface(SCS)* and a *reentrant surface(RS).* A SCS consists of points on the van der Waals atoms which are touchable by a probe. A RS is defined as points on the inward part of the surface of a probe, where the probe is in contact with atoms [37]. An RS, known as a blending surface in CAGD community, consists of two types of blending surface patches: *rolling blending* and *link blending.* These blending surface patches can be computed by rolling a spherical probe in every possible direction while keeping tangential contact with the atoms.

An efficient data structure for the proximity among atoms has to be accordingly devised for the efficient computation of the above-mentioned surfaces since queries about nearby atoms has to be correctly and efficiently answered. Previous studies have mainly used either an ordinary Voronoi diagram of center points of atoms [34, 37], a power diagram of atoms [3], or an α-hull [12].

Considering the fact that atoms constituting a protein have different sizes, the topology constructs in previous studies only provide close approximations to VD(S). Since a VD(S) is uniquely defined regardless of the probe size, the blending as well as the offsetting operations using different probes can be done with a uniquely defined VD(S).

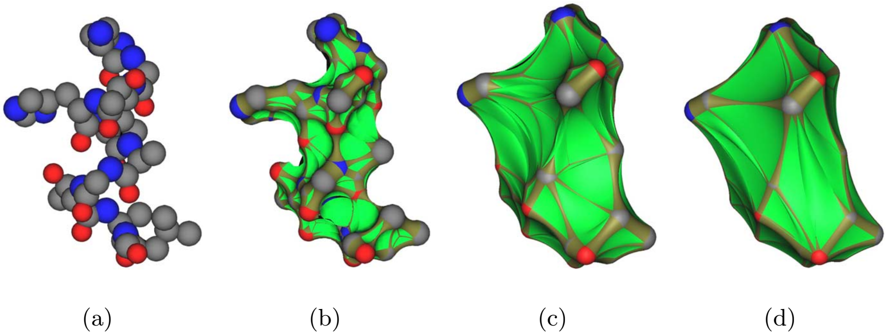


Fig. 3. Molecular surfaces for a subset of protein downloaded from PDB. (a) the molecule model, (b) the molecular surface using a probe with radius 1.4Å, (c) the radius is 4Å and (d) the radius is 8Å

Shown in Fig. 3(a) is a subset of protein data downloaded from PDB [35] which forms an α-helix. In the model, there are 41 C's, 14 N's, and 12 O's. Fig. 3(b), (c) and (d) show the molecular surfaces defined by different probes with radii of 1.4Å, 4Å, and 8Å, respectively. Note that all molecular surfaces are computed from the identical Euclidean Voronoi diagram of the model in Fig. 3(a).

5.2 Analyzing Interaction Interfaces

A protein is a macromolecule consisting of the permutation of 20 different kinds of amino acids. Amino acids are linearly connected to one another via peptide bonds to form chains. When a protein consists of two chains, it is called a *dimer* as shown in Fig. 4(a). Shown in Fig. 4(b) and (c) are examples of a *trimer* (three chains) and a *tetramer* (four chains), respectively. Since interaction among chains is critical for protein functions, understanding the interaction is getting more important and the geometric properties of the interactions are getting more attention.

The *interaction interface IIF* is defined in this paper as follows. Let $A = \{a_1, a_2, \ldots, a_m\}$, $B = \{b_1, b_2, \ldots, b_n\}$ be two chains in a protein, where a_i and b_j are atoms with appropriate centers and radii. The interaction interface between two chains A and B is defined as $IIF_\infty(A, B) = \{p \mid dist(p, A) = dist(p, B)\}$, where $dist(p, A)$ denotes the minimum Euclidean distance from p to the surfaces of all van der Waals atoms in the set A. Then, $IIF_\infty(A, B)$ is a subset of Voronoi faces in VD($A \bigcup B$). Hence, $IIF_\infty(A, B)$ can be easily located by simply checking each Voronoi face with its generating atom types. Note that $IIF_\infty(A, B)$ expands to infinity.

The infinite Voronoi faces in $IIF_\infty(A, B)$ are biologically less significant since proteins, as well as $IIF_\infty(A, B)$, are usually hydrated. Hence, we define a *trimmed interaction interface* $IIF(A, B)$ against a probe of a water molecule. Fig. 5(a) and (b) illustrate the van der Waals atoms of a dimer 1bh8 downloaded from PDB and the corresponding $IIF(A, B)$, respectively.

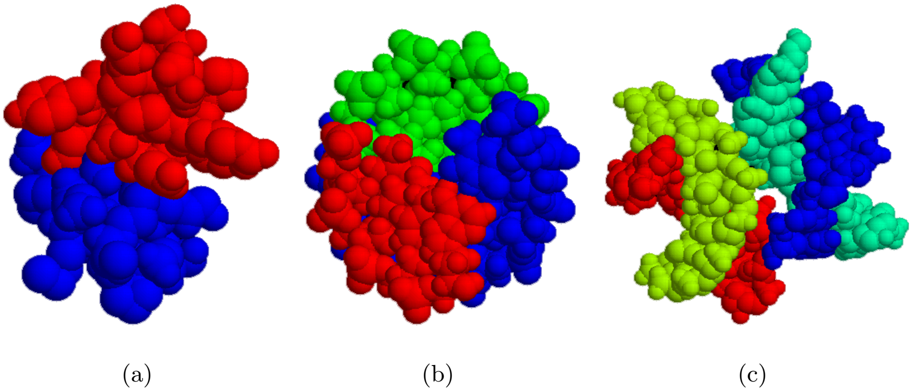


Fig. 4. Protein examples: (a) a dimer, (b) a trimer, and (c) a tetramer

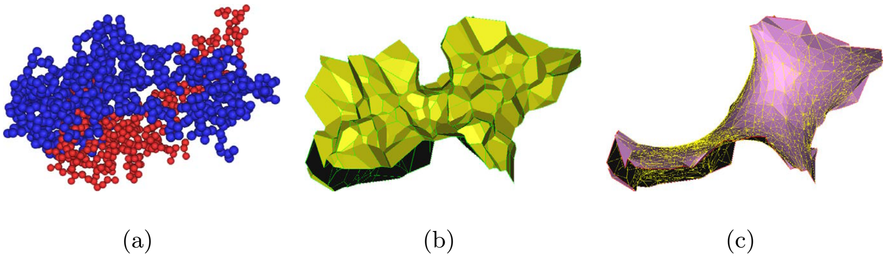


Fig. 5. A dimer (PDB ID: 1bh8): (a) van der Waals model of 1bh8, (b) $IIF(A, B)$, and (c) the corresponding $BS(A, B)$

To analyze the topology and geometry of an interaction interface, we define a *base surface BS(A, B)* which is defined as a smooth surface fitted through the trimming curve of $IIF(A, B)$ with a certain minimality condition. Fig. 5(c) shows the base surface $BS(A, B)$ of $IIF(A, B)$ in Fig. 5(b).

Once an interface and its corresponding base surface are obtained, various analyses can be done on the interaction behavior between chains in the protein. Fig. 6 shows an example an analysis of interaction interfaces from a topological point of view. Fig. 6(a) and (b) show that an interface may be disconnected, and Fig. 6(c) illustrates that an interface may even have a genus of one or more. Our experience shows that there are proteins with even stranger topological structures.

Once the IIF is computed, several other quantitative measures can be defined to characterize dimers. Shown in Fig. 7 is an example of the geometry analysis of the interaction interface which shows the distribution of areas of trimmed interaction interfaces IIF's for 3,561 dimers available in PDB. Note that the unit is Å^2. As shown in the figure, 1,353 dimers have areas between 1,000 and 2,000 Å^2. Only 40 dimers have areas larger than 5,000 Å^2. Similarly, the base surface

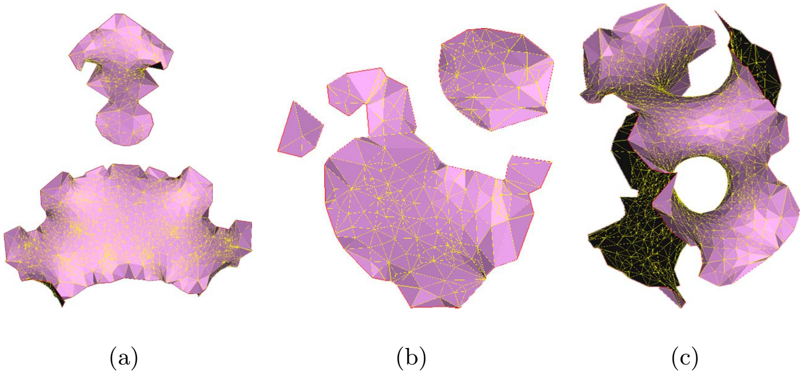


Fig. 6. Types of base surfaces: (a) base surface with two patches (PDB ID: 1aos), (b) base surface with three patches (PDB ID: 1hlk), and (c) base surface with tunnel (PDB ID: 1aoj)

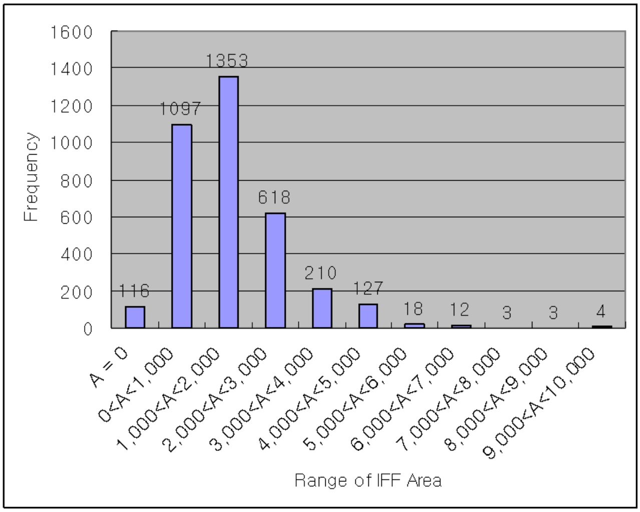


Fig. 7. The distribution of IIF's for 3,561 dimers in PDB

area BS can be computed. The ratio between BS and IIF can give information about the geometric complexity of the interaction between two separate groups of atoms in a protein. The distribution of distances of vertices in IIF from BS is also another measure of the interaction.

6 Conclusions

In this paper, we have presented properties and two algorithms for the Voronoi diagram of 3D spheres based on the Euclidean distance from the surface of spheres. Starting from a valid initial Voronoi vertex, the *edge-tracing algorithm* follows Voronoi edges until the construction is completed. On the other hand, the *region-expansion algorithm* constructs the desired diagram by expanding Voronoi regions for one sphere after another via a series of topology operations, starting from the ordinary Voronoi diagram for the centres of the spheres. In the worst-case, the edge-tracing algorithm takes $O(mn)$ time, and the region-expansion algorithm takes $O(n^3 \log n)$ time, where m and n are the numbers of edges and spheres, respectively.

We have also shown how such a Voronoi diagram can be used for solving various important geometric problems in biological systems by illustrating two examples: the computation of surfaces defined on a protein, and the extraction and characterization of interaction interfaces between multiple proteins.

While the presented Voronoi diagram provides precise results compared to the other topological constructs such as a power diagram or an α-hull, further research issues need to be addressed. For example, the robustness and the speed of construction of the diagram require further study.

However, we want to emphasize that the Euclidean Voronoi diagram of spheres in 3D has enormous applications for various disciplines in science and engineering. The Voronoi diagram will, we believe, provide new opportunities and challenges for geometers.

Acknowledgements

This research was supported by the Creative Research Initiatives from the Ministry of Science and Technology in Korea. The authors would like to express their deep appreciation to two anonymous reviewers who have provided constructive suggestions.

References

1. Angelov, B., Sadoc, J.-F., Jullien, R., Soyer, A., Mornon, J.-P., Chomilier, J.: Nonatomic solvent-driven Voronoi tessellation of proteins: an open tool to analyze protein folds. Proteins: Structure, Function, and Genetics **49**(4) (2002) 446–456.
2. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM Journal of Computing **16** (1987) 78–96.
3. Bajaj, C. L., Pascucci, V., Shamir, A., Holt, R. J., Netravali, A. N.: Dynamic maintenance and visualization of molecular surfaces. Discrete Applied Mathematics **127** (2003) 23–51.
4. Blum, H.: A transformation for extracting new descriptors of shape. Proc. Symp. Models for Perception of Speech & Visual Form (1967) 362–380.
5. Blum, H., Nagel, R.N.: Shape description using weighted symmetric axis features. Pattern Recognition **10** (1978) 167–180.
6. Boissonnat, J.D., Karavelas, M.I.: On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. in Proceedings of the 14th annual ACM-SIAM Symposium on Discrete Algorithms (2003) 305–312.
7. Cho, Y., Kim, D., Kim, D.-S.: Topology representation for Euclidean Voronoi diagram of spheres in 3D. in Proc. Digital Engineering Workshop/5th Japan-Korea CAD/CAM Workshop (2005).

8. Connolly, M. L.: Analytical molecular surface calculation. Journal of Applied Crystallography **16** (1983) 548–558.
9. Connolly, M, L.: Solvent-accessible surfaces of proteins and nucleic acids. Science **221** (1983) 709–713.
10. Connolly, M, L.: Molecular surfaces: a review. Network Sci. (1996).
11. Edelsbrunner, H. and Mücke, E.P.: Three-dimensional alpha shapes. ACM Transactions on Graphics **13**(1) (1994) 43–72.
12. Edelsbrunner, H., Facello, M., Liang, J.: On the definition and the construction of pockets in macromolecules. Discrete Applied Mathematics **88** (1998) 83–102.
13. Gavrilova, M.: Proximity and Applications in General Metrics. Ph.D. thesis: The University of Calgary, Dept. of Computer Science, Calgary, AB, Canada (1998).
14. Gavrilova, M., Rokne, J.: Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean *d*-dimensional space. Computer Aided Geometric Design **20**(4) (2003) 231–242.
15. Goede, A., Preissner, R., Frömmel, C.: Voronoi cell: new method for allocation of space among atoms: elimination of avoidable errors in calculation of atomic volume and density. Journal of Computational Chemistry **18**(9) (1997) 1113–1123.
16. Halperin, D., Overmars, M.H.: Spheres, Molecules, and Hidden Surface Removal. Proc. 10th ACM Symposium on Computational Geometry, (1994) 113-122.
17. Held, M.: On the Computational Geometry of Pocket Machining. Lecture Notes in Computer Science **500**, Springer-Verlag (1991).
18. Kim, D.-S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology. Computer Aided Geometric Design **18**(6) (2001) 541–562.
19. Kim, D.-S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. Computer Aided Geometric Design **18**(6) (2001) 563–585.
20. Kim, D.-S., Cho, Y., Kim, D., Cho, C.-H.: Protein structure analysis using Euclidean Voronoi diagram of atoms. Proc. International Workshop on Biometric Technologies (BT 2004) (2004) 125–129.
21. Kim, D.-S., Cho, Y., Kim, D.: Edge-tracing algorithm for Euclidean Voronoi diagram of 3D spheres. Proc. 16th Canadian Conference on Computational Geometry (2004) 176–179.
22. Kim, D,-S., Cho, Y., Kim, D., Kim, S., and Bhak, J.: Euclidean Voronoi Diagram of 3D Spheres and Applications to Protein Structure Analysis. International Symposium on Voronoi Diagrams in Science and Engineering, University of Tokyo, Tokyo, Japan (2004) 13–15.
23. Kim, D.-S., Cho, Y., Kim, D.: Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. Computer-Aided Design (in printing).
24. Kim, D., Kim, D.-S.: Euclidean Voronoi diagrams for spheres in 3D by expanding regions. (in prepration).
25. Kirkpatrick, D.G.: Efficient computation of continuous skeletons. Proc. 14th IEEE Symp. Foundations of Computer Science (1979) 18–27.
26. Lee, B., Richards, F.M.: The interpretation of protein structures: estimation of static accessibility. Journal of Molecular Biology, **55** (1971) 379–400.
27. Lee, D.T.: Medial axis transformation of a planar shape. IEEE Transactions on Pattern Analysis and Machine Intelligence **4** (1982) 363–369.
28. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis: McGraw-Hill, (1982).

29. Luchnikov, V.A., Medvedev, N.N., Oger, L., Troadec, J.-P.: Voronoi-Delaunay analyzis of voids in systems of nonspherical particles. Physical review E **59**(6) (1999) 7205–7212.
30. Montoro, J.C.G., Abascal, J.L.F.: The Voronoi polyhedra as tools for structure determination in simple disordered systems. The Journal of Physical Chemistry **97**(16) (1993) 4211–4215.
31. Naberukhin, Y.I., Voloshin, V.P., Medvedev, N.N.: Geometrical analysis of the structure of simple liquids: percolation approach. Molecular Physics **73** (1991) 917–936.
32. Noggle, J.H.: Physical Chemistry. 3rd Edition: Freedom Academy Publishing Co., (1996).
33. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. 2nd ed. Chichester: John Wiley & Sons, (1999).
34. Peters, K. P., Fauck, J., Frömmel, C.: The automatic search for ligand binding sites in protein of know three-dimensional strucutre using only geometric criteria. Journal of Molecular Biology **256** (1996) 201–213.
35. RCSB Protein Data Bank. http://www.rcsb.org/pdb/, (2004).
36. Richards, F.M.: The interpretation of protein structures: total volume, group volume distributions and packing density. Journal of Molecular Biology **82** (1974) 1–14.
37. Richards, F. M.: Areas, volumes, packing and protein structure. Annu. Rev. Biophys. Bioeng. **6** (1977) 151–176.
38. Sastry, S., Corti, D.S., Debenedetti, P.G., Stillinger, F.H.: Statistical geometry of particle packings. I. Algorithm for exact determination of connectivity, volume, and surface areas of void space in monodisperse and polydisperse sphere packings. Physical Review E **56** (1997) 5524–5532.
39. Thiessen, A.H.: Precipitation averages for large areas. Monthly Weather Review **39** (1911) 1082–1084.
40. Voloshin, V.P., Beaufils, S., Medvedev, N.N.: Void space analysis of the structure of liquids. Journal of Molecular Liquids **96-97** (2002) 101–112.
41. Weiler, K.: The radial edge structure: a topological representation for non-manifold geometric boundary modeling, in: Wozny, M.J., McLaughlin, H.W. and Encarnacao, J.L. Eds., Geometric Modeling for CAD Applications, North Holland, Elsevier Science Publishers B.V., (1988) 3–36.
42. Will, H.-M.: Computation of Additively Weighted Voronoi Cells for Applications in Molecular Biology. Ph.D. thesis, ETH, Zurich (1999).
43. Cambridge Crystallographic Data Centre, http://www.ccdc.cam.ac.uk/ , (2005).

The Importance of Polynomial Reproduction in Piecewise-Uniform Subdivision

Adi Levin

Cadent Ltd, Hata'asiya 17, Or-Yehuda, Israel
adi.levin@cadent.co.il

Abstract. We survey a number of related methods, which have been published by the author and collaborators, in the field of subdivision schemes for curves and surfaces. The theory presented in these works relies mainly on the notion of *polynomial reproduction*, i.e. the ability of a scheme to reproduce all polynomials up to a certain degree as limit functions. We demonstrate that the study of polynomial reproduction is central to smoothness analysis and to approximation. In particular, we show how to exploit polynomial reproduction in the context of piecewise-uniform stationary subdivision. The applications include boundary treatments for subdivision surfaces, interpolation of curves by surfaces, subdivision stencils around extraordinary vertices (construction of C^2 schemes), as well as schemes that involve different kinds of grids (triangular / quadrilateral).

1 Introduction

A subdivision scheme works by applying a refinement operator to given control points, repeatedly. The control points at increasing refinement levels converge to a function called the *limit function*. If the refinement rule is the same for all levels, this is a *stationary subdivision scheme*.

Subdivision schemes are especially useful for representing smooth surfaces, because they can operate over meshes of arbitrary topology, in contrast to parametric surface representations, which are limited to shapes homeomorphic to a subset of the plane.

Typically, a subdivision scheme consists of a small number of simple refinement rules. The meshes at increasing refinement levels have a regular structure, with the exception of so-called *extraordinary vertices*. In the regular areas of the mesh, the surface is governed by a *uniform subdivision scheme*. The special rules that apply in the vicinity of the extraordinary vertices are chosen with the purpose of maximizing the smoothness of the limit surfaces there.

The mathematical study of any system of surface representations investigates the *smoothness* of the resulting surfaces, as well as their *approximation* properties, i.e. the ability to efficiently reproduce a given shape, with desired accuracy. In the case of uniform subdivision, the notion of *polynomial reproduction* comes up as a key property, both in the conditions for good approximation properties, and in the conditions for smoothness.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 272–307, 2005.

It is known that any uniform subdivision scheme which generates C^m limit functions, is always capable of generating all of the polynomials up to degree m as limit functions [1]. Also, if a scheme generates all of the polynomials up to degree m, then it has approximation order $m+1$ (this term is explained in § 2.4, in the context of uniform subdivision).

This paper introduces *piecewise-uniform subdivision schemes*, which are subdivision schemes that apply different refinement rules at different regions of the mesh. These include boundary treatments, rules near extraordinary vertices, and subdivision rules along special curves on the surface. Our approach for constructing and for analyzing such schemes is built around the property of polynomial reproduction.

We start by a study of the polynomial reproduction properties of uniform subdivision. We ask, for a given uniform subdivision scheme, what polynomials can be generated as limit functions of that scheme, and how to calculate the initial control points that produce these polynomials in the limit. In our notation, we have a subdivision operator S, and we want to calculate an operator Q that maps polynomials up to degree m to sets of control points, with the property $S^\infty Qf = f, \quad \forall f \in \pi_m$. Given a polynomial f, Qf are the control points that produce f as in limit.

For interpolatory schemes, the operator Q is simply a restriction to the integer grid. Non-trivial formulae, such as $Qf(i) = f(i) - \frac{1}{6}f''(i)$, arise in the context of non-interpolatory schemes (see [2]).

Once the operator Q is identified for the uniform parts of our subdivision scheme, we use it in order to extend our scheme to the boundaries between regions and to extraordinary points.

The key theorem in the paper states, that for stationary subdivision schemes S,

$$S^\infty Q = id \iff SQ = Q\sigma, \tag{1}$$

over the space of polynomials up to degree m, where σ is the dilation operator $\sigma f = f\left(\frac{\cdot}{2}\right)$. This result was first presented in [3] for the case of *uniform subdivision.* In [2] the result was extended to stationary non-uniform subdivision.

The significance of (1), is that it reduces the equation $S^\infty Q = id$, which is a complicated equation to solve, if S is the unknown, to the equation $SQ = Q\sigma$, which is linear in S. This enables us to construct new subdivision schemes, in which the weights are calculated by solving small systems of linear equations that arise from the requirement $SQ = Q\sigma$.

We also use the operator Q in a method of approximation called *quasi-interpolation.* We show that for any uniform subdivision scheme S we can calculate Q and then extend it to operate on non-polynomial functions. This extension of Q is, in fact, an approximation scheme, i.e. a method of selecting control points such that limit function approximates a desired function.

The introduction of the operator Q in [2] serves as the basis for the work of Levin and Levin in [4], in which a new smoothness analysis tool is introduced. The general setting which is introduced there, called *quasi-uniform subdivision*, stands for a subdivision scheme which is uniform everywhere except for the neighborhood of an *extraordinary line*. This is a situations that appears naturally in piecewise-uniform subdivision schemes. In particular, when subdividing a mesh which consists of both triangles and quadrilaterals, the subdivision scheme near the boundary between triangles and quadrilaterals is quasi-uniform [5, 6]. We demonstrate our method by constructing a C^2 bivariate tri-quad scheme. Recently, Hakenberg [7] used the same methodology to construct a volumetric subdivision scheme over unstructured three-dimensional meshes that consist of cubes, triangular prisms, tetrahedra and octahedra. The limit surfaces of this scheme are C^2-continuous almost everywhere, and they reproduce all trivariate quadratic polynomials.

The above methodology, together with the analysis tool of quasi-uniform subdivision was used by Zulti et al. [8] in the construction of a subdivision scheme that generates C^2-continuous limit functions around an extraordinary vertex. In the setting of piecewise-uniform subdivision, defined over a parameterized grid, the requirement for reproduction of quadratic polynomials (which is a key property for C^2 continuity), is reduced to a finite system of linear equations, which can always be solved by large enough stencils. This is in contrast to the standard setting of subdivision, in which quadratic reproduction over the characteristic map seems to be especially difficult [9, 10, 11].

Similar methods were used in [3] for the construction and analysis of subdivision schemes that satisfy transfinite boundary conditions. Such schemes are given as input a set of control points and a function that describes the boundary conditions. By extending the identity (1) to this context, one can reduce the requirement of polynomial reproduction into a set of linear equations in which the unknowns are the subdivision weights. It is shown in [3] that if a scheme of this type reproduces polynomials up to degree m, then the C^m continuity of its limit function for any C^m boundary condition, follows from the C^m continuity of the limit function for zero boundary conditions. This powerful theorem enables to construct subdivision schemes for interpolation of curves by surfaces [12], for hole filling [13], and for other applications [3].

The paper is structured as follows. Section 2, taken from [2] with the proofs omitted, introduces polynomial reproduction in the context of uniform subdivision. Section 3, also taken from [2], introduces non-uniform stationary subdivision, and studies polynomial reproduction in that context. Section 4 summarizes the smoothness analysis method originally published in [4], including the C^2 tri-quad scheme. In section 5 we describe the construction of a C^2 subdivision scheme around an extraordinary vertex, originally published in [8]. Section 6 is an introduction to the theory of subdivision schemes that satisfy transfinite boundary conditions, originally presented in [3]. In particular, we demonstrate a modification of Catmull-Clark for satisfying C^1 boundary conditions.

2 Uniform Subdivision

In this section we use results from the theory of stationary subdivision and the theory of shift-invariant spaces. We introduce the operator Q, that leads to quasi-interpolation and to a condition for polynomial generation. This was originally presented in [3].

The study of the operator Q for uniform subdivision becomes useful when we proceed to construct *piecewise-uniform* subdivision schemes with prescribed approximation orders, as demonstrated in § 3.5.

2.1 Notations

Let $l = l(\mathbb{Z}^s)$ denote the collection of all sequences $P : \mathbb{Z}^s \to \mathbb{R}$. We refer to $P \in l$ as a set of *control points*. Let $C = C(\mathbb{R}^s)$ denote the space of continuous real functions on $\mathbb{R}^s$. Similarly, C^m stands for the spaces of m-times differentiable functions over $\mathbb{R}^s$ whose m-th order derivatives are continuous.

We use the standard multi-index notations for $\mathbb{Z}^s$, $j = (j_1, \ldots, j_s) \in \mathbb{Z}^s$, $j \geq 0$ if $j_1, \ldots, j_s \geq 0$, $|j| = j_1 + \ldots + j_s$, $x^j = x_1^{j_1} \cdot \ldots \cdot x_s^{j_s}$. For $j \geq 0$ we use $j! = j_1! \cdot \ldots \cdot j_s!$, $D^j = \frac{\partial^{|j|}}{\partial^{j_1} x_1 \cdots \partial^{j_s} x_s}$. The entries of a multi-index sequence $P \in l$ are denoted by $P(j)$, for $j \in \mathbb{Z}^s$.

The following notations are used for arithmetic operations on subsets of $\mathbb{R}^s$: Let $X, Y \subset \mathbb{R}^s$, $z \in \mathbb{R}^s$ and $\alpha \in \mathbb{R}$. Then

$$\begin{aligned} X + Y &= \{x + y \mid x \in X,\ y \in Y\}, \\ X + z &= \{x + z \mid x \in X\}, \\ \alpha X &= \{\alpha x \mid x \in X\}, \end{aligned}$$

For $k \geq 0$, the space of polynomials of degree at most k over $\mathbb{R}^s$, is denoted by $\pi_k = \pi_k(\mathbb{R}^s)$. Its restriction to the integer grid is denoted by $\pi_k(\mathbb{Z}^s)$.

Throughout the paper, we use the maximum norm in $\mathbb{R}^s$,

$$\|x\| = \max_{i=1,\ldots,s} |x_i|, \quad \forall x \in \mathbb{R}^s.$$

We denote by σ the dilation operator on C, which appears naturally in the context of stationary subdivision,

$$\sigma f = f\left(\frac{\cdot}{2}\right), \quad f \in C.$$

Uniform Subdivision Schemes

A uniform subdivision operator is a linear operator $S : l \to l$ which is defined by a finitely supported mask $a \in l$ through

$$(SP)(\alpha) = \sum_{\beta \in \mathbb{Z}^s} a(\alpha - 2\beta) P(\beta), \quad \forall \alpha \in \mathbb{Z}^s. \tag{2}$$

The repeated application of S to a given set of control points P, $\{S^n P\}_{n=0,\ldots,\infty}$, is called a *stationary subdivision scheme.*

A subdivision scheme S is termed *uniformly convergent*, if for every $P \in l$, there exists $F \in C$ (called the *limit function*) such that

$$\lim_{n \to \infty} \left\| S^n P - F\left(2^{-n} \cdot\right) \right\|_{\infty, \mathbb{Z}^s \cap 2^n D} = 0, \tag{3}$$

for any open and bounded domain $D \subset I\!R^s$. The introduction of the bounded domain D here is needed if we do not want to exclude unbounded sequences of control points and unbounded limit functions. In this paper we are specifically interested in the case where the limit function is a polynomial, and therefore, not bounded. We denote the limit function F, by $S^\infty P$.

S is called an interpolatory subdivision scheme, if $SP(2\alpha) = P(\alpha) \quad \forall \alpha \in \mathbb{Z}^s$.

We say that S belongs to the class C^m if S is uniformly convergent, and $S^\infty P \in C^m$ for every $P \in l$. For the smoothness analysis of uniform subdivision schemes the reader is referred to [1, 14]. For a uniformly convergent S we define the S-refinable function $\Phi = S^\infty \delta$, where $\delta \in l$ is the sequence which is 1 at the origin and zero anywhere else. The limit function $S^\infty P$ can be expressed as a sum of integer translates of Φ

$$S^\infty P = \sum_{\alpha \in \mathbb{Z}^s} P(\alpha) \Phi(\cdot - \alpha). \tag{4}$$

2.2 The Operator Q

In this section we discuss the known fact, that uniform subdivision schemes, under certain conditions, can generate polynomials as their limit functions. Our goal in the following lemmas is to establish the logical relation (1).

It is shown in [1], theorem 8.4 (see also [15]) that if $S \in C^m$ and the integer translates of Φ are linearly independent, then the space $\pi_m(\mathbb{Z}^s)$ is invariant under S. For many uniform subdivision schemes S, the space $\pi_m(\mathbb{Z}^s)$ is invariant under S even though S does not belong to C^m. Cavaretta et al. ([1], chapter 6) formulate conditions on the mask a that can be used to find the maximal m such that $\pi_m(\mathbb{Z}^s)$ is invariant under S (If the integer shifts of Φ are linearly independent, this amounts to simple algebraic conditions on a, by corollary 6.3 and theorem 6.3 in [1]). In the following results we study the mapping $S : \pi_m(\mathbb{Z}^s) \to \pi_m(\mathbb{Z}^s)$.

Lemma 1 (Proven in [2]). *If S maps $\pi_m(\mathbb{Z}^s)$ to itself, and S is convergent, then S^∞ maps $\pi_m(\mathbb{Z}^s)$ to $\pi_m(I\!R^s)$, and the leading coefficients of $S^\infty p$ are the same as those of p, for any $p \in \pi_m(\mathbb{Z}^s)$.*

In the following, we show that S is similar to σ on $\pi_m(\mathbb{Z}^s)$. The similarity operator Q is then shown to be the inverse of S^∞ on π_m. A similar result also appeared in the context of wavelet theory in [16] (Lemma 3.2.3).

Lemma 2 (Proven in [2]). *If S^∞ is a 1-1 map from $\pi_m(\mathbb{Z}^s)$ to $\pi_m(I\!R^s)$, then the restriction of S to $\pi_m(\mathbb{Z}^s)$ is similar to the dilation operator,*

$$SQ = Q\sigma, \quad \text{on } \pi_m(I\!R^s), \tag{5}$$

and the similarity operator $Q : \pi_m(I\!R^s) \to \pi_m(\mathbb{Z}^s)$ is the inverse of S^∞ on $\pi_m(I\!R^s)$. Moreover, the leading coefficients of Qf are the same as those of f, for all $f \in \pi_m(I\!R^s)$.

Note that the lemma only requires that S^∞ is 1-1 over $\pi_m(\mathbb{Z}^s)$ and not for all sequences of control points. Since this is a finite-dimensional space, this property is not difficult to check. In particular, it holds for any scheme S which satisfies the conditions of Lemma (1).

Lemma 3. *If $Q : \pi_m(\mathbb{R}^s) \to \pi_m(\mathbb{Z}^s)$ satisfies*

$$SQ = Q\sigma, \quad \text{on } \pi_m(\mathbb{R}^s),$$

and the leading coefficients of Qf are the leading coefficients of f for all $f \in \pi_m(\mathbb{R}^s)$, then $S^\infty Q|_{\pi_m(\mathbb{R}^s)} = id$, namely

$$S^\infty Qf = f, \quad \forall f \in \pi_m(\mathbb{R}^s).$$

By Lemmas 2 and 3:

Corollary 1. *If S is convergent, and $Q : \pi_m(\mathbb{R}^s) \to \pi_m(\mathbb{Z}^s)$ preserves leading coefficients then*

$$S^\infty Q = id \iff SQ = Q\sigma,$$

both identities restricted to $\pi_m(\mathbb{R}^s)$.

The significance of Corollary (1) is the reduction of the property $S^\infty Q = id$, which is the formal notation for polynomial generation, to the relation $SQ = Q\sigma$, in which S appears as a linear term. This is useful for constructing new subdivision schemes, in which case S is the unknown. It enables us to derive the weights of S from the solution of a system of linear equations.

2.3 Polynomial Eigenvectors of S

From the relation $SQ = Q\sigma$, over the polynomials of degree up to m, we can easily derive the most significant eigenvectors of the operator S. These are the eigenvectors that produce polynomials in the limit. For $k = (k_1, \ldots, k_s) \in \mathbb{Z}^s_+$, we take the monomial

$$f(x) = x^k, \quad x \in \mathbb{R}^s.$$

Then it follows that

$$SQf = Q\sigma f = Q2^{-|k|}f = 2^{-|k|}Qf.$$

Hence, Qf is an eigenvector of S with the eigenvalue $2^{-|k|}$. Moreover, we know that Qf is a polynomial of degree $|k|$, provided that Q preserves leading coefficients.

We also know the limit function that corresponds to this eigenvector. It is simply the monomial x^k, because

$$S^\infty Qf = f.$$

For example, when S is the well-known cubic B-spline subdivision scheme, its corresponding Q operator is

$$Qf(i) = f(i) - \tfrac{1}{6}f''(i), \quad \forall i \in \mathbb{Z}^s, \quad f \in \pi_3(\mathbb{R}). \tag{6}$$

This is shown in detail in [2]. We can immediately write down the four eigenvectors of S with eigenvalues $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$. These are:

$$\begin{aligned} Q(1) &= (1) = (\ldots, 1, 1, 1, 1, 1, \ldots) \\ Q(x) &= (x) = (\ldots, -2, -1, 0, 1, 2, \ldots) \\ Q(x^2) &= (x^2 - \tfrac{1}{3}) = (\ldots, 3\tfrac{2}{3}, \tfrac{2}{3}, -\tfrac{1}{3}, \tfrac{2}{3}, 3\tfrac{2}{3}, \ldots) \\ Q(x^3) &= (x^3 - x) = (\ldots, -7, 0, 0, 0, 7, \ldots), \end{aligned}$$

using the notation $(f) = f|_{\mathbb{Z}^s}$.

2.4 Quasi-Interpolation

The operator Q, introduced in section § 2.2 is defined as an operator from $\pi_m(\mathbb{R}^s)$ to $\pi_m(\mathbb{Z}^s)$. In this section, we show how Q can be extended to an operator from $C(\mathbb{R}^s)$ to $l(\mathbb{Z}^s)$ which is local and bounded (in the sense of Lemma (5)). We then use Q to define a quasi-interpolation operator, and we prove an approximation order result.

We assume that for a given convergent subdivision scheme S, we have that $S^\infty Q f = f$ for all $f \in \pi_m(\mathbb{R}^s)$. Since S^∞ is shift invariant over π_m, it follows that Q is shift invariant as well. The following lemma uses this fact to show that Q can be represented as a sum of differential operators, when viewing Q as an operator on the space of polynomials up to degree m. For the proofs for the following lemmas, the reader is referred to [2].

Lemma 4. *Q can be expressed in the form*

$$Q = \left. \sum_{|i| \le r} a_i D^i \right|_{\mathbb{Z}^s},$$

taking functions in $\pi_m(\mathbb{R}^s)$ to sequences in $\pi_m(\mathbb{Z}^s)$.

Lemma 5 (An extension of Q). *Q can be extended to a shift invariant operator $Q : C(\mathbb{R}^s) \to l(\mathbb{Z}^s)$ which is bounded and local, namely, there exists an open and bounded domain $A \in \mathbb{R}^s$ and $c > 0$ such that*

$$|Qf(\alpha)| \le c\|f\|_{\infty, \alpha + A}, \quad \forall \alpha \in \mathbb{Z}^s, \quad \forall f \in C(\mathbb{R}^s).$$

As an example for such an extension, we can extend Q, originally defined only for cubics, by (6), to all continuous functions, by

$$Qf(i) = \tfrac{4}{3} f(i) - \tfrac{1}{6} f(i-1) - \tfrac{1}{6} f(i+1), \quad \forall i \in \mathbb{Z}^s, \quad f \in C(\mathbb{R}). \tag{7}$$

The expressions in (6) and (7) coincide when f is a cubic polynomial.

Using the extension of Q to $C(\mathbb{R}^s)$, we can approximate any continuous function f by $S^\infty Q f$. This method is known as *quasi-interpolation*. The following theorem can also be derived as a particular case from the general theory of PSI spaces, and quasi-interpolation (see e.g. [17, 18]).

Theorem 1 (Quasi-interpolation). *Let S denote a uniformly convergent subdivision scheme, and let $Q : C(I\!R^s) \to l(Z\!\!Z^s)$ be a local and bounded operator (in the sense of lemma 5), such that*

$$S^\infty Qp = p, \quad p \in \pi_m(I\!R^s).$$

Then there exists $C > 0$ such that

$$\|S^\infty Qf - f\|_\infty \leq C\,\|f\|_{m+1}\,,$$

for every $f \in C^{m+1}(I\!R^s)$ with $\|f\|_{m+1} < \infty$, where $\|f\|_{m+1}$ denotes the supremum of the partial derivatives of f of order $m + 1$.

Approximation Order

This leads to an estimate of the decay of approximation error, when considering finer and finer approximations of a given function f, using the quasi-interpolation operator $S^\infty Q$. The way we refine the approximation, is to apply it to the dilated version of f, σf. We then dilate the approximation back in order to compare it to the original function f. Therefore, we study the approximation error $\sigma^{-n}S^\infty Q\sigma^n f - f$. From theorem 1 we get

$$\begin{aligned}\left\|\sigma^{-n}S^\infty Q\sigma^n f - f\right\|_\infty &= \|S^\infty Q\sigma^n f - \sigma^n f\|_\infty \leq C\,\|\sigma^n f\|_{m+1} \\ &= 2^{-n(m+1)}C\,\|f\|_{m+1}\,.\end{aligned}$$

Due to the coefficient $2^{-n(m+1)}$ in the error estimate, we say that this quasi-interpolation scheme has an *order of approximation* $m + 1$.

3 Non-uniform Subdivision

In § 2 we restricted ourselves to subdivision schemes that are stationary and uniform. Namely, the same subdivision mask is applied everywhere on the integer grid and at every level of refinement. They were also defined as operators on the integer grid, $l(Z\!\!Z^s)$, which is uniformly distributed over the space $I\!R^s$.

In this section, we extend the results of § 2 to schemes that are stationary but non-uniform. The scheme is still stationary in the sense that a single operator S operates at every level of refinement. However, S is no longer restricted to the form (2). It can no longer be captured by a single mask a that operates everywhere. Also, S doesn't necessarily operate on the integer grid $Z\!\!Z^s$. The grid itself can be non-uniform.

The theoretical results developed in this section describe the relationships between S and the operator Q that are necessary and sufficient for establishing the polynomial generation properties of a subdivision scheme. The practical purpose for this analysis is to introduce a method for constructing new non-uniform subdivision schemes with high approximation orders. Due to the constructive nature of our approach, we do not deal with the question of the existence of Q for given S.

In § 3.1 we extend the notations of stationary subdivision to the non-uniform case. In § 3.2 we discuss the kinds of schemes for which this theory is applicable. In sections 3.3–3.4 we extend the results of § 2 to the non-uniform case. In § 3.5 we demonstrate the construction of a non-uniform scheme with maximal approximation order. This is a new univariate scheme which is interpolatory on the negative side of the real line, and non-interpolatory on the positive side. We show how the relation (1) is used as the basis for constructing the scheme.

3.1 Notations

The integer grid, that was used for defining uniform subdivision, is replaced here by a set of points $X \subset I\!R^s$. A set of control points is a sequence of values on X, $P \in l(X)$. The subdivision operator S is a linear operator $S : l(X) \to l(X)$. A stationary subdivision scheme is defined as the repeated application of S to given control points $P \in l(X)$.

We say that S is *uniformly convergent*, if for every $P \in l(X)$, there exists $F \in C(I\!R^s)$ (called the limit function) such that

$$\lim_{n\to\infty} \left\| S^n P - F\left(2^{-n}\cdot\right)\right\|_{\infty, X\cap 2^n D} = 0, \tag{8}$$

for any open and bounded domain $D \subset I\!R^s$. We denote $S^\infty P = F$. We also require, as part of the definition of uniform convergence, that $S^\infty P$ is non-zero for some P. We restrict our attention to grids X such that $2X \subset X$. This enables us to define interpolatory subdivision schemes S, as schemes where $SP(2x) = P(x)$ for all $x \in X$, and for all $P \in l(X)$. We also require that the dilations of X are dense in $I\!R^s$, namely

$$\overline{\bigcup_{n=0}^{\infty} 2^{-n} X} = I\!R^s.$$

This guarantees that if the limit function (8) exists, it is unique.

In contrast to uniform subdivision, $S^\infty P$ can no longer be expressed as a sum of integer translates of one compactly supported refinable basis function, as in (4). However, we are still interested in the notion of locality of the subdivision scheme. We say that S is local with support $\Omega \subset I\!R^s$, or that S^∞ is supported on Ω, if for all $x \in I\!R^s$, and for all $P \in l(X)$,

$$P|_{(x+\Omega)\cap X} = 0 \Rightarrow S^\infty P(x) = 0. \tag{9}$$

3.2 Kinds of Non-uniform Schemes

A non-uniform scheme can be defined on a non-uniform grid X. Several examples of non-uniform grids in $I\!R^1$ and in $I\!R^2$ are shown in Fig. 1. They all satisfy the relation $2X \subset X$, and $2^{-n}X$ converges to a set which is dense in $I\!R^1$ and $I\!R^2$ respectively. A particular extension of the four-point scheme to a semi-uniform grid such as in Fig. 1(a) was analyzed in [3]. Subdivision schemes for the grid in Fig. 1(b) have not yet been studied. The tri-quad grid in Fig. 1(c) is relevant

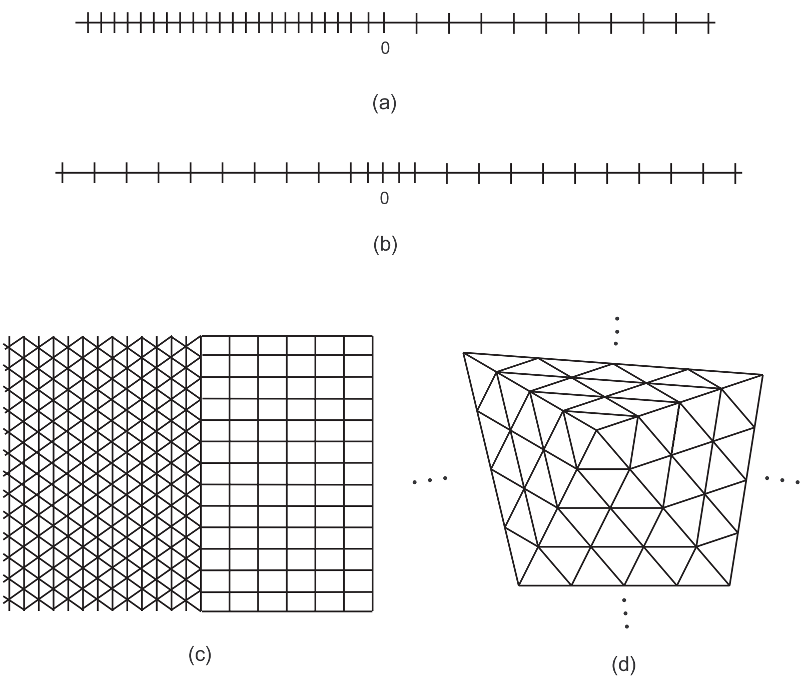


Fig. 1. Non-uniform grids. (a) Different grid spacing from both sides of the origin. (b) Higher density near the origin. (c) A grid that combines quadrilaterals on one side of the axis with triangles on the other. (d) A k-regular grid. The only vertex that has valency other than 6 is at the origin

to applications in which we subdivide meshes that consist of both triangles and quadrilaterals. A grid such as the k-regular grid in Fig. 1(d) appears naturally around extraordinary vertices in subdivision meshes.

A subdivision scheme defined on a uniform grid may also be non-uniform, if it operates differently on different areas of the grid. A class of univariate schemes operating differently on either side of the origin was analyzed in [15]. In the bivariate case, such non-uniformity is used in crease-rules for subdivision surfaces (see e.g. [19, 20]). These are similar to boundary rules that operate near boundaries of meshes [21]. The setting where the grid is uniform, but the rules change near the boundary is the one used in *combined subdivision schemes*, [22, 13, 12, 3], which are subdivision schemes that satisfy boundary conditions.

The kinds of non-uniformity that we consider in this paper do not include schemes that have no regular structure in them, such as the univariate schemes described in [23, 24], that can add samples not only at the middle of intervals, and operate differently at each refinement level.

3.3 The Operator Q

In this section we show that (1) is valid in the non-uniform case, under certain assumptions. The lemmas are proven in [2].

We observe a significant difference between the uniform and the non-uniform case, as far as polynomial generation is concerned. In both cases, we ask how to choose the control points so that we get a polynomial in the limit. The answer comes in the form of an operator Q, because $S^\infty Q$ is the identity on a subspace of polynomials. But, in the uniform case, in order to get a polynomial in the limit, we had to take control points that lie on a polynomial, whereas in the non-uniform case this is no longer true.

The operator Q is no longer an operator from $\pi_m(\mathbb{R}^s)$ to $\pi_m(X)$, and we consider a more general

$$Q : \pi_m(\mathbb{R}^s) \to l(X).$$

However, we will still need a notion parallel to preservation of leading coefficients, which is one of the conditions of Lemma 3. We say that $Q : \pi_m(\mathbb{R}^s) \to l(X)$ preserves leading coefficients, if

$$f \in \pi_k(\mathbb{R}^s) \Rightarrow |Qf(x) - f(x)| = o(\|x\|^k), \quad \text{as } \|x\| \to \infty, \tag{10}$$

for all $k \leq m$.

The following propositions establish (1) in the non-uniform case.

Lemma 6. *Let $S : l(X) \to l(X)$ be a convergent subdivision operator. If S^∞ is an injection, namely, $S^\infty P = 0 \Rightarrow P = 0$, and $Q : \pi_m(\mathbb{R}^s) \to l(X)$ is such that*

$$S^\infty Q f = f, \quad \forall f \in \pi_m(\mathbb{R}^s), \tag{11}$$

then

$$SQf = Q\sigma f, \quad \forall f \in \pi_m(\mathbb{R}^s). \tag{12}$$

Lemma 7. *Let $S : l(X) \to l(X)$ be a convergent subdivision operator, and let Q denote a linear operator $Q : \pi_m(\mathbb{R}^s) \to l(X)$. If*

$$SQf = Q\sigma f, \quad \forall f \in \pi_m(\mathbb{R}^s), \tag{13}$$

and Q preserves leading coefficients, in the sense of (10), then

$$S^\infty Q f = f, \quad \forall f \in \pi_m(\mathbb{R}^s), \tag{14}$$

Corollary 2. *If S is convergent, S^∞ is an injection, and $Q : \pi_m(\mathbb{R}^s) \to l(X)$ preserves leading coefficients in the sense of (10), then*

$$S^\infty Q = id \iff SQ = Q\sigma,$$

both identities restricted to $\pi_m(\mathbb{R}^s)$.

The observations in § 2.3 are also easily extended to the non-uniform case. Provided that Q satisfies $SQ = Q\sigma$ over the polynomials up to degree m, then Qf is an eigenvector of S with eigenvalue $2^{-|k|}$, if f is the monomial $f(x) = x^k$, for $|k| \leq m$. The difference from the uniform case is, that now the values of the eigenvector Qf do not necessarily lie on a polynomial. The corresponding limit function is still a polynomial, however, because $S^\infty Qf = f$.

3.4 Quasi-Interpolation

In § 2.4 we have shown that if $Q : \pi_m(\mathbb{R}^s) \to \pi_m(\mathbb{Z}^s)$ satisfies

$$S^\infty Q f = f, \quad \forall f \in \pi_m(\mathbb{R}^s),$$

then there is a bounded and local extension to Q as an operator $Q : C(\mathbb{R}^s) \to l(\mathbb{Z}^s)$, and this extension is used in an approximation scheme that has approximation order $m+1$ (Theorem 1).

For the extension of Q to non-polynomial functions in § 2.4, we used the fact that Q was shift-invariant. In the non-uniform case, Q is not necessarily shift-invariant. Therefore, we will not describe how Q can be extended. Instead, we will assume here that Q can be extended to an operator $Q : C(\mathbb{R}^s) \to l(X)$, which is bounded and local, in the sense that there exists an open and bounded domain $A \in \mathbb{R}^s$ and $c > 0$ such that

$$|Qf(\alpha)| \le c\|f\|_{\infty,\alpha+A}, \quad \forall \alpha \in X, \forall f \in C(\mathbb{R}^s). \tag{15}$$

Theorem 1 extends easily to the non-uniform case, with a similar proof, provided that we assume that S^∞ is local and bounded, in the following sense: We say that S is local, if (9) is satisfied with a support Ω which is bounded. We require, in addition, that there exists $c > 0$ such that

$$|S^\infty P(x)| \le c\,\|P\|_{(x+\Omega)\cap X}\,, \quad \forall P \in l(X), \quad \forall x \in \mathbb{R}^s.$$

In the uniform case, this bound on $S^\infty P$ follows easily from the convergence of S. In the non-uniform case, this is not necessarily true. However, since this in itself is a fundamental property that is desirable to have in S, it doesn't add any new restrictions.

The approximation order $m+1$ for the function values, and $m+1-|j|$ for derivatives of order j, follows easily, just as in the uniform case.

3.5 Example

In this section, we use the above theory to construct a non-uniform univariate scheme, that is interpolatory on the left side of the real line, and non-interpolatory on the right side.

The four point interpolatory scheme [25] is a family of interpolatory schemes, given by the mask $a = [-w, 0, \frac{1}{2}+w, 1, \frac{1}{2}+w, 0, -w]$, supported on $\{-3,\dots,3\}$, where w is a tension parameter. It is shown in [14] (see also [25]) that the limit function Φ belongs to C^1 when w is in the range $0 < w < \frac{-1+\sqrt{5}}{8}$. Moreover, when w is in this range, Φ has Hölder continuous first derivatives, of order $1-\epsilon$, for all $0 < \epsilon < 1$. In this section we restrict our attention to the case $w = \frac{1}{16}$ where the four-point scheme reproduces cubic polynomials (see also [26]). Since the scheme is interpolatory, the corresponding operator Q is the identity.

We now proceed to combine the four-point scheme with the cubic B-spline scheme. Using the notations of § 3.1, the grid X is the set of integers $X = \mathbb{Z}$, and our goal is to construct $S : l(\mathbb{Z}) \to l(\mathbb{Z})$ that generates all cubic polynomials

$\pi_3(I\!R)$, such that it is interpolatory on one half and non-interpolatory on the other half of the real line.

From Corollary 2 we know that it is sufficient to show that for some $Q : \pi_3(I\!R) \to l(\mathbb{Z})$,

$$SQ = Q\sigma. \tag{16}$$

So, in order to generate S we also need to determine the appropriate Q. Clearly, there is no unique way to satisfy this equation. We will look for the scheme S that differs from the cubic B-spline scheme or from the four-point scheme at the minimal number of points, and whose support is minimal.

With these guidelines, we first decide on Q, and then solve the equation for S. We define $Q : \pi_3(I\!R) \to l(\mathbb{Z})$ as follows:

$$Qf(i) = \begin{cases} f(i) & i \leq 0 \\ f(i) - \frac{1}{6}f''(i) & i > 0 \end{cases}, \quad \forall f \in \pi_3(I\!R), \quad \forall i \in \mathbb{Z}.$$

The left side of Q is the identity, which coincides with the Q operator for the interpolatory four-point scheme. The right side of Q coincides with the Q operator for the cubic B-spline scheme (see [2]).

For given $P \in l(\mathbb{Z})$ we define $SP(i)$ for $i = 3, 4, \ldots$ by the cubic B-spline scheme, and for $i = 0, -2, -3, -4, \ldots$ by the four point scheme with tension parameter $w = \frac{1}{16}$. It is then easy to see that $SQf(i) = Q\sigma f(i)$ for all cubics f and for $i \in \mathbb{Z}$ except for $i = -1, 1, 2$. We now need to define $SP(-1)$, $SP(1)$ and $SP(2)$ for arbitrary P, such that S satisfies (16).

We look for subdivision rules near the origin that have as small support as possible. The weights of the stencils will be determined by the linear equation (16). We look for S of the form:

$$\begin{aligned} SP(-1) &= a_0P(-2) + a_1P(-1) + a_2P(0) + a_3P(1) \\ SP(1) &= b_0P(-1) + b_1P(0) + b_2P(1) + b_3P(2) \\ SP(2) &= c_0P(-1) + c_1P(0) + c_2P(1) + c_3P(2) \end{aligned} \tag{17}$$

The four variables $a_0, \ldots, a_3$ are determined by the condition $SQf(-1) = Q\sigma f(-1), \quad \forall f \in \pi_3(I\!R)$. By substituting in f, the monomials up to degree 3, we get four equations:

$$\begin{aligned} 1 :& \quad & a_0 + a_1 + a_2 + a_3 &= 1 \\ x :& \quad & -2a_0 - a_1 + a_3 &= -\tfrac{1}{2} \\ x^2 :& \quad & 4a_0 + a_1 + \tfrac{2}{3}a_3 &= \tfrac{1}{4} \\ x^3 :& \quad & -8a_0 - a_1 &= -\tfrac{1}{8} \end{aligned}$$

Similarly, from the conditions $SQf(1) = Q\sigma f(1)$ and $SQf(2) = Q\sigma f(2)$ for all cubics f, we get the following equations for the rest of the subdivision mask:

$$\begin{array}{rrr} 1: & b_0+b_1+b_2+b_3=1 & c_0+c_1+c_2+c_3=1 \\ x: & -b_0+b_2+2b_3=\frac{1}{2} & -c_0+c_2+2c_3=1 \\ x^2: & b_0+\frac{2}{3}b_2+\frac{11}{3}b_3=\frac{1}{6} & c_0+\frac{2}{3}c_2+\frac{11}{3}b_3=\frac{11}{12} \\ x^3: & -b_0+6b_3=0 & -c_0+6c_3=\frac{3}{4} \end{array}$$

All of these systems have unique solutions, given by $a = (-\frac{3}{64}, \frac{1}{2}, \frac{41}{64}, -\frac{3}{32})$, $b = (-\frac{3}{37}, \frac{24}{37}, \frac{33}{74}, -\frac{1}{74})$, $c = (-\frac{3}{148}, \frac{6}{37}, \frac{109}{148}, \frac{9}{74})$. The rules (17), together with the four-point scheme on the left and the cubic B-spline scheme on the right, form the new scheme S. All of stencils used in S are depicted in Fig. 2.

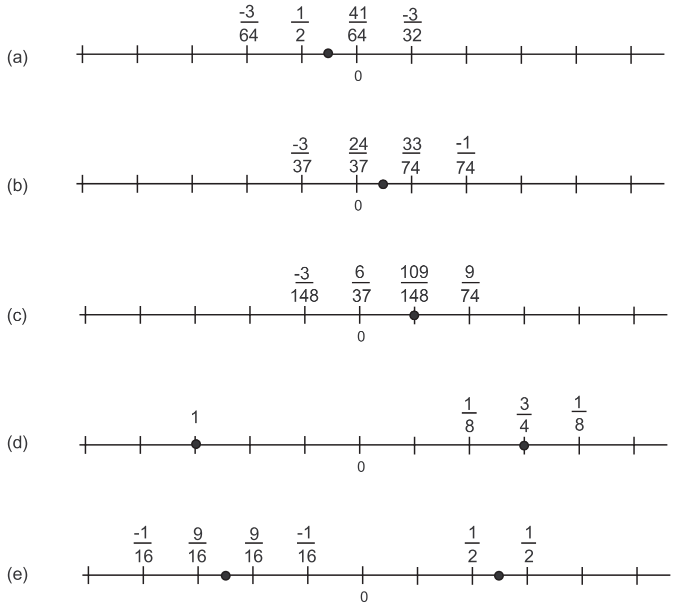


Fig. 2. A non-uniform subdivision scheme combining the four-point scheme on the left side of the real line with the cubic B-spline scheme on the right side. (a) The stencil for $SP(-1)$. (b) The stencil for $SP(1)$. (c) The stencil for $SP(2)$. (d-e) The regular four-point and cubic B-spline stencils

We can easily show that the limit functions of S are C^1-continuous. Clearly, they are C^1-continuous away from the origin, because both cubic B-splines and the limit function of the four-point scheme are C^1-continuous. For the smoothness analysis near the origin it is sufficient to show that the 5*5 local subdivision matrix mapping $P(-2), \ldots, P(2)$ to $SP(-2), \ldots, SP(2)$ has eigenvalues $1, \frac{1}{2}, \lambda_1, \lambda_2, \lambda_3$, with $|\lambda_i| < \frac{1}{2}$. In fact, we found that $\lambda_1 = \frac{1}{4}$, $\lambda_2 = \frac{57}{296} = 0.1926\ldots$, $\lambda_3 = \frac{1}{8}$.

4 Analysis of Quasi-Uniform Subdivision

A *quasi-uniform bivariate scheme* consists of different uniform rules on each side of the y-axis, far enough from the axis, some different rules near the y-axis, and is uniform in the y-direction. This is the case in the triangular-quadrilateral setting, which we describe in the following. We demonstrate how to construct a scheme over the tri-quad grid, which generates all quadratic polynomials, and how to analyze its smoothness. In particular, we establish its C^2-continuity.

The method of construction is the same as we used in the univariate case in § 3.5, namely, we fix Q, and then solve the linear equation $SQ = Q\sigma$ (from Corollary 2), in which the unknowns are the subdivision weights.

For the smoothness analysis, we use a special procedure, first published in [4], which requires to estimate the joint spectral radius of two matrices. In particular, the polynomial reproduction property plays an important role in the smoothness analysis procedure (see § 4.2). The proof of correctness of the smoothness analysis procedure can be found in [4].

4.1 The Tri-quad Scheme – Construction

Considering the *tri-quad grid* in Figure 3, we would like to define a quasi-uniform scheme over this grid, which coincides with the tensor product cubic B-spline scheme, or the Catmull-Clark scheme [27], on the right half plane, and the C^2 quartic three-directional box-spline scheme, or the Loop scheme [28], on the left half plane. The masks of these schemes are depicted in Figure 4.

The goal is to define special rules on the y-axis and near it so that overall the scheme will be C^2, i.e., as smooth as the right and left schemes. These special rules are constructed together with an operator Q, which also requires a special definition near the y-axis, so that the condition $SQ = Q\sigma$ holds for π_2 over the entire plane. The operator Q away from the y-axis is derived from the right and left uniform schemes, using the method of [2]:

$$Qf = Q^+f = f - \frac{1}{6}f_{xx} - \frac{1}{6}f_{yy} \ , \quad x \geq 0 \ ,$$

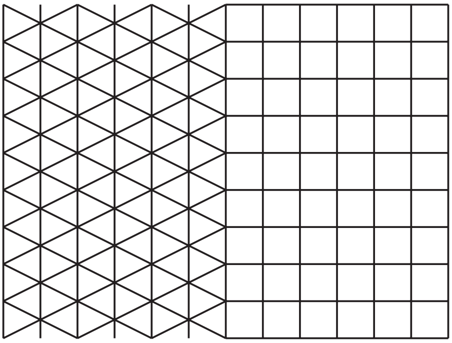


Fig. 3. The tri-quad grid

$$Qf = Q^{-}f = f - \frac{1}{6}f_{xx} - \frac{1}{8}f_{yy}\ , \quad x < 0\ .$$

Given this choice of Q, the special subdivision rules near the y-axis are defined by requiring the $SQ = Q\sigma$ over $\pi_2(I\!R^2)$. The equations coming out of this equation are solvable, but not uniquely. The challenge is to find a scheme of the smallest possible support which fulfills the equations. A scheme with positive weights and of small support, though probably not the smallest possible, is described by the rules shown in Figure 5. Note that the convolution stencil (c) is only used for calculating temporary values before the application of the uniform left scheme.

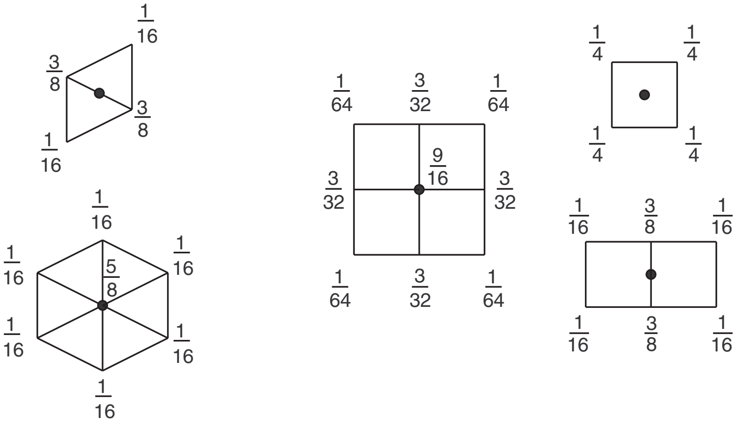


Fig. 4. The scheme masks away from the y-axis: Catmull-Clark scheme on the right and Loop scheme on the left

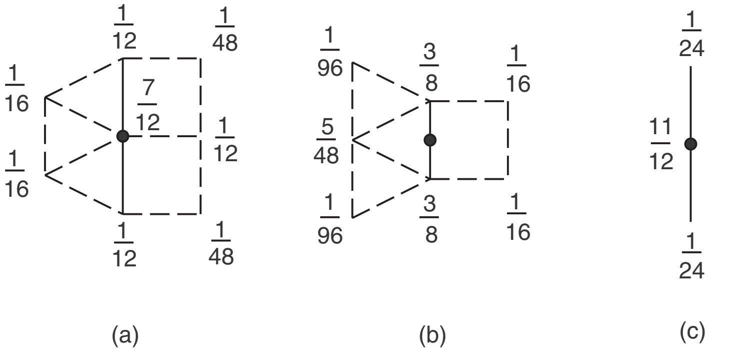


Fig. 5. The scheme near the y-axis: (a) The stencil for a new value at old grid points on the y-axis. (b) The stencil for a new value at new grid points on the y-axis. (c) The stencil of the operator defining temporary values on the y-axis before the application of the Loop scheme on $x < 0$

Now, that we have a scheme which reproduces all quadratic polynomials, it is left to show that this scheme generates C^2 limit functions over the entire plane.

Remark 1. The choice $Q = Q^+$ on the y-axis is somewhat arbitrary. Different choices of Q lead to different subdivision rules. By experimenting with other choices of Q on the y-axis, we found that for some of them there does not exist subdivision schemes S with positive weights. (e.g. $Q = Q^-$ or $Q = \frac{Q^- + Q^+}{2}$ on the y-axis). With $Q = Q^+$ on the y-axis we were able to get a subdivision scheme that consists of only three special rules, in which all weights are positive. Warren and Schaefer, in [5], chose $Q = \frac{Q^- + Q^+}{2}$ on the y-axis, which results in a scheme that has negative weights (They also extended the tri-quad scheme to meshes of arbitrary topology).

4.2 The Analysis Procedure and the Tri-quad Example

In the following, we describe the procedure for checking whether a given quasi-uniform scheme S is C^m. We assume that S generates polynomials up to degree m, in the sense that (13) is satisfied for some Q. The justification of the different steps is given in [4].

The following method was employed by Warren and Schaefer for analysing their version of tri-quad subdivision [5], and also by Hakenberg [7], for the analysis of volumetric subdivision schemes over three-dimensional meshes.

The Analysis Procedure:

1. Let $L \subset X$ denote a subset of mesh points around the origin such that the values of the limit function in $[-1, 1] \times [0, 1]$ depend only on control points in L. Furthermore, the values at iteration 1 in L and in $\mathcal{E}L$, namely $SP|_L$ and $SP|_{\mathcal{E}L}$, depend only on the initial values in L, $P|_L$, where $\mathcal{E}$ is a shift operator, $\mathcal{E}L = \{(i, j + 1)|(i, j) \in L\}$.
2. Let A denote the local subdivision operator taking values in L to values in L after one subdivision iteration. Let B denote the operator taking values in L to values in $\mathcal{E}L$.
3. Using the left and right eigenvectors of A, form a basis V for the vectors of values in L such that the matrix form of A in the new basis is

$$\widetilde{A} = \left[\begin{array}{c|c} \Lambda & C_0 \\ \hline 0 & Y_0 \end{array}\right] , \tag{18}$$

   Where $\Lambda = diag(1, 0.5, 0.5, ..., 2^{-m}, ..., 2^{-m})$. One way to do it is to complete the $(m+1)(m+2)/2$ right eigenvectors,

$$Qf|_L \ , \quad f = x^i y^j \ , \quad 0 \le i + j \le m \ , \tag{19}$$

   to a basis.
4. From the polynomial generation assumption about the scheme, it turns out that the matrix form of B in the basis V is

$$\widetilde{B} = \left[\begin{array}{c|c} \Theta & C_1 \\ \hline 0 & Y_1 \end{array}\right] , \tag{20}$$

where Θ is an upper-triangular matrix that has the same diagonal as Λ. Moreover, Θ has certain zero values above the diagonal, creating such diagonal blocks of sizes $1, 2, 3, 4, \ldots$, e.g., for $m = 2$

$$\Theta = \left[\begin{array}{c|cc|ccc} 1 & * & * & * & * & * \\ \hline 0 & 0.5 & 0 & * & * & * \\ 0 & 0 & 0.5 & * & * & * \\ \hline 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 \end{array}\right] . \tag{21}$$

5. A sufficient condition for C^m continuity is that the *joint spectral radius* of Y_0 and Y_1, $\rho_\infty(Y_0, Y_1)$, is strictly less than 2^{-m}, where

$$\rho_\infty(Y_0, Y_1) = \limsup_{k \in \mathbb{Z}_+ \backslash 0} \left(\max \left\{ \|Y_{\varepsilon_k} Y_{\varepsilon_{k-1}} \cdots Y_{\varepsilon_1}\|_\infty : \varepsilon_i \in \{0, 1\}, i = 1, \ldots, k \right\} \right)^{\frac{1}{k}} . \tag{22}$$

Moreover, if $\rho_\infty(Y_0, Y_1) = 2^{-(m+\alpha)}$, $0 < \alpha \leq 1$ then the m-th order derivatives of the limit function are Hölder continuous with exponent $\alpha - \epsilon$ for arbitrarily small $\epsilon > 0$. Of course, this only holds if the limit function away from the y-axis is known to have that Hölder exponent.

Remark 2. Practical methods for estimating the joint spectral radius are given in an appendix in [8]. In particular, an upper bound for the joint spectral radius $\rho_\infty(Y_0, Y_1)$ can be computed by estimating the norms of all possible products of finite length k of Y_0 and Y_1. I.e.,

$$\rho_\infty(Y_0, Y_1) \leq \rho_\infty^{[k]}(Y_0, Y_1) , \tag{23}$$

where

$$\rho_\infty^{[k]}(Y_0, Y_1) = \left(\max \left\{ \|Y_{\varepsilon_k} Y_{\varepsilon_{k-1}} \cdots Y_{\varepsilon_1}\|_\infty : \varepsilon_i \in \{0, 1\}, i = 1, \ldots, k \right\} \right)^{\frac{1}{k}} . \tag{24}$$

Remark 3. The condition $\rho_\infty(Y_0, Y_1) < 2^{-(m+\alpha)}$, in view of the special basis V used in (18), implies that the mth degree Taylor expansion coefficients of $S^\infty P$ at dyadic points on the y-axis are all uniformly bounded. This is the main idea behind the proof in [4]

4.3 The Tri-quad Scheme – C^2 Analysis

Let us apply our analysis tool to the tri-quad scheme presented in § 4.1. The set L is the set of $|L| = 45$ points

$$L = \{(i, j) : i = 0, 1, 2 \ , \ -4 \leq j \leq 4 \ , \ j \in \mathbb{Z}\} \cup$$
$$\{(i, j + 0.5i) : i = -1, -2 \ , \ -4 \leq j \leq 4 \ , \ j \in \mathbb{Z}\} .$$

The matrices A and B are evaluated as follows: First we choose an ordering of the points in L, $L = \{(i_1, j_1), \cdots, (i_{|L|}, j_{|L|}\}$. An entry $A_{k,\ell}$ in A corresponds to a pair of points $((i_k, j_k), (i_\ell, j_\ell))$. Applying the subdivision scheme to initial data set $P = \delta_{(i_\ell, j_\ell)}$ which is 1 at the point (i_ℓ, j_ℓ) and zero elsewhere, we have

$$A_{k,\ell} = (S\delta_{(i_\ell,j_\ell)})_{(i_k,j_k)}, \quad k = 1, \cdots, |L|, \quad \ell = 1, \cdots, |L|.$$

The entries of the matrix B are

$$B_{k,\ell} = (S\delta_{(i_\ell,j_\ell)})_{(i_k,j_k+1)}, \quad k = 1, \cdots, |L|, \quad \ell = 1, \cdots, |L|.$$

The matrices $\widetilde{A}$ and $\widetilde{B}$ are just the representation of A and B, respectively, in another basis V. The construction of this basis is described in item 3 of the analysis procedure above, and it involves the computation of the polynomial eigenvectors of S by (19).

The upper-left block Θ of $\widetilde{B}$ for the tri-quad scheme is

$$\Theta = \left[\begin{array}{c|cc|ccc} 1 & -0.1859 & 0.0476 & -0.0039 & 0.0271 & -0.0181 \\ \hline 0 & 0.5 & 0 & -0.0036 & -0.1398 & 0.0921 \\ 0 & 0 & 0.5 & -0.0968 & 0.0241 & -0.0216 \\ \hline 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 \end{array}\right]. \tag{25}$$

A bound for $\rho_\infty(Y_0, Y_1)$ may be estimated by $\rho_\infty^{[k]}(Y_0, Y_1)$ using Remark 2, and this is used to compute a lower bound $\alpha_k = -2 - log_2(\rho_\infty^{[k]}(Y_0, Y_1))$ of the Hölder exponent. We obtained

$$\alpha \geq \alpha_{18} = -2 - log_2(\rho_\infty^{[18]}(Y_0, Y_1)) = 0.5942. \tag{26}$$

Hence, we deduce that the tri-quad scheme is at least in $C^{2.5942}$. A straightforward extrapolation of the values α_k as a function of $1/k$ indicates that they tend to 1, leading to the conjecture that the tri-quad scheme is $C^{3-\epsilon}$ for any $\epsilon > 0$. This conjecture is, at least, in agreement with the spectral radii of Y_0 and Y_1, $\rho(Y_0) = \rho(Y_1) = \frac{1}{8}$.

5 C^2 Subdivision Around an Extraordinary Vertex

In this section, we summarize the results of a recent paper by Zulti et. al [8], in which the above method was utilized for designing subdivision rules that generate C^2-continuous limit functions around an extraordinary vertex.

In commonly-used subdivision schemes, such as Loop and Catmull-Clark, smoothness analysis around extraordinary vertices requires to find a special parametrization (*characteristic map*), in which the surface can be written as a differentiable function of two variables [9, 10, 11].

Typically, subdivision schemes generate surfaces that are C^1-continuous in the vicinity of extraordinary points. However, most subdivision schemes do

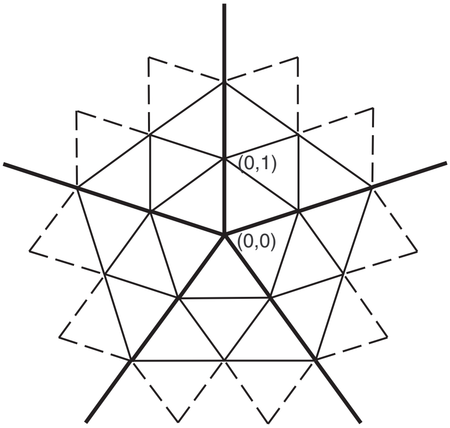


Fig. 6. The grid X_n - a triangulation with one extraordinary vertex ($n = 5$)

not achieve C^2-continuity near extraordinary points. One way to achieve C^2-continuity is to enforce zero curvature (flatness) at extraordinary points, which is a disturbing shape deficiency [29]. The degree estimates by Prautzsch and Reif [30] suggest that a subdivision scheme must be quite complicated in order to achieve C^2-continuity without flat points. Such schemes have been shown to exist by Prautzsch [31] and by Reif [32], but they have especially large supports, and are considered impractical for applications in surface design.

The setting chosen by Zulti and Levin [8] differs from the standard setting of subdivision, by the fact that the underlying mesh is assumed be regular, with the exception of a single extraordinary vertex. Such a mesh fits well into the framework of non-uniform stationary subdivision (see Fig. 1(d)). We let X_n denote the set of vertices of the infinite triangulation, where the extraordinary vertex is positioned at the origin, and where all edges emanating from it have length 1, and all triangles are equivalent (see Fig. 6). In the following, n stands for the valency of the extraordinary vertex. The n infinite rays emanating from the extraordinary vertex are called *extraordinary lines*, and are marked by thick lines in Fig. 6.

The scheme is constructed by the same method as we used for the piecewise-uniform univariate scheme in § 3.5 and the tri-quad scheme in § 4.1. The smoothness analysis is mainly based on the procedure described in § 4.

As in the tri-quad scheme from § 4.1, this is again a demonstration of the fact that the route to high order smoothness goes through polynomial reproduction. In particular, before we can achieve C^2 continuity, we must fulfill the conditions for reproduction of quadratic polynomials.

The new scheme is based on the quartic three-directional Box-spline scheme (as in Loop's scheme [28]), and is guaranteed to generate C^2 limit functions whenever the valency n of the extraordinary vertex is in the range $4 \leq n \leq 20$. It consists of two special sets of subdivision rules: Subdivision stencils that operator in the vicinity of the *extraordinary vertex*, and subdivision stencils that operate along the *extraordinary lines*. This is in contrast to the standard subdivision schemes (e.g. Loop's scheme) which only use special subdivision rules near the extraordinary vertex.

The parameterization used to define the surface is easily derived from the mesh X_n. In the characteristic map approach, the parameterization depends not only on the mesh connectivity, but also on the subdivision weights, making the problem of deriving subdivision weights non-linear and complicated. The addition of the special rules along extraordinary lines, together with the choice of a simple parameterization, allow us to achieve C^2 continuity via polynomial reproduction.

Unfortunately, because of the existence of *extraordinary lines*, the scheme cannot be directly applied to meshes of arbitrary topology. On-going work is being done by Zulti et. al. on using this method for C^2 hole-filling, which has practical applications.

5.1 Subdivision Near the Extraordinary Lines

In this section we construct and analyze subdivision rules that operate in the vicinity of the extraordinary lines, emanating from the extraordinary vertex. We consider a grid X_c, in which the extraordinary line is an infinite boundary between two regular domains, as shown in Fig. 7. With $c = cos(\frac{2\pi}{n})$, this configuration is similar, up to an affine transformation, to the configuration of triangles near the extraordinary lines of X_n (Fig. 6).

Construction

From each side of the extraordinary line, and away from the extraordinary vertex, we will use the well-known quartic Box-spline subdivision scheme, which was previously generalized by Loop to arbitrary triangular meshes [28] (see left part of Fig. 4). This defines the subdivision operator S away from the extraordinary line.

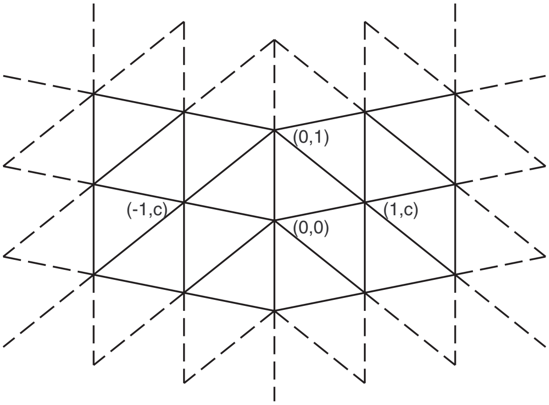


Fig. 7. The grid X_c—an infinite boundary between two regular domains

In the following, we adopt the following notation for partial derivatives of polynomials, in order to shorten some of the formulas:

$$f_{ij} = \frac{\partial^{i+j}}{\partial x_1^i \partial x_2^j} f.$$

Using the method described in [2], it is easy to show that the corresponding Q operator, for reproduction of all cubic bivariate polynomials, is given by

$$Qf(x) = f(x) - \tfrac{1}{6}\left(f_{11} + (2c-1)f_{12} + (c^2 - c + 1)f_{22}\right)(x), \quad x_1 < 0, \tag{27}$$

on the left half-plane, and

$$Qf(x) = f(x) - \tfrac{1}{6}\left(f_{11} - (2c-1)f_{12} + (c^2 - c + 1)f_{22}\right)(x), \quad x_1 > 0, \tag{28}$$

on the right half-plane. Here c is a parameter that determines the grid X_c, over which the scheme is defined (Fig. 7). In order to complete the definition of Q for all $x \in X_c$, we define Q on the extraordinary line by an average of (27) and (28),

$$Qf(x) = f(x) - \frac{1}{6}\left(f_{11} + (c^2 - c + 1)f_{22}\right)(x), \quad x_1 = 0, \tag{29}$$

The strategy we employ, for computing the subdivision operator S, is based on requiring that S reproduces all quadratic polynomials. As a consequence of Corollary 2, this requirement reduces to a system of linear equations in which the unknowns are the subdivision weights.

We have to complete the definition of S by computing new subdivision rules at all points where the regular Box-spline rule uses points on the extraordinary line. Therefore, we need to redefine $SP(x)$ for all $x \in X$ such that $-2 \leq x_1 \leq 2$. Due to the symmetry between the right and left sides of the extraordinary line, and the invariance under integer shifts in the upwards direction, we only need to compute six new stencils. Fig 8 depicts the new stencils with the unknown subdivision weights.

Remark 4. Formula (29) that defines Q on the extraordinary line is not the unique choice that leads to a C^2 scheme. Other formulas that we tried led to schemes with lower fractional smoothness, or to complicated formulas for the subdivision weights. It is possible, though, that a better choice of Q still exists. The choice of stencils in Fig. 8 is also not unique. In particular, stencil #1, which uses 8 points, could be replaced by a four-point stencil, at the expense of lower fractional smoothness of the limit function.

In order to achieve reproduction of all quadratic polynomials, we require that S satisfies (13) for $m = 2$. For each of the unknown stencils, we reduce (13) to a system of linear equations. As a detailed example, consider stencil #6, with the 6 unknowns $w_{61}, \ldots, w_{66}$. Without loss of generality, we place the origin at the vertex with the weight w_{62}. The formal notation for stencil #6 then becomes:

$$\begin{aligned} SP(2, 2c-1) = {} & w_{61}P(0,-1) + w_{62}P(0,0) + w_{63}P(0,1) \\ & + w_{64}P(1,c-1) + w_{65}P(1,c) + w_{66}P(2,2c-1). \end{aligned} \tag{30}$$

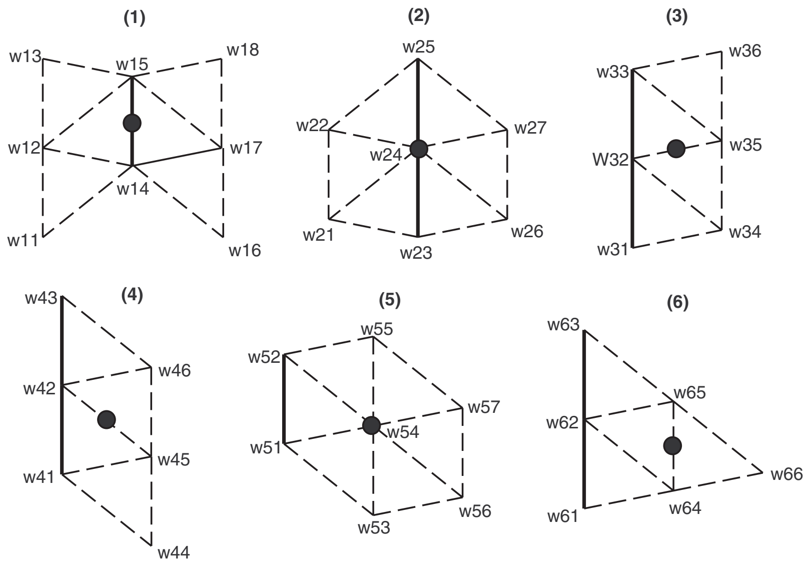


Fig. 8. Stencils near the extraordinary line

From (13) we get the requirement

$$SQf(2,2c-1) = Q\sigma f(2,2c-1), \quad \forall f \in \pi_2. \tag{31}$$

Using (30) and (28)-(29), we compute $SQf(2,2c-1)$ with f taken from a basis of the polynomials up to degree 2. This gives us the left-hand side of (31). For the right-hand side, $Q\sigma f(2,2c-1)$, we simply substitute σf in (28), and evaluate at $(2,2c-1)$. This yields the following 6 equations, for the polynomials $f = 1$,x_1,x_2,x_1^2,x_1x_2, $x_2^2 + \frac{1}{3}(c^2-c+1)$ respectively.

$$\begin{aligned}
w_{61} + w_{62} + w_{63} + w_{64} + w_{65} + w_{66} &= 1 \\
w_{64} + w_{65} + 2w_{66} &= 1 \\
-w_{61} + w_{63} + (c-1)w_{64} + cw_{65} + (2c-1)w_{66} &= c - \tfrac{1}{2} \\
-\tfrac{1}{3}\left(w_{61} + w_{62} + w_{63}\right) + \tfrac{2}{3}\left(w_{64} + w_{65}\right) + \tfrac{11}{3}w_{66} &= \tfrac{11}{12} \\
\tfrac{4c-5}{6}w_{64} + \tfrac{4c+1}{6}w_{65} + \tfrac{11}{6}(2c-1)w_{66} &= \tfrac{11}{24}(2c-1) \\
w_{61} + w_{63} + (c-1)^2 w_{64} + c^2 w_{65} + (2c-1)^2 w_{66} &= \tfrac{5c^2-5c+2}{4}
\end{aligned}$$

Remark 5. The usage of $f = x_2^2 + \frac{1}{3}(c^2-c+1)$ instead of $f = x_2^2$ serves the purpose of simplifying the last equation, but is not necessary. Any choice of basis for the quadratic polynomials yields an equivalent linear system.

This system has a unique solution for any value of c, given in Table 1. For stencils 1-5 we found that the system of 6 equations resulting from (13) has an infinite number of solutions. We therefore added extra conditions until the solution became unique for all c. In particular, the extra conditions are chosen such

Table 1. Subdivision weights near the extraordinary line, and the additional conditions that were imposed for uniqueness. For valence n, substitute $c = cos(\frac{2\pi}{n})$

$w_{11} = w_{16},\ w_{13} = w_{18}$ $w_{14} = w_{15}$ .... $w_{11} = w_{16} = \frac{-c+2c^2}{16}$ $w_{12} = w_{17} = \frac{1+4c-4c^2}{16}$ $w_{13} = w_{18} = \frac{1-3c+2c^2}{16}$ $w_{14} = w_{15} = \frac{3}{8}$	$w_{23} = w_{25}$ .... $w_{21} = w_{26} = \frac{c}{8}$ $w_{22} = w_{27} = \frac{1-c}{8}$ $w_{23} = w_{25} = \frac{1-2c+2c^2}{8}$ $w_{24} = \frac{1+c-c^2}{2}$	$w_{31} = w_{36}$ .... $w_{31} = w_{36} = \frac{1-3c+2c^2}{24}$ $w_{32} = w_{35} = \frac{9+2c-4c^2}{24}$ $w_{33} = w_{34} = \frac{2+c+2c^2}{24}$
$w_{43} = w_{44}$ .... $w_{41} = w_{46} = \frac{5-5c+2c^2}{24}$ $w_{42} = w_{45} = \frac{7+6c-4c^2}{24}$ $w_{43} = w_{44} = \frac{-c+2c^2}{24}$	$w_{53} = w_{55}$ .... $w_{51} = w_{57} = \frac{5+2c}{96}$ $w_{52} = w_{56} = \frac{7-2c}{96}$ $w_{53} = w_{55} = \frac{7-4c+4c^2}{96}$ $w_{54} = \frac{29+4c-4c^2}{48}$	$w_{61} = \frac{1-3c+2c^2}{48}$ $w_{62} = \frac{5+4c-4c^2}{48}$ $w_{63} = \frac{-c+2c^2}{48}$ $w_{64} = \frac{17+2c}{48}$ $w_{65} = \frac{19-2c}{48}$ $w_{66} = \frac{1}{8}$

that we get the regular Box-spline scheme in the specific case $n=6$ ($c= \frac{1}{2}$). The extra conditions are presented in Table 1. For the entire collection of subdivision weights derived in this paper, the reader is referred to the Excel sheet in [33].

Analysis

The above construction guarantees that S, as defined over the grid X_c reproduces all quadratic polynomials, for any value of c. For the purpose of subdivision over X_n, we are only interested in the particular case $c = cos(\frac{2\pi}{n})$ for $n \geq 4$.

Our subdivision scheme over X_c falls into the category of *quasi-uniform* schemes, because the same subdivision weights are used everywhere along the extraordinary line. Having established the reproduction of all quadratic polynomials, we can directly use the necessary and sufficient condition for convergence and for C^m continuity from Levin and Levin [4], as summarized in § 4.2

The most difficult part of the analysis procedure is the estimation of joint spectral radius (step 5). The definition of joint spectral radius, and a collection of techniques for estimating it, are given in an appendix in [8]. Using the methods therein, we have shown that in the range $0 \leq c \leq 1$, the scheme is at least $C^{2.5847}$-continuous. The range $0 \leq c \leq 1$ contains all values of $c = cos(\frac{2\pi}{n})$, for valencies $n \geq 4$. For $c = 0, \frac{1}{2}, 1$ it can be shown that the joint spectral radius is exactly $\frac{1}{8}$, and therefore the scheme is $C^{3-\epsilon}$-continuous. The scheme has been found to be at least C^2-continuous in the range $-\frac{3}{8} \leq c \leq \frac{11}{8}$, which does not include valency $n = 3$, where $c = -\frac{1}{2}$.

5.2 Subdivision Near the Extraordinary Vertex

In this section we complete the definition of the subdivision scheme on the grid X_n (see Fig. 6), by introducing new subdivision rules that operate near the extraordinary vertex.

The method of construction is the same as the one used to construct the special rules near extraordinary lines. We define the operator Q over X_n, and then find a set of subdivision rules that satisfy (13). The new subdivision scheme is C^2-continuous for valencies $4 \leq n \leq 20$.

Construction

We start by defining the operator Q over X_n. Recall that the operator Q over X_c is given by (27)-(29). We define Q at any point in X_n, except for the origin, to be equivalent to (27)-(29), but since the parametrization of X_n is different than that of X_c, we rewrite Q as follows:

$$Qf(x) = f(x) - \tfrac{1}{24}\sum_{k=1}^{6} D^2_{x^k - x} f, \quad \forall f \in \pi_2, \quad x \in X_n \setminus \{0\}, \tag{32}$$

where $x^1, \ldots, x^6 \in X_n$ denote the 6 neighboring vertices of x, and $D^2_{x^k-x}$ represents the directed second derivative in the direction of the k-th edge emanating from x, defined by

$$D^2_{(a,b)} f = a^2 f_{11} + 2ab f_{12} + b^2 f_{22}.$$

It can be shown that (32) coincides with (27)-(29) over the grid X_c, and is invariant to affine reparametrizations of the grid. We define Q at the origin by the formula

$$Qf(0) = f(0) - \left(\frac{1}{6} - \frac{c}{12}\right)(f_{11} + f_{22}), \quad , f \in \pi_2. \tag{33}$$

Remark 6. This choice of Q is somewhat arbitrary. The reasoning that led to the expression in (33) is the following: From symmetry considerations it follows that Q must have the form $Qf(0) = f(0) - q(f_{11} + f_{22})$. The value of q is such that $f(0) - Qf(0)$ is the average of $f(x) - Qf(x)$ along the n extraordinary lines for all $f \in \pi_2$.

The special subdivision rules near the vertex involve 8 new stencils, as depicted in Fig. 9. For the edge rules, we choose the 8-point stencil, as in the Butterfly scheme [34]. Although this is not necessary for achieving C^2 smoothness, this provides us with additional degrees of freedom for maximizing the fractional smoothness.

The weights for each of the 8 special subdivision rules are computed by solving a system of linear equations. Six of the linear equations arise directly from substituting the 6 monomials in (13). The remaining degrees of freedom are used to enforce symmetries, and to tune the scheme for best smoothness properties. In particular, we make sure that for $n = 6$ (regular mesh), we get the regular Box-spline scheme.

For the complete derivation, the reader is referred to [8]. The entire collection of subdivision weights is also available in the form of an Excel Sheet, from [33].

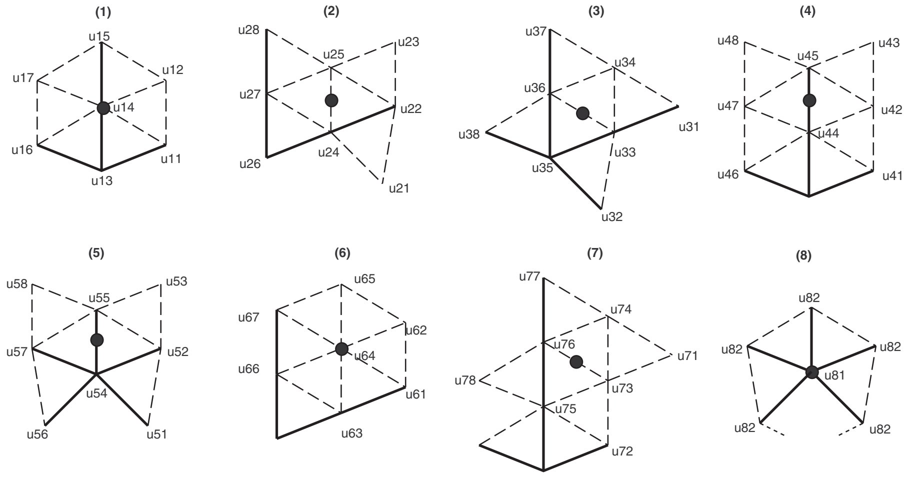


Fig. 9. Stencils near the extraordinary vertex

Analysis

The C^m smoothness around any given vertex of the initial mesh is easy to characterize, provided that we have established the smoothness in the regular regions, and along the extraordinary lines. By construction, we guarantee that the local subdivision operator has the eigenvalues $1,\frac{1}{2},\frac{1}{2},\dots,2^{-m},\dots,2^{-m}$, with eigenvectors that produce the polynomials up to degree m in the limit (see § 2.3). We only need to establish the smoothness of the limit functions corresponding to the rest of the eigenvectors. Therefore, a sufficient condition for C^m continuity, is that all the rest of the eigenvalues have modulus less than 2^{-m}. This is stated formally in [8].

For valencies $4 \leq n \leq 20$, we found that the subdivision matrix has the leading eigenvalues $1,\frac{1}{2},\frac{1}{2}, \frac{1}{4},\frac{1}{4},\frac{1}{4}, \lambda_n$, with $\lambda_n < \frac{1}{4}$. As the valency increases, for

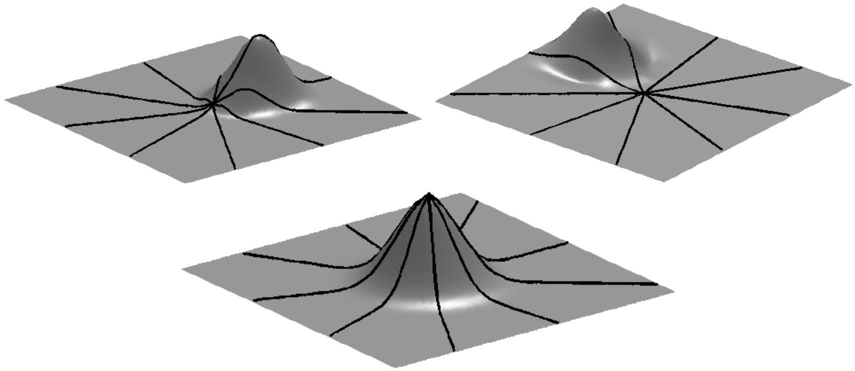


Fig. 10. Basis functions near the extraordinary vertex of valency 9

$n > 6$, the smoothness decreases, but it remains above C^2 for $4 \leq n \leq 20$. For more details, the reader is referred to [8].

Figure 10 shows some 'basis functions' of the subdivision schemes, i.e., the limit functions which result from initial data which is 1 at one vertex and 0 everywhere else. The black lines correspond to values of the limit function on the extraordinary lines.

6 Subdivision with Transfinite Boundary Conditions

Subdivision schemes that consider transfinite boundary conditions were first introduced under the name *Combined Subdivision Schemes* in [3]. The limit surfaces generated by these schemes are capable of satisfying boundary conditions that are not necessarily given by polynomials or spline functions. For example, the scheme described in [12] generates limit surfaces that interpolate arbitrary parametric curves. The curves are not required to be subdivision curves or spline curves. This is achieved by special subdivision rules that operate near the curves, and involve evaluations of the parametric curves at each refinement iteration.

In section 6.1 we give a brief introduction to the theory and practice of Combined Subdivision Schemes. As a demonstration of their construction and application, section 6.2 presents a modification of the Catmull-Clark scheme, which is designed to generate surfaces under C^1 boundary conditions. For a more in-depth and formal presentation, the reader is referred to [3].

6.1 An Introduction to the General Theory of Combined Subdivision Schemes

Definition

Recall that ordinary subdivision schemes are defined by

$$P^{n+1} = SP^n, \quad n = 0, 1, \ldots,$$

where $P^n \in l(\mathbb{Z}^s)$ signifies a set of control points at level n. A *Combined subdivision scheme* takes the form

$$P^{n+1} = S\left(P^n, \sigma^n f\right), \quad n = 0, 1, \ldots, \tag{34}$$

where σ is the dilation operator $\sigma f = f\left(\frac{\cdot}{2}\right)$. The function $f : \mathcal{E} \subset \mathbb{R}^s \longmapsto \mathbb{R}$, plays the role of the boundary condition. It is defined over a subset $\mathcal{E}$ of the s-dimensional space, which we call the *exterior* of the domain of S. The limit function $S^\infty(P, f)$ is defined over $\mathbb{R}^s \setminus \mathcal{E}$.

Different choices of $\mathcal{E}$ serve the purposes of different applications (see Fig. 11). For prescribing the value and partial derivatives of the limit function at isolated points, we use $\mathcal{E} = \{0\}$. For satisfying transfinite boundary conditions given along a hyper-plane, we use $\mathcal{E} = \{x \in \mathbb{R}^s \mid x_1 \leq 0\}$. The boundary is then associated with the hyper-plane $x_1 = 0$, and the boundary condition is formally defined over the entire half-space $\mathcal{E}$. For interpolating given values and partial

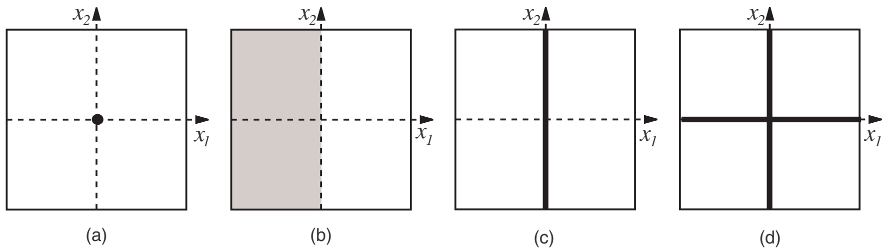


Fig. 11. Different exteriors for bivariate combined subdivision schemes. (a) $\mathcal{E} = \{0\}$. (b) $\mathcal{E} = \{x \in \mathbb{R}^2 \mid x_1 \leq 0\}$. (c) $\mathcal{E} = \{x \in \mathbb{R}^2 \mid x_1 = 0\}$. (d) $\mathcal{E} = \{x \in \mathbb{R}^2 \mid x_1 \cdot x_2 = 0\}$. The limit function of the combined subdivision scheme is defined over $\mathbb{R}^2 \setminus \mathcal{E}$

derivatives on a hyper-plane, by a bivariate function defined over the entire plane, we use $\mathcal{E} = \{x \in \mathbb{R}^s \mid x_1 = 0\}$. For interpolating given values and partial derivatives on the x and y axes, we use $\mathcal{E} = \{x \in \mathbb{R}^2 \mid x_1 \cdot x_2 = 0\}$.

We assume that the operator S is a linear and local operator, both in P and in f. We also assume that S is bounded, in the sense that $S(P, f)$ is bounded by a constant times the magnitude of certain partial derivatives and values of f. S is allowed to use values and partial derivatives of f, and in general, any linear, bounded and local functional applied to f. For the exact formulation of these assumptions, see [3].

Polynomial Reproduction

The formulation of polynomial reproduction uses the notion of the operator Q, introduced in § 2.2, and extends it to this context. We say that S reproduces all polynomials up to degree m, if there exists an operator $Q : \pi_m(\mathbb{R}^s) \mapsto \ell(\mathbb{Z}^s)$ such that

$$S^\infty(Qf, f) = f, \quad \forall f \in \pi_m(\mathbb{R}^s). \tag{35}$$

A simple of extension of Lemma 3 states that (35) follows if we have

$$S(Qf, f) = Q\sigma f, \quad f \in \pi_m, \tag{36}$$

where σ is the dilation operator $\sigma f = f(\frac{\cdot}{2})$. As a consequence, the requirement of polynomial reproduction is reduced to a set of linear equations in which the unknowns are the subdivision weights. This is the key for constructing new subdivision schemes, as demonstrated by the example in § 6.2.

Smoothness Analysis

The C^m smoothness criterion requires that $S^\infty(P, f)$, defined over the open set $\mathbb{R}^s \setminus \mathcal{E}$ can be extended to a C^m continuous function defined over the closure of $\mathbb{R}^s \setminus \mathcal{E}$, for any P and any smooth enough f.

Since f play the role of boundary conditions, we also require that the limit function $S^\infty(P, f)$ has a C^r connection with f, for some $0 \leq r \leq m$. More accurately, we require that the function which coincides with f over $\mathcal{E}$, and coincides with $S^\infty(P,f)$ over $I\!\!R^s \setminus \mathcal{E}$, is C^r-continuous.

In the open set $I\!\!R^s \setminus \mathcal{E}$, smoothness is typically guaranteed by the properties of the ordinary subdivision scheme that is applied away from the boundary. Special analysis tools are required in order to establish the smoothness near the boundary of $\mathcal{E}$, for arbitrary (yet smooth enough) functions f. Fortunately, there exists a theorem that reduces the analysis to the case $f \equiv 0$.

It is shown in [3], that if S satisfies the requirement of polynomial reproduction up to degree m (35), then the same smoothness properties that hold for $S^\infty(P,0)$ will hold for $S^\infty(P,f)$, as long as f is C^m-continuous itself. It is also proven that $S^\infty(P,f)$ will always have a C^r connection ($r \leq m$) to f at the boundary of $\mathcal{E}$, if all partial derivatives of order $\leq r$ of $S^\infty(P,0)$ vanish there.

As a consequence of this powerful theorem, smoothness analysis can concentrate entirely on the case $f = 0$. For this case, existing analysis tools can be used, such as eigenvalue analysis, as in § 5, and the joint-spectral radius test from § 4.

6.2 Example: Catmull-Clark Surfaces with C^1 Boundary Conditions

In this section we develop subdivision rules that operate at the boundary of a Catmull-Clark surface, where C^1 boundary conditions are prescribed. These rules interpolate boundary curves and cross-boundary first order derivatives along the boundary curves. The scheme originally appeared in [13], and the details of its construction and analysis can be found in [3].

As in the tri-quad scheme from § 4.1, we apply certain rules to position the boundary vertices, and then simply use the Catmull-Clark scheme to calculate all of the internal vertices. The boundary rule is chosen in such a way that the overall scheme reproduces all bivariate cubic polynomials, and generates C^2-continuous limit functions (except at extraordinary vertices).

In the following, we show how to construct and analyze the boundary positioning rule, and how to use it for smoothly blending N surfaces surrounding an N-sided hole. Examples of the limit surfaces of our scheme are shown in Figures 12 and 13.

Construction of a Boundary Rule

For the construction and analysis, we consider only regular parts of the mesh, in which bicubic B-spline subdivision is applied. For the formal definition of our combined subdivision scheme, we use the exterior $\mathcal{E} = \{x \in I\!\!R^2 \mid x_1 \leq 0\}$ (see Fig. 11(b)).

We look for a combined subdivision operator S that operates over control points P defined on the right half-plane $Z\!\!\!Z^2 \setminus \mathcal{E}$, and a function f defined on $\mathcal{E}$. We require that S reproduces cubic polynomials in the sense of (35), and that it generates C^2-continuous limit functions, and has C^1-continuous connection with the function f.

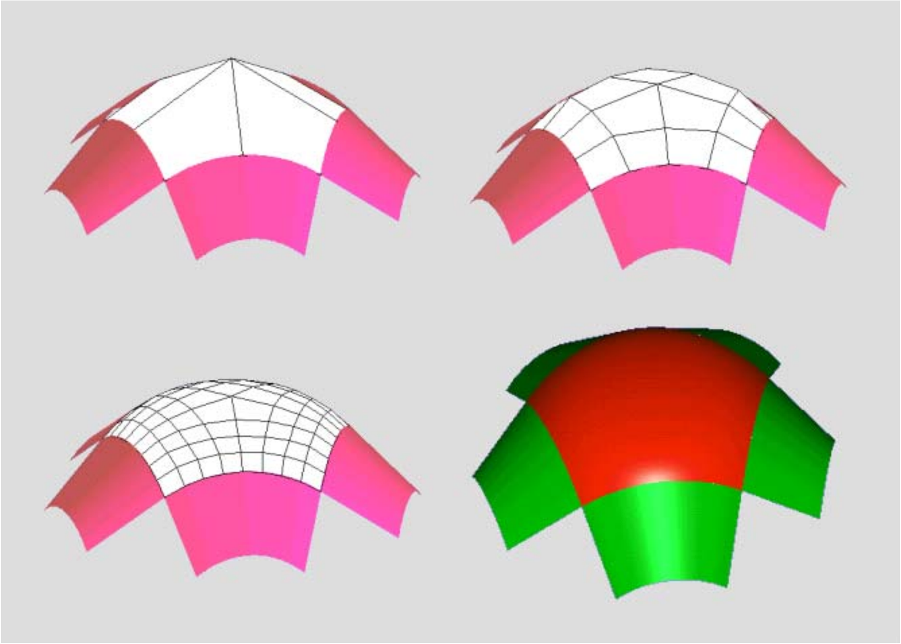


Fig. 12. Filling a 5-sided hole with C^1 boundary conditions: the initial data, two iterations of subdivision, and the limit surface

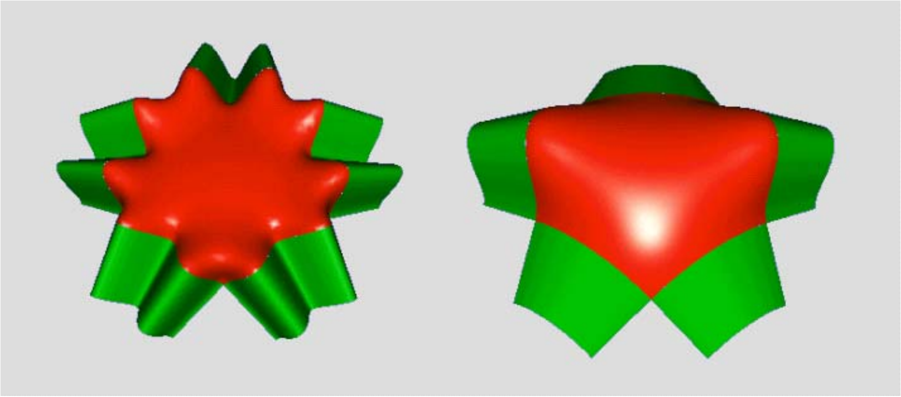


Fig. 13. 5-sided surfaces generated by our algorithm

We choose to construct S by combining a *boundary positioning rule* with an ordinary subdivision rule. In each iteration, we compute the control points P at $x_1 = 0$, by a linear combination of the value of P at $x_2 = 1$ and values and partial derivatives of f at $x_2 = 0$. Then, we apply the well known bicubic B-spline subdivision operator. The boundary positioning is defined as followed (see Fig. 14):

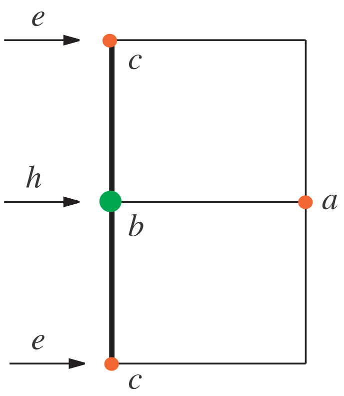


Fig. 14. A stencil for C^1 contact

$$\begin{aligned}P^*(0,\alpha_2) =&aP(1,\alpha_2)+bf(0,\alpha_2)+c\left(f(0,\alpha_2+1)+f(0,\alpha_2-1)\right)+\\ &h\frac{\partial}{\partial x_1}f(0,\alpha_2)+e\left(\frac{\partial}{\partial x_1}f(0,\alpha_2+1)+\frac{\partial}{\partial x_1}f(0,\alpha_2-1)\right)\end{aligned} \tag{37}$$

It is shown in [3] that the Q operator, for bicubic B-spline subdivision, is given by

$$Qf=f-\frac{1}{6}\left(\frac{\partial^2 f}{\partial x_1^2}+\frac{\partial^2 f}{\partial x_2^2}\right), \quad \forall f\in\pi_3.$$

From (36) it is easy to see, that in order to reproduce all cubic polynomials, it is sufficient that the boundary positioning rule satisfies

$$P^*(0,\alpha_2)=Qf(0,\alpha_2), \quad \forall\alpha_2\in\mathbb{Z}, \tag{38}$$

whenever $P=Qf|_{\mathbb{Z}^2\setminus\mathcal{E}}$. By substituting the cubic monomials in f, and $\alpha_2=0$, in (37) and (38), we get the following constraints on a,b,c,h,e:

$$\begin{array}{ll} f=1 & a+b+2c=1\\ f=x_1 & a+h+2e=0\\ f=x_1^2 & \dfrac{2a}{3}=-\dfrac{1}{3}\\ f=x_2^2 & -\dfrac{a}{3}+2c=-\dfrac{1}{3}\\ f=x_1x_2^2 & -\dfrac{a}{3}+2e=0\end{array}$$

The monomials $f=x_2,x_1x_2,x_1^2x_2,x_1^3,x_2^3$ only add redundant linear equations, due to the symmetry in the x_2 direction. The unique solution is

$$a=-\frac{1}{2},\ b=2,\ c=-\frac{1}{4},\ h=\frac{2}{3},\ e=-\frac{1}{12} \tag{39}$$

Smoothness Analysis

As proven in [3], it is sufficient to consider the case $f \equiv 0$. In this case the boundary positioning rule (37) reduces to

$$P^*(0, \alpha_2) = -\frac{1}{2}P(1, \alpha_2).$$

It is then easy to show that the operation of the combined subdivision scheme in the right half plane $\mathbb{Z}^2 \setminus \mathcal{E}$ is equivalent to the uniform bicubic B-spline scheme operating on initial control points P that satisfy

$$\begin{aligned} P(-\alpha_1, \cdot) &= P(\alpha_1, \cdot), \quad \forall \alpha_1 \in \mathbb{Z} \setminus \{0\}, \\ P(0, \cdot) &= -\frac{1}{2}P(1, \cdot). \end{aligned}$$

For such P we have that $S^\infty P$ vanishes on the x_2-axis along with its first order partial derivatives. It follows that S generates C^2-continuous limit functions, with C^1 connection to f at $x_1 = 0$, whenever f is C^2-continuous.

Application to Hole Filling

In the following application, the user prescribes the C^1 boundary conditions in terms of a collection of parametric curves c, with corresponding cross-boundary derivative functions d. The special positioning rule at the boundary follows directly from (37) and (39).

Before the n-th iteration of subdivision, we calculate the position of the vertex v by the following boundary positioning rule: (see Fig. 15)

$$\begin{aligned} p(v) =& 2c(u) - \frac{c(u_1) + c(u_2)}{4} - \frac{1}{2}p(v_1) + \\ & 2^{-n}\left(\frac{2}{3}d(u) - \frac{d(u_1) + d(u_2)}{12}\right). \end{aligned} \tag{40}$$

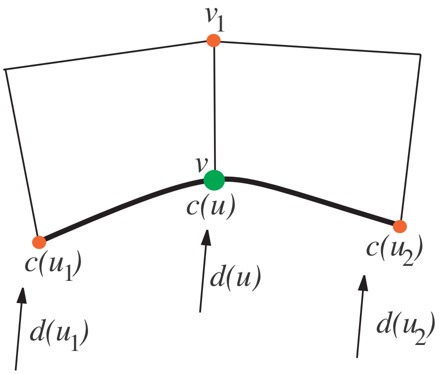


Fig. 15. A boundary rule that involves cross-boundary derivatives

The factor 2^{-n} that multiplies values of d comes from the dilation σ^n in the definition of a combined subdivision scheme (34).

This positioning rule guarantees a smooth limit surface, only if the boundary curve c is smooth, and d is continuous. It is also required that the boundary vertices are evenly spaced along the curve, namely $u - u1 = u2 - u$.

A Corner Rule

To enable the interpolation of a piecewise-smooth boundary, we also need to construct a special positioning rule at the corner between two smooth boundary curves. This is done in [3], by considering a combined subdivision operator with the exterior

$$\mathcal{E} = \{x \in I\!R^2 \mid x_1 \leq 0 \text{ or } x_2 \leq 0\}.$$

In order for the above rule to be applicable to our net, we require that all corner vertices be isolated, and have only two edges emanating from them. We assume, for simplicity of the presentation, that the corner vertex v corresponds to the points $c_1(0) = c_2(0)$ of two intersecting curves, and that the parametric intervals between two boundary vertices at level zero is 1. Figure 16 shows the neighborhood of the corner vertex v before the n-th iteration ($n = 0$ corresponds to the first iteration).

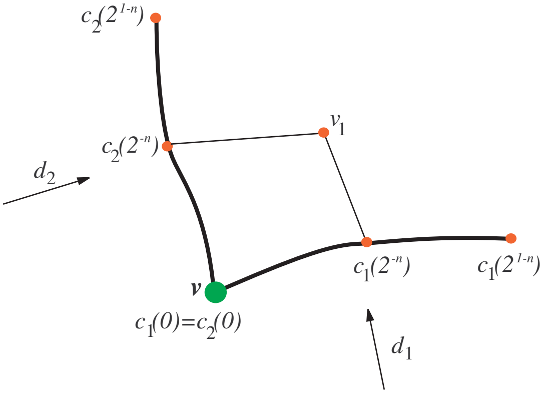


Fig. 16. A corner rule that involves cross-boundary derivatives

The corner rule operates before the n-th iteration of subdivision, and calculates $p(v)$ by the following formula:

$$\begin{aligned} p(v) =& \frac{5}{2}c_1(0) - \left(c_1(2^{-n}) + c_2(2^{-n})\right) \\ &+ \frac{1}{4}p(v_1) + \frac{1}{8}\left(c_1(2^{1-n}) + c_2(2^{1-n})\right) \\ &+ 2^{-n}\left(\frac{29}{48}\left(d_1(0) + d_2(0)\right) - \frac{1}{12}\left(d_1(2^{-n}) + d_2(2^{-n})\right)\right) \\ &- 2^{-n}\left(\frac{1}{48}\left(d_1(2^{1-n}) + d_2(2^{1-n})\right)\right). \end{aligned} \tag{41}$$

Compatibility Conditions

Since the surface first derivatives are determined at the corner vertex by the given curves, the cross-boundary derivatives there $d_1(0)$ and $d_2(0)$ cannot be arbitrarily chosen. For C^1 smoothness, we require that

$$\begin{aligned} d_1(0) &= \dot{c}_2(0), \\ d_2(0) &= \dot{c}_1(0). \end{aligned} \tag{42}$$

For C^2 continuity, we add the requirement that

$$\dot{d}_1(0) = \dot{d}_2(0). \tag{43}$$

7 Conclusions

This paper presents a methodology that enables to construct subdivision schemes for different applications. The common property of these subdivision schemes is that they are *piecewise-uniform.* They all consist of uniform subdivision rules, and special subdivision rules that are used near the boundaries between two regular regions.

Near the region boundaries, different subdivision rules meet and interact. For example, in the tri-quad scheme presented in § 4, Bicubics and quartic Box-splines meet along the y-axis. Our approach identifies the 'harmony' between the different subdivision rules with the property that the overall subdivision scheme reproduces polynomials (quadratics or cubics)—a crucial property for high order approximation, and for smoothness analysis.

We choose identity (11) as the formulation of polynomial reproduction, even though it requires to introduce the new operator Q, because it extends well from the uniform to the non-uniform case, and because we have the means (Corollary 2) to transform it into a system of linear equations for deriving the subdivision weights. As shown in [3], this theory also extends to the setting in which transfinite boundary conditions are involved in each subdivision iteration.

For smoothness analysis, we have presented the joint-spectral radius test for *quasi-uniform* schemes (see § 4), and the powerful theorem from [3] that reduces the analysis of subdivision under transfinite boundary conditions to the case of zero boundary conditions. Both analysis tools rely on polynomial reproduction.

References

1. Cavaretta, A.S., Dahmen, W., Micchelli, C.A.: Stationary Subdivision. Volume 453 of Memoirs of AMS. American Mathematical Society (1991)
2. Levin, A.: Polynomial generation and quasi-interpolation in stationary non-uniform subdivision. Computer Aided Geometric Design **20(1)** (2003) 41–60
3. Levin, A.: Combined Subdivision Schemes. PhD thesis, Tel-Aviv university (2000)
4. Levin, A., Levin, D.: Analysis of quasi-uniform subdivision. Applied and Computational Harmonic Analysis **15(1)** (2003) 18–32

5. Schaefer, S., Warren, J.: On c^2 triangle/quad subdivision. Accepted to ACM Transactions on Graphics (2004)
6. Stam, J., Loop, C.: Quad/triangle subdivision. Computer Graphics Forum **22(1)** (2003) 79–85
7. Hakenberg, J.: Smooth subdivision for mixed volumetric meshes. Master's thesis, Rice University, Department of Computer Science (2004)
8. Zulti, A., Levin, A., Levin, D., Teicher, M.: c^2 subdivision over triangulations with one extraordinary point. to appear in Computer Aided Geometric Design (2004)
9. Reif, U.: A unified approach to subdivision algorithms near extraordinary points. Computer Aided Geometric Design **12** (1995) 153–174
10. Warren, J., Weimer, H.: Subdivision Methods for Geometric Design. Morgan Kaufmann (2002)
11. Zorin, D.: Smoothness of stationary subdivision on irregular meshes. Preprint (1998)
12. Levin, A.: Interpolating nets of curves by smooth subdivision surfaces. In: Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series. (1999) 57–64
13. Levin, A.: Filling n-sided holes using combined subdivision schemes. In Laurent, P.J., Sablonniere, P., Schumaker, L.L., eds.: Curve And Surface Design, Vanderbilt University Press, Nashville, TN (1999) 221–228
14. Dyn, N.: Subdivision schemes in computer aided geometric design. Advances in Numerical Analysis II, Subdivision algorithms and radial functions, W.A. Light (ed.), Oxford University Press (1992) 36–104
15. Dyn, N., Greogory, J.A., Levin, D.: Piecewise uniform subdivision schemes. In Dahlen, M., Lyche, T., Schumaker, L., eds.: Mathematical Methods for Curves and Surfaces. Vanderbilt University Press, Nashville (1995) 111–120
16. Ron, A.: Wavelets and their associated operators. In Chui, C.K., Schumaker, L.L., eds.: Approximation Theory IX, Vanderbilt University Press, Nashville (1998) 283–317
17. deBoor, C.: Quasiinterpolants and approximation power of multivariate splines. In Gasca, M., Michelli, C.A., eds.: Computation of curves and surfaces. Dordrecht, Netherlands: Kluwer Academic Publishers (1990) 313–345
18. deBoor, C., Ron, A.: The exponentials in the span of the multiinteger translates of a compactly supported function: quasiinterpolation and approximation order. Journal of London Mathematical Society (2) **45** (1992) 519–535
19. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., W.Stuetzle: Piecewise smooth surface reconstruction. Computer graphics **28(3)** (1994) 295–302
20. Schweitzer, J.: Analysis and Applications of Subdivision Surfaces. PhD thesis, University of Washington, Seattle (1996)
21. Biermann, H., Levin, A., Zorin, D.: Piecewise smooth subdivision surfaces with normal control. In: Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series. (2000) 113–120
22. Levin, A.: Combined subdivision schemes for the design of surfaces satisfying boundary conditions. Computer Aided Geometric Design **16(5)** (1999) 345–354
23. Daubechies, I., Guskov, I., Sweldens, W.: Regularity of irregular subdivision. Constructive Approximation **15** (1999) 381–426
24. deBoor, C.: Cutting corners always works. Computer Aided Geometric Design **4** (1987) 125–131
25. Dyn, N., Greogory, J.A., Levin, D.: A four-point interpolatory subdivision scheme for curve design. Computer Aided Geometric Design **4** (1987) 257–268

26. Deslauriers, G., Dubuc, S.: Symmetric iterative interpolation. Constructive Approximation **5** (1989) 49–68
27. Catmull, E., Clark, J.: Recursively generated b-spline surfaces on arbitrary topological meshes. Computer Aided Design **10** (1978) 350–355
28. Loop, C.: Smooth spline surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics (1987)
29. Prautzsch, H., Umlauf, G.: A g^2 subdivision algorithm. In Farin, G., Bieri, H., Brunnet, G., DeRose, T., eds.: Geometric Modeling, Computing Supplements. New York: Springer-Verlag (1998) 217–224
30. Prautzsch, H., Reif, U.: Degree estimates of c^k piecewise polynomial subdivision surfaces. Advances in Computational Mathematics **10** (1999) 209–217
31. Prautzsch, H.: Freeform splines. Computer Aided Geometric Design **14** (1997) 201–206
32. Reif, U.: Turbs: Topologically unrestricted b-splines. Constructive Approximation **4** (1998) 55–77
33. Zulti, A., Levin, A., Levin, D., Taicher, M.: An electronic appendix to the paper: c^2 subdivision over triangulations with one extraordinary point. http://www.math.tau.ac.il/~levin/adi/zulti-levin_c2_sbd.htm (2004)
34. Dyn, N., Greogory, J.A., Levin, D.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Transactions on Graphics **9** (1990) 160–169

A Hybrid Approach to Extract Tooth Models from CT Volumes

Sheng-Hui Liao, Wei Han, Ruo-Feng Tong, and Jin-Xiang Dong

State Key Laboratory of CAD and CG, Department of Computer Science
and Engineering, Zhejiang University, China
{liaoshenhui, ivan, trf, djx}@cs.zju.edu.cn

Abstract. As the tooth root has similar bone density to the jaw where it is embedded, its complete boundaries are either missing or at low contrast in the computed tomography (CT) volume data. This paper proposes a hybrid method to create a 'best-fit' polygonal surface of the patient-specific tooth. First, a level-set based shape prior segmentation procedure is employed to extract a coarse whole tooth surface model from CT volume. The surface model produced captures the smooth root part, while losing details of the tooth crown. So, a post process - thin-plate splines transform, involving a consistent semi-automatic landmarks selection and re-placing procedure – is used to warp the crown part of the coarse surface to recover the patient-specific local details of the crown.

1 Introduction

Accurate 3-dimensional geometric crown models are now available from different sources, such as destructive scanning method, laser system, intra-oral camera, and so on. However there are applications being developed for modeling and treatment simulation that require additionally tooth root information. All the former methods provide data on the tooth crowns only and nothing on root form. Therefore, researchers have explored various methods to produce 3D whole tooth models.

Nishii and Terai et al. use root information from CT volumes with crown forms obtained using laser scanning [1, 2]. Registration of the data sets is accomplished using ceramic markers present in both images. For these methods, both patient-specific CT volume and patient-specific laser scanning crown data are needed.

In [3], Enciso et al. use a patient-specific 2D radiograph and a 3D geometric prior model to produce a patient-specific 3-dimensional model of the whole tooth using radial basis functions. But a 2D radiograph lacks depth information, and the warp procedure couldn't reach the 'best-fit' in some directions.

In this paper we use the patient-specific CT volume and a 3D geometric prior model to produce a 'best-fit' patient specific 3D surface model of the whole tooth in two steps: first, use a level-set based shape prior segmentation procedure to get a coarse tooth surface model, then recover the local details of the tooth crown using thin-plate splines warp.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 308–317, 2005.

The rationale for this approach is that, at present 3D CT imaging of dental patients is routine, but current widely used imaging devices are not specifically for dentistry and lack detail particular to the tooth root. Note that, 3D volumetric imaging devices developed specifically for dentistry are recently released (NewTom from Aperio Services) which feature a lower absorbed radiation dose. But these devices are too expensive, and won't be widely used in the very future. And also laser scanning isn't routine for dental patient inspection.

2 Create Coarse Surface Model Using Level-Set Based Shape Prior Segmentation

First, we must extract the 3D whole tooth model from the patient CT volume data. While as the tooth root is embedded in the jaw, the complete boundaries are either missing or are at low resolution and low contrast, such as Fig. 1 shows, those **all-intensity** based segmentation technologies do not work at all. So, the use of information about shape is indispensable.

There are many works on shape priors segmentation in the literature. For example, Leventon et al. 2000 [4], Chen et al. 2002 [5], Cremers et al. 2003 [6].

Our segmentation process is mainly based on Chen–Caselles's model.

In [7], by extending the traditional energy-based active contours (snakes), Caselles et al. developed an energy function $\mathbf{E(C)}$ over a curve $\mathbf{C}$, which reads:

$$\overset{\min}{C(q)} \int \mathrm{g}(|\nabla I(C(q))|) \left| C^{'}(q) \right| dq \tag{1}$$

where $I : \Omega \rightarrow R$ is an image defined on Ω, and g is a function of the image gradient, usually it is of the form:

$$\mathrm{g}(|\nabla \mathrm{I}|) = \frac{1}{1 + |\nabla \mathrm{I}|^2} \tag{2}$$

By minimizing the energy $\mathbf{E(C)}$, the curve $\mathbf{C}$ will be stable on the boundaries of objects in the image, where the gradients of intensity are very large.

In 2002, Chen et al. [5] developed a variational framework by incorporating shape priors into Caselles' model. They proposed a modified procrustes method to describe the shape priors. The method is as follows. Let C^* be a curve, called **shape prior**, representing the boundary of an object, and $\mathbf{C}$ be another curve, then the difference between curve C^* and $\mathbf{C}$, e.g., the two objects, is defined as:

$$\overset{\min}{C(q)} \int d^2(\mu R C(q) + T) \left| C^{'}(q) \right| dq \tag{3}$$

where μ, R and T are the parameters for scaling, rotation and translation, and $\mathrm{d(x,y) = d(C^*,(x,y))}$ is the distance from a point (x, y) to the curve C^*. Then, by combining these two terms, Chen et al. proposed the energy for shape prior segmentation as:

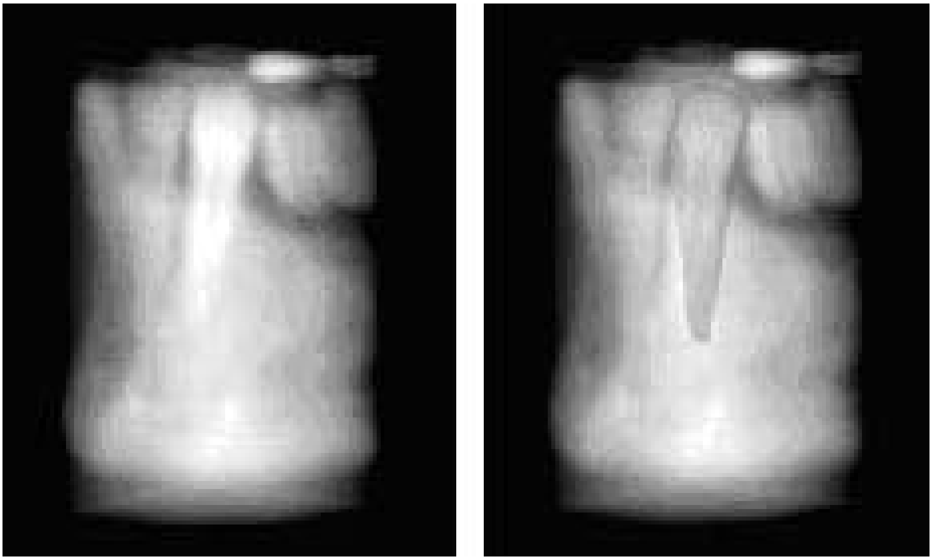


Fig. 1. Volume-rendering VOI and overlaid prior model

$$\mathrm{E(C,}\mu\mathrm{,R,T)} = \int \{\mathrm{g(}\,|\nabla \mathrm{I(C(q))}|\,\mathrm{)}+\frac{\lambda}{2}\mathrm{d}^2(\mu\mathrm{RC(q)+T)}\}\left|\mathrm{C}^{'}\mathrm{(q)}\right|\mathrm{dq} \tag{4}$$

where $\lambda > 0$ is a parameter. Moreover, they choose $\mathrm{g}(|\nabla I|)$ as:

$$\mathrm{g}(|\nabla I|) = \frac{1}{1+\beta\,|\nabla G_\sigma * I|} \tag{5}$$

where $\beta > 0$ is a parameter, and $G_\sigma(x) = \frac{1}{\sigma}e^{-|x|^2/4\sigma^2}$.

To reduce the computation time, we first overlay the volume-of-interest (VOI) in the CT volume and the 3D geometric prior model in close proximity, such as Fig. 1 shows.

Fig. 2 shows several time steps in the surface evolution procedure of segmentation and Fig. 3 shows the geometric polygonal surface of the whole tooth model extracted from the final segmentation result.

We note that the resultant surface model has an acceptable smooth root part, while losing local details of the tooth crown. The reason is that these shape prior segmentation approaches are extrinsic and depend to a large extent on the given shape prior model. It works well for the tooth root part, as it is smooth and bears much similarity to the shape. While the real tooth crown has many patient-specific local details, the coarse model produced is 'too smooth' and its shape is not well consistent with the patient CT volume data. So a post process must be applied to recover the local details of the tooth crown for the coarse surface model.

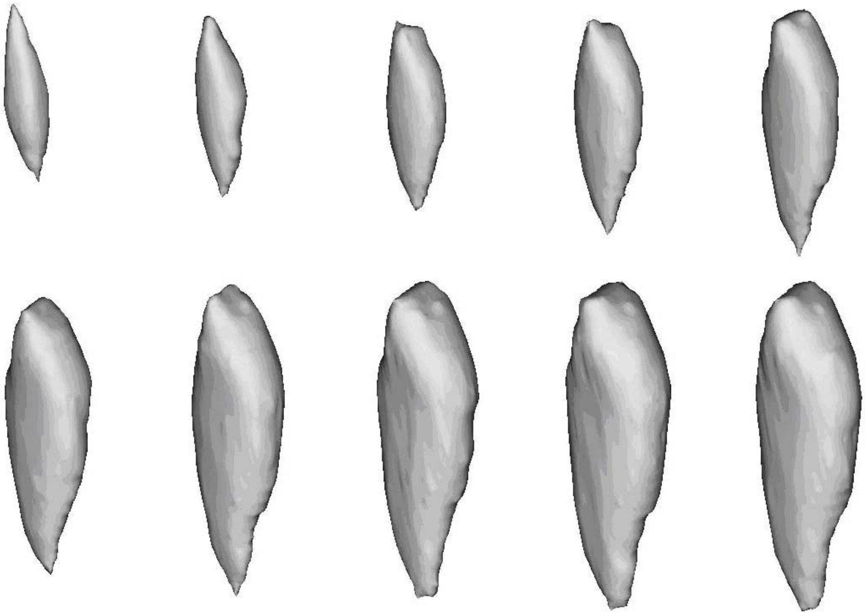


Fig. 2. Several time steps in the surface evolution procedure

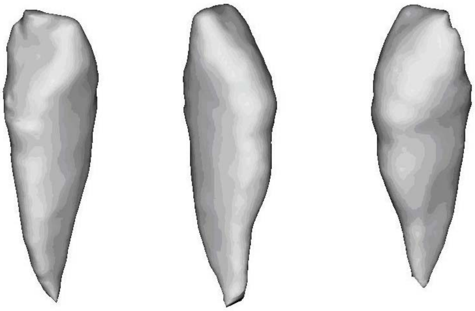


Fig. 3. Level-set based shape prior segmentation producing coarse surface model

3 Recover the Patient-Specific Details of the Tooth Crown by Thin-Plate Splines

As the tooth crown has a high contrast boundary feature in the CT volume, we use thin-plate splines to deform the crown part of the coarse surface model to match the patient CT volume information, producing the 'best-fit' polygonal surface of the whole tooth.

3.1 Background

Thin-plate splines (TPS) are widely used in morphometrics to define changes of shape between subjects of the same species [8, 9, 10], and used in scattered data interpolation [11]. The basic idea is interpolating specified points while minimizing an approximate curvature (integrated squared second derivative), resulting in a smooth deformation without unexpected ripples and variations:

$$\int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 + \left(\frac{\partial^2 f}{\partial z^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial z}\right)^2 + 2\left(\frac{\partial^2 f}{\partial y \partial z}\right)^2 dxdydz \tag{6}$$

Often, when the number of points to be interpolated is small, a simple radial basis function (RBF) is used for an analytical solution to the minimum of the functional (6).

The radial basis formulation of TPS is:

$$\delta(\mathbf{p}) = \sum_{k=1}^{n} c_k \left|\mathbf{p} - \mathbf{p}_k\right|^2 \log\left|\mathbf{p} - \mathbf{p}_k\right| + a_1 + (a_2, a_3)\mathbf{p} \tag{7}$$

where $\delta(\mathbf{p})$ is the desired displacement at a point $\mathbf{p}$, $\mathbf{p}_k$ are the n landmarks (each with a corresponding destination landmark $\mathbf{t}_k$), and a_j are coefficients of an affine registration (the measure (6) is invariant to the offset and linear slope of the optimized function so separate terms are added to handle this).

The weights c_k, a_j(k = 1 . . . n, j = 1 . . . 3) needed to interpolate the data can be found by solving the block matrix system

$$\begin{bmatrix} \kappa & \rho \\ \rho^{\mathbf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \delta \\ \mathbf{0} \end{bmatrix} \tag{8}$$

where $\kappa_{\mathbf{r},\mathbf{c}} = \left\|\mathbf{p}_\mathbf{r} - \mathbf{p}_\mathbf{c}\right\|^2 \log(\mathbf{p}_\mathbf{r} - \mathbf{p}_\mathbf{c})$, r, c $\in$ 1 . . . n; ρ is a $n \times 4$ matrix containing the constant (1) vector and the landmark location x_k, y_k and z_k; and δ are the data values to be interpolated. In a registration of 3D labeling application there are three such interpolations needed, one for each of x, y and z; but the matrix κ depends only on the data locations (not their values), so the matrix inverse is needed only once.

In practice it is desirable to 'regularize' the system (8) [12]. Regularization can be motivated both in terms of matrix conditioning and weight decay considerations, and by assuming that the landmarks have some error associated with them. In practice it involves adding an amount λI (= 0.001 for example) to the diagonal of κ before inverting. This has the effect of reducing large coefficients, thereby producing a much smoother warp.

3.2 Select Landmarks and Re-place Slidable Landmarks Consistently

First, extract 3D anatomical landmarks from the tooth crown's volume of interest, and geometric landmarks from the coarse polygonal surface model produced

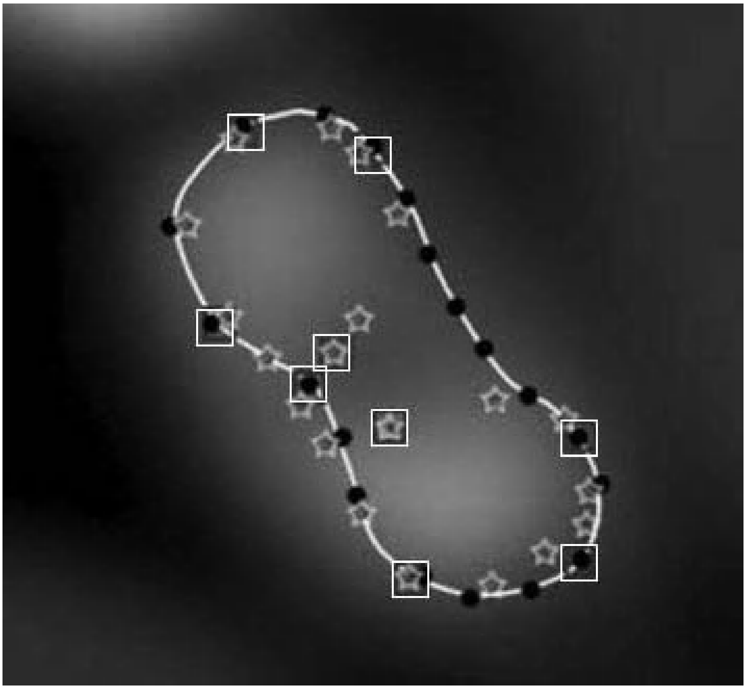


Fig. 4. Geometric cross contour and landmarks on Z-slice. These black ball points on the contour are geometric landmarks of surface model, and the grey star points are anatomical landmarks of VOI. Fixed landmarks are encapsulated by little rectangles

in section 2. The 3D operator *Op3* which is based on only first-order partial derivatives is employed to extract the anatomical landmarks of the CT volume [13]. For the 3D coarse polygonal surface model, we detect the curvature extrema and coordinate extreme points as the geometric landmarks.

As the result of automatic 3D landmark detection is not very satisfying and there are false detections especially in the anatomical landmarks, we select desired landmarks from the automatic detected candidates; at same time, select a list of Z-slices of the CT volume. Calculate the cross geometric contours of the crown part of coarse polygonal surface model, and overlay them on these slices respectively, such as the white contour showed in Fig. 4.

Then, on these Z-slices, create groups of geometric landmarks on the 2D contours, based on curvature extreme, curve length information and symmetry consideration; and indicate interactively with the mouse corresponding anatomical landmarks on the CT slices, as shown in Fig. 4.

We distinguish 'slidable' (contour-constrained) landmarks from unambiguous (fixed) landmarks. The latter may include corners (curvature extrema), extreme points (coordinate extrema), and symmetry points that are unambiguous in visual inspection. Contour-constrained landmarks are those that require a subjective judgment regarding where along the contour they are placed. Both the fixed anatomical landmarks and fixed geometry landmarks are indicated by little encapsulation rectangles in Fig. 4. Obvious, these selected landmarks from first step are all fixed landmarks on their respective Z-slices.

Although the placement of these fixed landmarks is fairly unambiguous, the placement of point landmarks along a contour is quite arbitrary. Unfortunately this arbitrary choice of landmark placement determines the quality of the resulting registration.

So, we adjust the placement of the contour-constrained landmarks along their respective contours so as to produce the smoothest thin-plate splines that interpolate the contours and fixed landmarks, which removes an undesired ambiguity in the case of TPS registration.

The basic idea is to minimize the squared (approximate) curvature (integral of second derivative squared) of the splines, the same criterion that TPS minimizes, which can be approximated by the sum square of the RBF coefficients $\sum c^T c \;=\; \mathbf{c}^T\mathbf{c}$ [14].

Assume that $\mathbf{p}$ is a particular landmark in the source image, $\mathbf{t_1}$ is the corresponding destination landmark, $\mathbf{v}$ is a unit vector through $\mathbf{t_1}$ toward the adjacent landmark on one side, and $\mathbf{t}$ is an undetermined point on this line. The objective of minimizing the coefficient energy while being constrained to the line $\mathbf{v}$ can be expressed as

$$\min_t \mathbf{c^T c} + \lambda((\mathbf{t} - \mathbf{t_1}) \cdot \mathbf{n}) \tag{9}$$

where $\mathbf{n}$ is the unit perpendicular vector (normal) to $\mathbf{v}$ and λ is a Lagrange multiplier enforcing the constraint.

Recall the RBF expression (2D form here) $\mathbf{t} - \mathbf{p} = \kappa\mathbf{c} + \rho\mathbf{a}$, we get $\mathbf{c} = \kappa^{-1}(\mathbf{t} - \mathbf{p} - \rho\mathbf{a})$, substituted into $\mathbf{\min_t} \mathbf{c^T c}$:

$$\min_t (\mathbf{t} - \mathbf{p} - \rho\mathbf{a})^{\mathbf{T}}(\kappa^{\mathbf{-1}})^{\mathbf{T}}\kappa^{\mathbf{-1}}(\mathbf{t} - \mathbf{p} - \rho\mathbf{a}) \tag{10}$$

Note that the derivative is zero at the minimum. We get the gradient with respect to $\mathbf{t}$:

$$\mathbf{2}(\kappa^{\mathbf{-1}})^{\mathbf{T}}\kappa^{\mathbf{-1}}\mathbf{t} - \mathbf{2}(\kappa^{\mathbf{-1}})^{\mathbf{T}}\kappa^{\mathbf{-1}}\mathbf{p} - \mathbf{2}(\kappa^{\mathbf{-1}})^{\mathbf{T}}\kappa^{\mathbf{-1}}\rho\mathbf{a} \tag{11}$$

The function of λ in Eq.(9) is simply to keep the new landmarks $\mathbf{t}$ on the original line segments, and this can also be achieved simply by moving $\mathbf{t}$ down its gradient (excluding the constraint) and then projecting it back on the contour.

Now, with the gradient defined, the landmark re-placement algorithm iteratively moves the contour-constrained landmarks a small distance down the gradient until the gradient approaches zero. In this procedure the landmarks slide along the piecewise linear contour defined by the original (unadjusted) landmarks.

Fig. 5 shows the effect of landmark re-placing procedure. In contrast to before landmark re-placing, these contour-constrained slidable anatomical landmarks (these star points) have moved along the piecewise linear contour defined by the original landmarks. And the resultant warped contour after landmark re-placing, which is indicated by the black stipple contour, is smooth and fit well to the CT data.

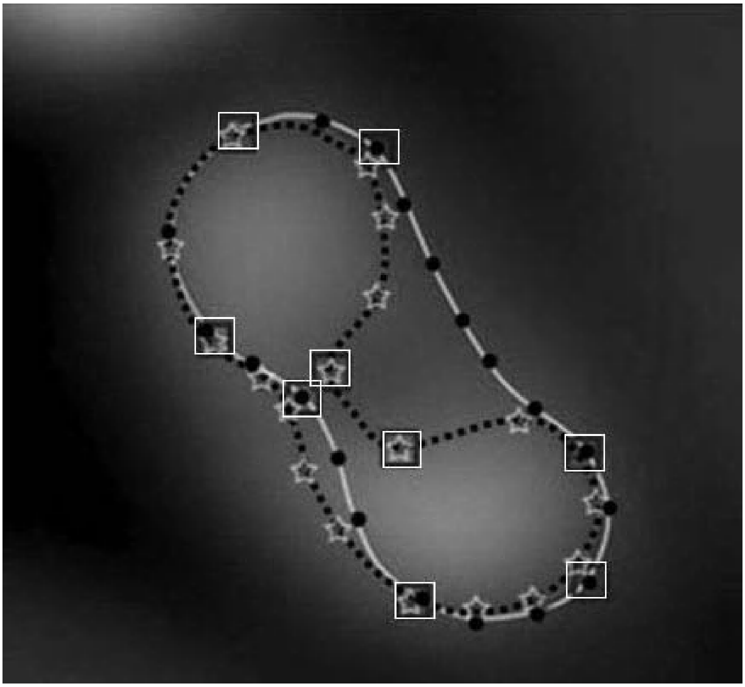


Fig. 5. Landmarks re-placing result. These star 'slidable' anatomical landmarks have moved to some extent compare to the original landmarks in Fig. 4, and the resultant warped contour is indicated by black stipple contour

One thing must be pointed out: as we use the same criterion of 2D TPS – minimize the squared (approximate) curvature of the splines – the result is consistent with 3D TPS transform in 3D space in the following procedure.

There is something that should be revised. The before-mentioned landmarks selection and re-placing procedure on 2D slices simplifies the landmark correspondence question and strengthens the robustness of TPS, but that does not work for some coordinate extreme landmarks - more precisely, for the Z coordinate extreme landmarks. In fact, those corresponding geometric and anatomical landmarks usually have different Z coordinates. The revising procedure is accomplished by manual interaction. Usually there are only a few of these cases.

3.3 Warp with Thin-Plate Splines

At last, our program first computes the thin-plate coefficients from these landmarks for the X, Y and Z (separately) displacements and then warps the crown part of the coarse polygonal surface model to recover the patient-specific local details of the crown.

Fig. 6 shows the final 'best-fit' polygonal surface model from different view angles. Note that compared to Fig. 3, there are more local details on the crown part.

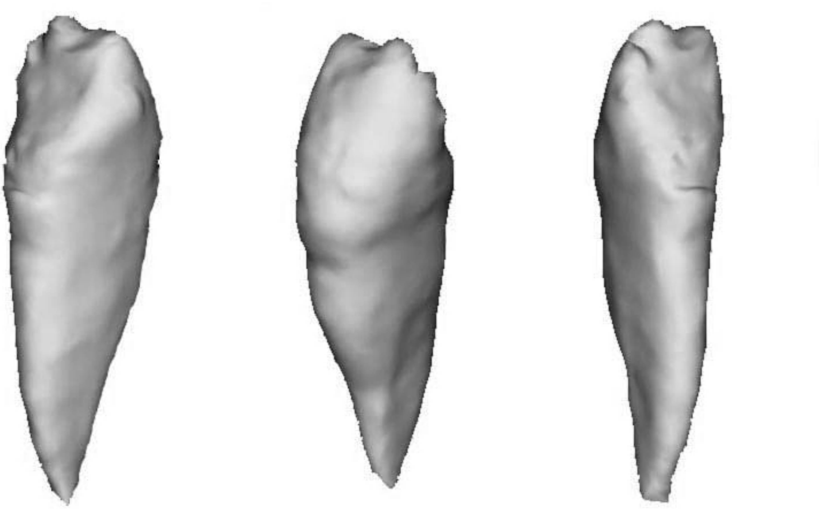


Fig. 6. Final ‘best-fit’ polygonal surface model

4 Conclusion

In this paper, we present an integrated whole tooth reconstruction approach by using mainly level-set based shape prior segmentation and thin-plate splines warp. According to the patient-specific CT volume and a 3D geometric prior model, the approach can generate a ‘best-fit’ model which is smooth on the whole tooth shape and conserves the local details on the crown part.

Acknowledgements. This project was supported by the Education Bureau (No.20030267) of Zhejiang Province, China, and the Natural Science Foundation (No.M603129) of Zhejiang Province, China.

References

1. Nishii Y, Nojima K, Takane Y, Isshiki Y, Integration of the maxillofacial three-dimensional CT image and the three-dimensional dental surface image, *The Journal of Japan Orthodontic Society*, 57, 189–194, 1998.
2. Terai H, Shimahara M, Sakinaka Y, Tajima S, Accuracy of Integration of Dental Casts in Three-Dimensional Models, *J Oral Maxillofacial Surgery*, 57, 662–665, 1999.
3. R. Enciso, John P. Lewis, U. Neumann, and J. Mah, 3D Tooth Shape from Radiographs using Thin-Plate Splines, *MMVR11 - NextMed: Health Horizon, The 11th Annual Medicine Meets Virtual Reality Conference*, Newport Beach, California, 22–25, 2003.
4. M.E. Leventon, W.E. Grimson and O. Faugeraus, Statistical shape influence in geodesic active contours. *Proc. IEEE Conf. Computer Vision and Patt. Recog.*, 316–323.

5. Y. Chen, H. Tagare, S. Thiruvenkadam, F. Huang, D. Wilson, K. Gopinath, R. Briggs and E. Geiser, Using prior shapes in geometric active contours in a variational framework, *International Journal of Computer Vision* 50 (3), 315–328, 2002.
6. D. Cremers, N. Sochen, and C. Schnorr, Towards recognition-based variational segmentation using shape priors and dynamic labeling, In L. Griffith, editor, *Int. Conf. on Scale Space Theories in Computer Vision* volume 2695 of LNCS, 388–400, 2003. Springer.
7. V.Caselles, R. Kimmel and G. Sapiro, Geodesic active contours, *International Journal of Computer Vision*, 22 (1), 61–79, 1997.
8. Bookstein FL, Principal warps: Thin-plate splines and the decomposition of deformations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, 567–585, 1989.
9. Carr JC, Fright WR, Beatson RK, Surface interpolation with radial basis functions for medical imaging, *IEEE Transactions on Medical Imaging* 16, 96–107, 1997.
10. Fang S, Raghavan R, Richtsmeier JT, Volume Morphing Methods for Landmark Based 3D Image Deformation, In: *SPIE International Symposium on Medical Imaging*, Newport Beach, CA., 2710, 404–415, 1996.
11. R.Franke and G.M.Nielson, Scattered Data Interpolation and Applications: A Tutorial and Survey, *Geometric Modelling: Methods and Their Application*, , H. Hagen and D. Roller, eds., 131–160. Berlin, Springer-Verlag, 1991.
12. K. Rohr, H. S. Stiehl, R. Sprengel, W. Beil, T. M. Buzug, J. Weese, and M. H. Kuhn, Point-based elastic registration of medical image data using approximating thin-plate splines. In: *Visualization in Biomedical Computing (VBC'96)*, 297–306, 1996.
13. T. Hartkens, K. Rohr, and H.S. Stiehl, Evaluation of 3D Operators for the Detection of Anatomical Point Landmarks in MR and CT Images, *Computer Vision and Image Understanding*, 86 (2), 118–136, 2002.
14. J. Lewis, H.-J. Hwang, U. Neumann, and R. Enciso, Smart Point Landmark Distribution for Thin-Plate Splines, In: *Proc. SPIE Medical Imaging* (San Diego), 1236–1243, 2002.

Bézier Surfaces of Minimal Internal Energy

Yongwei Miao[1,2], Huahao Shou[2], Jieqing Feng[1],
Qunsheng Peng[1], and A. Robin Forrest[3]

[1] State Key Lab. of CAD&CG, Zhejiang University,
Hangzhou 310027, P.R.China
[2] College of Science, Zhejiang University of Technology,
Hangzhou 310032, P.R.China
[3] School of Computing Sciences, University of East Anglia,
Norwich, NR4 7TJ, U.K
{miaoyw, jqfeng, peng}@cad.zju.edu.cn

Abstract. In this paper the variational problems of finding Bézier surfaces that minimize the bending energy functional with prescribed border for both cases of triangular and rectangular are investigated. As a result, two new bending energy masks for finding Bézier surfaces of minimal bending energy for both triangular and rectangular cases are proposed. Experimental comparisons of these two new bending energy masks with existing Dirichlet, Laplacian, harmonic and average masks are performed which show that bending energy masks are among the best.

1 Introduction

For parametric curves and surfaces, the various techniques of variational design are widely used in geometric modeling [2, 5, 8, 9, 14, 15, 16, 17]. Variational method is to optimize the shape of surface patches by minimizing a given functional that represents the energy of the surface [5, 8, 15, 16]. A well known objective functional is called surface internal energy. In the CAGD literature, the surface internal energy is defined as a linear combination of stretching energy and bending energy. Because of its physical background, which represents surface tension and rigidity, the internal energy for surface design was studied extensively [8, 9, 14, 17].

The stretching energy is also called the Dirichlet functional in the mathematical literature. It is a linear substitute for the area functional. The problem of finding a surface which minimizes the area functional with prescribed border has been studied by many researchers [1, 3, 4, 7, 10, 11]. The first non trivial example of polynomial surface with minimal area is the Enneper's surface [3, 7]. The problem of finding a surface with minimal area among all Bézier surfaces with prescribed border is called Plateau-Bézier problem [3, 10, 11]. There are some methods to find approximate solutions. The extremals of a Dirichlet functional can be obtained by solving linear systems. In [5], Farin and Hansford proposed a control net generation scheme for a Bézier surface by using a mask derived from the discretization of the Laplacian operator. In [10, 11], Monterde proposed two

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 318–335, 2005.

new masks related to the Plateau-Bézier problem, i.e. the Dirichlet mask and the Laplacian mask. He also compared the results of several masks and Dirichlet extremals for several different configurations of the boundary conditions. In [1], Arnal et al. studied the triangular Plateau-Bézier problem.

The bending energy is an important part of internal energy, which can be applied in surface and mesh fairing [12, 13, 18]. The standard approach for fair surfaces and meshes is based on the idea of minimizing a certain fairness metric, punishing features that are inconsistent with the fairness principle of the simplest shape. In interactive surface modeling, bending energy functional, also called as thin plate energy, is always chosen as a measure for the fairness of a surface and mesh. Unfortunately, because the bending energy is rather difficult to handle, it is usually approximated by its discrete version [13, 18]. We will discuss the accurate answers to the problem in the case of Bézier surfaces.

The problem of finding a surface which minimizes the bending energy functional with prescribed border is studied in this article. For triangular or rectangular Bézier surface, the prescribed border is determined by the border control points. Thus the problem can be formulated as: Given the exterior control points, find the interior ones such that the resulting Bézier surface has minimal bending energy among all Bézier surfaces with the same border. In this paper variational problems for both the triangular and rectangular surfaces are discussed, and corresponding masks related with surfaces of minimal energy are given.

2 The Bending Energy Functional

A triangular Bézier surface is defined as weighted sum of its control points. Let $\{P_{i,j,k}\}_{i+j+k=n}$ be the control points of a triangular Bézier surface of degree n, the Bernstein polynomials of degree n over a non-degenerate triangle $V_1V_2V_3$ are defined by

$$B^n_{i,j,k}(u,v,w) = \frac{n!}{i!j!k!}u^i v^j w^k \quad i+j+k=n,\ i,j,k \in N$$

where u,v,w are barycentric coordinates with respect to the triangule domain $V_1V_2V_3$. Then the parametric triangular Bézier surface is defined as:

$$P(u,v,w) = \sum_{i+j+k=n} P_{i,j,k}B^n_{i,j,k}(u,v,w) \quad u+v+w=1, u \geq 0, v \geq 0, w \geq 0$$

The triangular Bézier surface can also be represented as the bivariate form:

$$P(u,v) = \sum_{i+j=0}^{n} P_{i,j,n-i-j}B^n_{i,j,n-i-j}(u,v,1-u-v)$$

If we denote

$$P_{i,j} = P_{i,j,n-i-j}, B^n_{i,j}(u,v) = B^n_{i,j,n-i-j}(u,v,1-u-v)$$

where $B^n_{i,j}(u,v) = \begin{pmatrix} n \\ i,j \end{pmatrix} u^i v^j (1-u-v)^{n-i-j}$ and $\begin{pmatrix} n \\ i,j \end{pmatrix} = \frac{n!}{i!j!(n-i-j)!}$.

Then

$$P(u,v)=\sum_{i+j=0}^{n} P_{i,j}B_{i,j}^{n}(u,v)$$

is defined on the triangle region: $\triangle=\{(u,v)\in R^2 : u\geq 0, v\geq 0, u+v\leq 1\}$.

To simplify notation and the sequence of deduction, we shall make use of the following operators:

- the invariant operator $I : IP_{i,j}=P_{i,j}$;
- the shift operators $E_i : E_1P_{i,j}=P_{i+1,j}, E_2P_{i,j}=P_{i,j+1}$;
- the difference operators

$$\Delta^{1,0}P_{i,j}=\Delta_1 P_{i,j}=(E_1-I)P_{i,j}=P_{i+1,j}-P_{i,j},$$

$$\Delta^{0,1}P_{i,j}=\Delta_2 P_{i,j}=(E_2-I)P_{i,j}=P_{i,j+1}-P_{i,j}.$$

Then the triangular Bézier surface can be represented as:

$$P(u,v)=(I+u\Delta_1+v\Delta_2)^n P_{0,0}$$

and the derivatives of triangular Bézier surface can be represented as:

$$P_u(u,v)=n\sum_{i+j=0}^{n-1}\Delta^{1,0}P_{i,j}B_{i,j}^{n-1}(u,v)$$

$$P_v(u,v)=n\sum_{i+j=0}^{n-1}\Delta^{0,1}P_{i,j}B_{i,j}^{n-1}(u,v)$$

A rectangular Bézier surface of degree $n\times m$ can be represented as:

$$P(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}B_i^n(u)B_j^m(v)$$

which defined on the rectangular region: $\Omega=\{(u,v)\in R^2 : 0\leq u,v\leq 1\}$, where: $B_i^n(u)=\frac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$.

For the sake of simplification, we use the difference operators, then:

$$P(u,v)=(I+u\Delta_1)^n(I+v\Delta_2)^m P_{0,0}$$

and the derivatives of rectangular Bézier surface can be represented as:

$$P_u(u,v)=n\sum_{i=0}^{n-1}\sum_{j=0}^{m}\Delta^{1,0}P_{i,j}B_i^{n-1}(u)B_j^m(v)$$

$$P_v(u,v)=m\sum_{i=0}^{n}\sum_{j=0}^{m-1}\Delta^{0,1}P_{i,j}B_i^n(u)B_j^{m-1}(v)$$

In the CAGD literature, the internal energy of a surface $P(u,v)$ is the energy that depends only on definition of the surface. In [14], the following functionals are defined for stretching energy $E_{stretch}$ and bending energy E_{bend}:

$$E_{stretch}(P) = \int_{\Omega} (\|P_u\|^2 + \|P_v\|^2) dudv$$

$$E_{bend}(P) = \int_{\Omega} (\|P_{uu}\|^2 + 2\|P_{uv}\|^2 + \|P_{vv}\|^2) dudv$$

They depend only on intrinsic property of the surface. The internal energy $E_{internal}$ is a linear combination of $E_{stretch}$ and E_{bend}.The associated Euler-Largrange equations of $E_{stretch}$ and E_{bend} are harmonic equation: $\Delta P = 0$ and multi-harmonic equation: $\Delta\Delta P = 0$ respectively.

3 Extremals of the Energy Functional for Triangular Bézier Surface

3.1 Extremals of the Bending Energy Functional

For triangular Bézier surface, the bending energy E_{bend} is bounded from below because it is defined as integrals of bounded positive functions. Moreover, the bending energy depends on $\frac{(n-1)(n-2)}{2}$ inner control points, each point has three coordinates. Thus for triangular Bézier surface, the bending energy can be considered as a functional defined on $R^{\frac{3(n-1)(n-2)}{2}}$. The functional will reach its extremals when its gradient vanishes.

Proposition 3.1. The control points $\{P_{i,j}\}_{i+j=0}^{n}$ is the extremal of the bending energy functional with the prescribed border if and only if

$$\sum_{k+l=0}^{n-2} \frac{\binom{n-2}{k,l}}{\binom{2n-4}{i+k,j+l}} \Big(A_{i,j}^{k,l} \Delta^{2,0} P_{k,l} + 2B_{i,j}^{k,l} \Delta^{1,1} P_{k,l} + A_{j,i}^{l,k} \Delta^{0,2} P_{k,l} \Big) = 0 \quad (1)$$

for any $i > 0, j > 0$ and $i + j < n$ where the coefficient

$$A_{i,j}^{k,l} = \frac{i(2n-i-j-k-l-3)(i^2+ij+ik-il-3i+2jk-2kn-j+k+l+2)}{(n-i-j)(n-i-j-1)(i+k)(i+k-1)} + 1$$

$$B_{i,j}^{k,l} = \frac{(2n-i-j-k-l-3)(i^2j+i^2l+ij^2-iln-2ij+j^2k-jkn)}{(n-i-j)(n-i-j-1)(i+k)(j+l)} + 1$$

Proof. For any $a \in \{1, 2, 3\}$:

$$\frac{\partial E(p)}{\partial x_{ij}^a} = 2 \int_{\triangle} \Big(\langle \frac{\partial P_{uu}}{\partial x_{ij}^a}, P_{uu} \rangle + 2 \langle \frac{\partial P_{uv}}{\partial x_{ij}^a}, P_{uv} \rangle + \langle \frac{\partial P_{vv}}{\partial x_{ij}^a}, P_{vv} \rangle \Big) dudv$$

For the sake of simplification, we use the difference operators and compute the second order partial derivatives:

$$P_{uu}(u,v) = n(n-1)\sum_{i+j=0}^{n-2} \varDelta^{2,0} P_{i,j} B_{i,j}^{n-2}(u,v)$$

$$P_{uv}(u,v) = n(n-1)\sum_{i+j=0}^{n-2} \varDelta^{1,1} P_{i,j} B_{i,j}^{n-2}(u,v)$$

$$P_{vv}(u,v) = n(n-1)\sum_{i+j=0}^{n-2} \varDelta^{0,2} P_{i,j} B_{i,j}^{n-2}(u,v)$$

Then, we get:

$$\begin{aligned}\frac{\partial P_{uu}}{\partial x_{ij}^a} &= \frac{\partial^2}{\partial u^2}\Big(\frac{\partial P}{\partial x_{ij}^a}\Big) = \frac{\partial^2}{\partial u^2}\Big(\frac{\partial}{\partial x_{ij}^a}\sum_{k+l=0}^{n} P_{k,l}B_{k,l}^{n}(u,v)\Big) = \frac{\partial^2}{\partial u^2}\Big(B_{i,j}^{n}(u,v)\Big)e^a \\ &= n(n-1)\Big(B_{i-2,j}^{n-2}(u,v) - 2B_{i-1,j}^{n-2}(u,v) + B_{i,j}^{n-2}(u,v)\Big)e^a\end{aligned}$$

where $e^a (a \in \{1,2,3\})$ denotes the ath vector of the canonical basis. Similarly, we can get the following partial derivatives:

$$\begin{aligned}\frac{\partial P_{uv}}{\partial x_{ij}^a} &= n(n-1)\Big(B_{i-1,j-1}^{n-2}(u,v) - B_{i-1,j}^{n-2}(u,v) - B_{i,j-1}^{n-2}(u,v) + B_{i,j}^{n-2}(u,v)\Big)e^a \\ \frac{\partial P_{vv}}{\partial x_{ij}^a} &= n(n-1)\Big(B_{i,j-2}^{n-2}(u,v) - 2B_{i,j-1}^{n-2}(u,v) + B_{i,j}^{n-2}(u,v)\Big)e^a\end{aligned}$$

Therefore, the differential of the bending energy functional:

$$\begin{aligned}\frac{\partial E(p)}{\partial x_{ij}^a} = 2n(n-1)\Bigg[\int_{\triangle}\Big(&\big(B_{i-2,j}^{n-2}(u,v) - 2B_{i-1,j}^{n-2}(u,v) + B_{i,j}^{n-2}(u,v)\big)\langle e^a, P_{uu}\rangle \\ &+ 2\big(B_{i-1,j-1}^{n-2}(u,v) - B_{i-1,j}^{n-2}(u,v) - B_{i,j-1}^{n-2}(u,v) + B_{i,j}^{n-2}(u,v)\big)\langle e^a, P_{uv}\rangle \\ &+ \big(B_{i,j-2}^{n-2}(u,v) - 2B_{i,j-1}^{n-2}(u,v) + B_{i,j}^{n-2}(u,v)\big)\langle e^a, P_{vv}\rangle\Big)dudv\Bigg]\end{aligned}$$

Apply the following identity of Bernstein Polynomials and integral equation:

$$B_{i,j}^{m}(u,v)B_{k,l}^{n}(u,v) = \frac{\binom{m}{i,j}\binom{n}{k,l}}{\binom{m+n}{i+k,j+l}}B_{i+k,j+l}^{m+n}(u,v)$$

$$\int_{\triangle} B_{i,j}^{n}(u,v)dudv = \frac{1}{(n+1)(n+2)}$$

We get the gradient of bending energy functional $\frac{\partial E(p)}{\partial x_{ij}^a}$, which is:

$$
\begin{aligned}
&K\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i-2,j}}{\binom{2n-4}{i+k-2,j+l}}-2\frac{\binom{n-2}{i-1,j}}{\binom{2n-4}{i+k-1,j+l}}+\frac{\binom{n-2}{i,j}}{\binom{2n-4}{i+k,j+l}}\Big]\binom{n-2}{k,l}I_{k,l}^{2,0}\\
&+2K\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i-1,j-1}}{\binom{2n-4}{i+k-1,j+l-1}}-\frac{\binom{n-2}{i-1,j}}{\binom{2n-4}{i+k-1,j+l}}\Big]\binom{n-2}{k,l}I_{k,l}^{1,1}\\
&-2K\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i,j-1}}{\binom{2n-4}{i+k,j+l-1}}-\frac{\binom{n-2}{i,j}}{\binom{2n-4}{i+k,j+l}}\Big]\binom{n-2}{k,l}I_{k,l}^{1,1}\\
&+K\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i,j-2}}{\binom{2n-4}{i+k,j+l-2}}-2\frac{\binom{n-2}{i,j-1}}{\binom{2n-4}{i+k,j+l-1}}+\frac{\binom{n-2}{i,j}}{\binom{2n-4}{i+k,j+l}}\Big]\binom{n-2}{k,l}I_{k,l}^{0,2}
\end{aligned}
$$

where the constant $K=\frac{2n^2(n-1)^2}{(2n-3)(2n-2)}$, and the notation $I_{k,l}^{p,q}=\left\langle e^a, \Delta^{p,q}P_{k,l}\right\rangle$.

So, the condition for the extremals of bending energy functional is $\frac{\partial E(p)}{\partial x_{ij}^a}=0$. By expanding $\frac{\partial E(p)}{\partial x_{ij}^a}$, we have:

$$
\begin{aligned}
0=&\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i-2,j}}{\binom{2n-4}{i+k-2,j+l}}-2\frac{\binom{n-2}{i-1,j}}{\binom{2n-4}{i+k-1,j+l}}+\frac{\binom{n-2}{i,j}}{\binom{2n-4}{i+k,j+l}}\Big]\binom{n-2}{k,l}\Delta^{2,0}P_{k,l}\\
&+2\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i-1,j-1}}{\binom{2n-4}{i+k-1,j+l-1}}-\frac{\binom{n-2}{i-1,j}}{\binom{2n-4}{i+k-1,j+l}}\Big]\binom{n-2}{k,l}\Delta^{1,1}P_{k,l}\\
&-2\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i,j-1}}{\binom{2n-4}{i+k,j+l-1}}-\frac{\binom{n-2}{i,j}}{\binom{2n-4}{i+k,j+l}}\Big]\binom{n-2}{k,l}\Delta^{1,1}P_{k,l}\\
&+\sum_{k+l=0}^{n-2}\Big[\frac{\binom{n-2}{i,j-2}}{\binom{2n-4}{i+k,j+l-2}}-2\frac{\binom{n-2}{i,j-1}}{\binom{2n-4}{i+k,j+l-1}}+\frac{\binom{n-2}{i,j}}{\binom{2n-4}{i+k,j+l}}\Big]\binom{n-2}{k,l}\Delta^{0,2}P_{k,l}
\end{aligned}
$$

If denote:

$$A_{i,j}^{k,l}=\frac{i(2n-i-j-k-l-3)(i^2+ij+ik-il-3i+2jk-2kn-j+k+l+2)}{(n-i-j)(n-i-j-1)(i+k)(i+k-1)}+1$$

$$B_{i,j}^{k,l}=\frac{(2n-i-j-k-l-3)(i^2j+i^2l+ij^2-iln-2ij+j^2k-jkn)}{(n-i-j)(n-i-j-1)(i+k)(j+l)}+1$$

We can prove the equations (1).

3.2 Example of $n=3$

The cubic triangular Bézier surface has only one inner control point $P_{1,1,1}$:

Proposition 3.2. The cubic triangular Bézier surface is an extremal of the bending energy functional with the prescribed border if and only if:

$$P_{1,1,1} = \frac{1}{12}\Big(2P_{0,0,3}+3P_{1,0,2}+3P_{0,1,2}-3P_{0,2,1}-3P_{2,0,1}+P_{0,3,0}+4P_{1,2,0}+4P_{2,1,0}+P_{3,0,0}\Big)$$

3.3 Bending Energy Mask

If the boundary curves of a Bézier surface are prescribed, then the boundary control points are fixed. Therefore, the problem of constructing a Bézier surface is equivalent to compute the inner control points. A simple way to solve the problem is through a mask.

From the condition in Proposition 3.1 and Proposition 3.2 we can try to solve a linear system to get minimal bending energy triangular Bézier surface according to the given exterior control points. The equations are:

$$\begin{aligned} 12P_{i,j,k} = {} & 2\ P_{i+2,j-1,k-1} + 3P_{i+1,j,k-1} + 3P_{i+1,j-1,k} - 3P_{i,j-1,k+1} - 3P_{i,j+1,k-1} \\ & + P_{i-1,j-1,k+2} + 4P_{i-1,j,k+1} + 4P_{i-1,j+1,k} + P_{i-1,j+2,k-1} \end{aligned} \qquad (2)$$

These equations can be expressed using the asymmetric *bending energy mask* shown in Fig. 1.

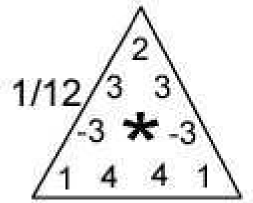


Fig. 1. Bending energy mask

3.4 Solvability for Linear System

The above linear system (2) tell us that every interior control point can be represented as a linear combination of its triangular neighbour control points, that is: $P_i = \sum_{j\in N_i} \lambda_{ij} P_j$, where N_i denotes 9 neighbour control points for interior control point P_i.

In order to study the linear system, we introduce some graph theory terminologies. All control points and corresponding neighbour control points can define a direct graph G as following: $V(G)$ denotes all control points, and $E(G)$ denotes all ordered control point pairs (P_i, P_j) such that P_j is a neighbour control point of P_i. As shown in Fig. 2, for every interior control point P_i, we can easily find a directed path to one boundary control point P_j, i.e. a directed path $P_i = P^1, P^2, P^3, \cdots, P^m = P_j$ such that (P^k, P^{k+1}) is an ordered control point pair for $k = 1, 2, 3, \cdots, m-1$. So all interior control points in the direct graph G are boundary connected, then we can conclude that the linear system (2) has a unique solution according to the Floater and Reimers' sufficient condition in reference [6].

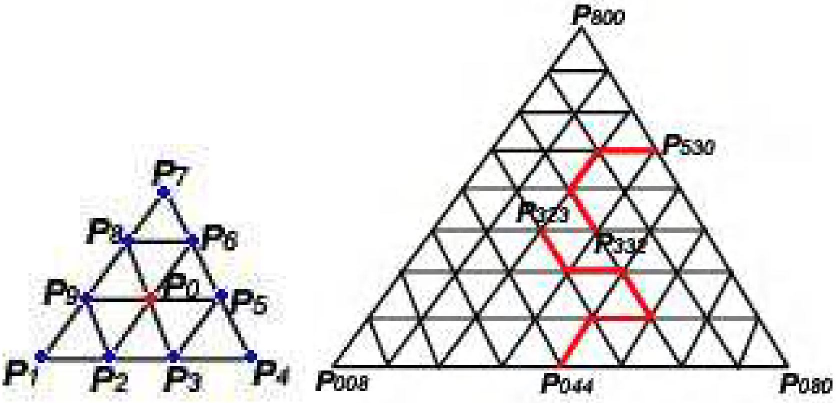


Fig. 2. The triangular neighbour control points and directed path (solid line) from interior control point to one boundary control point

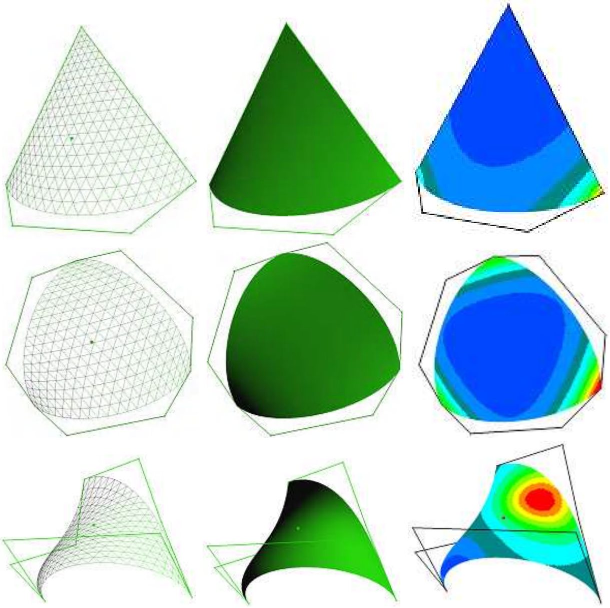


Fig. 3. The triangular Bézier surfaces associated to different boundary curves with bending energy mask. Left column, the wireframe version of different surfaces. Center column, the render version of different surfaces. Right column, the curvature version of different surfaces. Different grayscales on the curvature version are related to different Gaussian curvatures

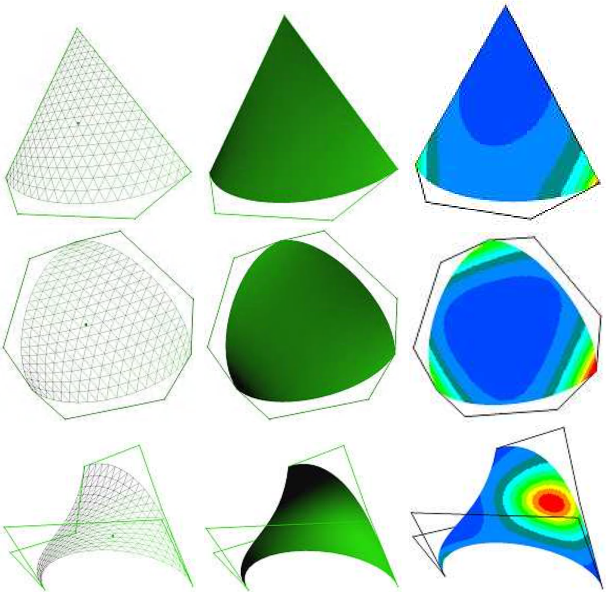


Fig. 4. The triangular Bézier surfaces associated to different boundary curves with Dirichlet mask

3.5 Examples

The reference [7] assert that whether there is better mask depends on the boundary conditions. In this section, we compare some examples for the cubic triangular Bézier surface case.

We first fix the three boundary curves with its control points, then construct the surface by computing the inner control point through average mask, the Laplacian mask, the Dirichlet mask and bending energy mask. Fig. 3 shows the different borders and the triangular Bézier surfaces constructed by means of the *bending energy mask*. Fig. 4 shows the corresponding triangular Bézier surfaces constructed by means of the *Dirichlet mask*. Fig. 5 shows the corresponding triangular Bézier surfaces constructed by means of the *Laplacian mask*.

Table 1. Comparison among different masks

Masks	Example 1		Example 2		Example 3	
	Stretching	Bending	Stretching	Bending	Stretching	Bending
Average Mask	2.4255	4.3627	3.2814	4.1667	1.3811	11.3149
Laplacian Mask	2.4309	4.5192	3.3004	4.5121	1.3916	10.8962
Dirichlet Mask	2.4158	4.3575	3.2625	5.6250	1.3733	12.9075
Bending Energy Mask	2.4224	3.7650	3.2791	4.1250	1.3963	10.8350

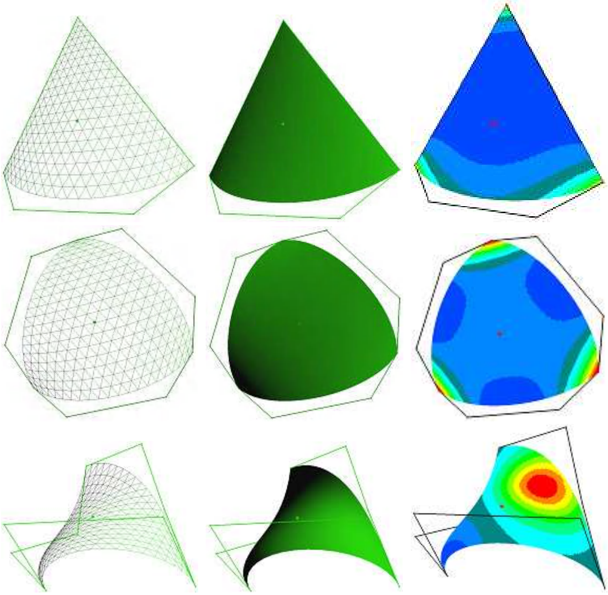


Fig. 5. The triangular Bézier surfaces associated to different boundary curves with Laplacian mask

In Table 1, the internal energy of the triangular Bézier surfaces are shown. From Table 1, we can assert that the internal energy of the triangular Bézier surfaces depends heavily on the boundary curve conditions. In three examples associated to different boundary curves with average, Laplacian, Dirichlet and bending energy mask, the bending energy mask is always the best one, i.e. the minimal internal energy.

4 Extremals of the Energy Functional for Rectangular Bézier Surface

4.1 Extremals of the Bending Energy Functional

For rectangular Bézier surface, the bending energy E_{bend} is bounded from below because it is defined as integrals of bounded positive functions. Moreover, the bending energy depends on $(n-1)\times(m-1)$ inner control points, each point has three coordinates. Thus, the bending energy for rectangular Bézier surface can be considered as a functional defined on $R^{3(n-1)(m-1)}$. We can compute the inner control points by setting the gradient of a real function as zero.

Proposition 4.1. The control points $\{P_{i,j}\}_{i,j=0}^{n,m}$ is the extremal of the bending energy functional with the prescribed border if and only if

$$
\begin{aligned}
0 = {} & \frac{n^2(n-1)^2}{(2n-3)(2m+1)}\binom{n-2}{i}\binom{m}{j}\sum_{k,l=0}^{n-2,m}\frac{\binom{n-2}{k}\binom{m}{l}}{\binom{2n-4}{i+k-2}\binom{2m}{j+l}}A_{n,i}^{k}\Delta^{2,0}P_{k,l} \\
& + \frac{2n^2m^2}{(2n-1)(2m-1)}\binom{n-1}{i}\binom{m-1}{j}\sum_{k,l=0}^{n-1,m-1}\frac{\binom{n-1}{k}\binom{m-1}{l}}{\binom{2n-2}{i+k-1}\binom{2m-2}{j+l-1}}B_{n,i}^{k}B_{m,j}^{l}\Delta^{1,1}P_{k,l} \\
& + \frac{m^2(m-1)^2}{(2n+1)(2m-3)}\binom{n}{i}\binom{m-2}{j}\sum_{k,l=0}^{n,m-2}\frac{\binom{n}{k}\binom{m-2}{l}}{\binom{2n}{i+k}\binom{2m-4}{j+l-2}}A_{m,j}^{l}\Delta^{0,2}P_{k,l} \qquad (3)
\end{aligned}
$$

for any $i \in \{1, 2, \cdots, n-1\}$ and $j \in \{1, 2, \cdots, m-1\}$ where the coefficient

$$
\begin{aligned}
A_{n,i}^{k} &= \frac{nk(n-1)(k-1) + i^2(n-2)(n-3) - i(n-3)(2kn+n-2k-2)}{(n-i)(n-i-1)(2n-i-k-2)(2n-i-k-3)} \\
B_{n,i}^{k} &= \frac{ni-nk-i}{(n-i)(2n-k-i-1)}
\end{aligned}
$$

Remark. The proof of this proposition will be given in the Appendix.

4.2 Example of $n = m = 2$

The biquadratic rectangular Bézier surface has only one inner control point $P_{1,1}$:

Proposition 4.2. The biquadratic rectangular Bézier surface is an extremal of the bending energy functional with the prescribed border if and only if:

$$
P_{1,1} = \frac{1}{44}\Big(4P_{0,0} + 7P_{0,1} + 4P_{0,2} + 7P_{1,0} + 7P_{1,2} + 4P_{2,0} + 7P_{2,1} + 4P_{2,2}\Big)
$$

4.3 Example of $n = m = 3$

The bicubic rectangular Bézier surface has four inner control points $P_{1,1}, P_{1,2}$, $P_{2,1}, P_{2,2}$. There are four equations corresponding to four inner control points:

Proposition 4.3. The bicubic rectangular Bézier surface is an extremal of the bending energy functional with the prescribed border if and only if:

$$
P_{1,1} = \frac{1}{48807}\begin{pmatrix} 6746P_{0,0} + 2910P_{0,1} + 16572P_{0,2} - 5336P_{0,3} + 2910P_{1,0} + 5352P_{1,3} + \\ 16572P_{2,0} + 1788P_{2,3} - 5336P_{3,0} + 5352P_{3,1} + 1788P_{3,2} - 529P_{3,3} \end{pmatrix}
$$

$$
P_{1,2} = \frac{1}{48807}\begin{pmatrix} -5336P_{0,0} + 16572P_{0,1} + 2910P_{0,2} + 6764P_{0,3} + 5352P_{1,0} + 2910P_{1,3} + \\ 1788P_{2,0} + 16572P_{2,3} - 529P_{3,0} + 1788P_{3,1} + 5352P_{3,2} - 5336P_{3,3} \end{pmatrix}
$$

$$
P_{2,1} = \frac{1}{48807}\begin{pmatrix} -5336P_{0,0} + 5352P_{0,1} + 1788P_{0,2} - 529P_{0,3} + 16572P_{1,0} + 1788P_{1,3} + \\ 2910P_{2,0} + 5352P_{2,3} + 6764P_{3,0} + 2910P_{3,1} + 16572P_{3,2} - 5336P_{3,3} \end{pmatrix}
$$

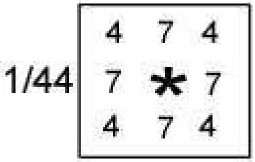
$$
P_{2,2} = \frac{1}{48807}\begin{pmatrix} -529P_{0,0} + 1788P_{0,1} + 5352P_{0,2} - 5336P_{0,3} + 1788P_{1,0} + 16572P_{1,3} + \\ 5352P_{2,0} + 2910P_{2,3} - 5336P_{3,0} + 16572P_{3,1} + 2910P_{3,2} + 6764P_{3,3} \end{pmatrix}
$$

4.4 Bending Energy Mask

From the condition in Proposition 4.1 and Proposition 4.2 we can try to solve a linear system to get minimal bending energy rectangular Bézier surface according to the given exterior control points. The equations are:

$$
\begin{aligned}
44P_{i,j} = {} & 4\ \Big(P_{i-1,j-1} + P_{i+1,j-1} + P_{i-1,j+1} + P_{i+1,j+1}\Big) \\
& + 7\Big(P_{i,j-1} + P_{i-1,j} + P_{i+1,j} + P_{i,j+1}\Big)
\end{aligned} \tag{4}
$$

These equations can be expressed using the symmetric *bending energy mask* shown in Figure 6.



$$1/44 \begin{array}{|ccc|} \hline 4 & 7 & 4 \\ 7 & * & 7 \\ 4 & 7 & 4 \\ \hline \end{array}$$

Fig. 6. Bending energy mask

4.5 Solvability for Linear System

The above linear system (4) tell us that every interior control point can be represented as a convex linear combination of its rectangular neighbour control points, that is: $P_i = \sum_{j \in N_i} \lambda_{ij} P_j$, where N_i denotes 8 neighbour control points for interior control point P_i.

Similar to the triangular case, all control points and corresponding neighbour control points can define a direct graph G. As shown in Fig. 7, for every interior control point P_i, we can also easily find a directed path to one boundary control point P_j, so according to the sufficient condition in reference [6], we can conclude that the linear system (4) has a unique solution.

4.6 Examples

In this section, we compare some examples for the bi-quadratic rectangular Bézier surface case.

We first fix the four boundary curves with its control points, then construct the surface by computing the inner control point through the harmonic mask, the Laplacian mask, the Dirichlet mask, and bending energy mask. Fig. 8 shows the different borders and the rectangular Bézier surfaces constructed by means of the *bending energy mask*. Fig. 9 shows the corresponding rectangular Bézier surfaces constructed by means of the *Dirichlet mask*. Fig. 10 and Fig. 11 shows the corresponding rectangular Bézier surfaces constructed by means of the *Laplacian mask* and *harmonic mask* respectively.

In Table 2,the internal energy of the rectangular Bézier surfaces are shown. From Table 2, we can also assert that the internal energy of the rectangular

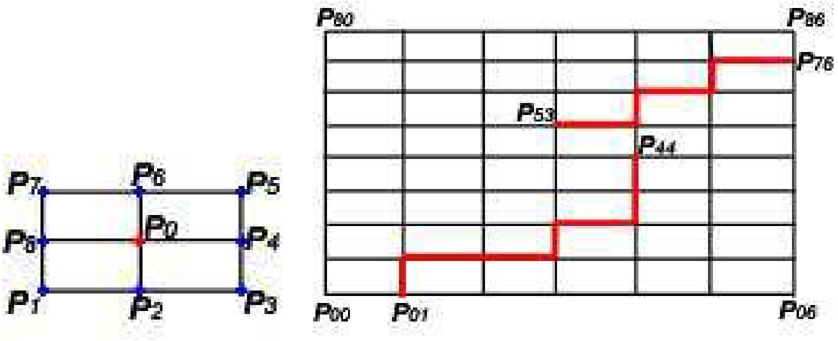


Fig. 7. The rectangular neighbour control points and directed path (solid line) from interior control point to one boundary control point

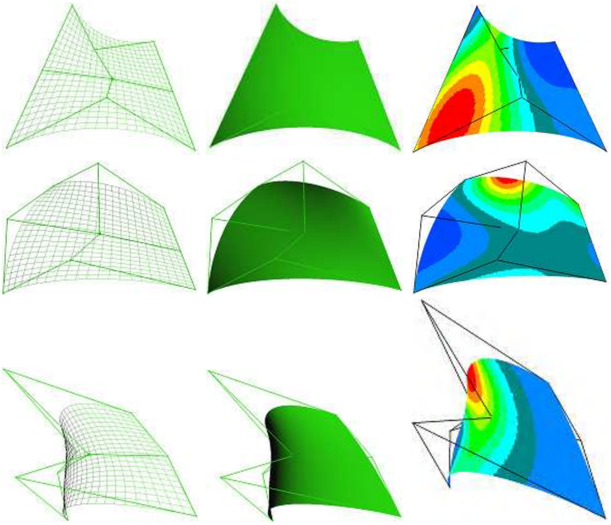


Fig. 8. The rectangular Bézier surfaces associated to different boundary curves with bending energy mask. Left column, the wireframe version of different surfaces. Center column, the render version of different surfaces. Right column, the curvature version of different surfaces. Different grayscales on the curvature version are related to different Gaussian curvatures

Bézier surfaces depends heavily on the boundary curve conditions. In three examples associated to different boundary curves with harmonic, Laplacian, Dirichlet and bending energy mask, the bending energy mask is always the best one, i.e. the minimal internal energy.

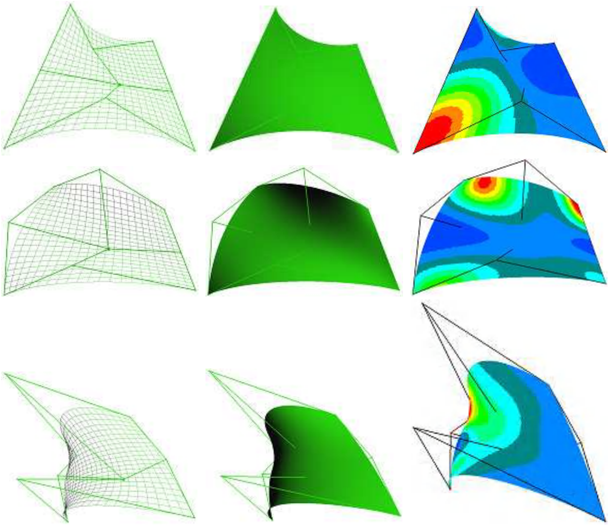


Fig. 9. The rectangular Bézier surfaces associated to different boundary curves with Dirichlet mask

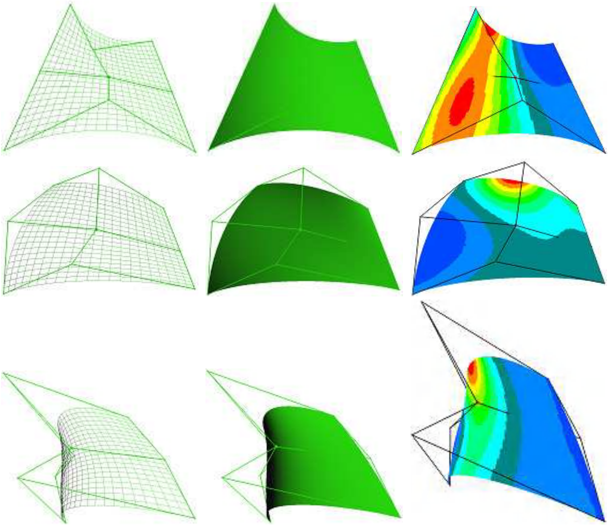


Fig. 10. The rectangular Bézier surfaces associated to different boundary curves with Laplacian mask

5 Conclusion

We discuss the variational problem in this paper: Given the boundary control points of a triangular or a rectangular control net, find out the inner ones in

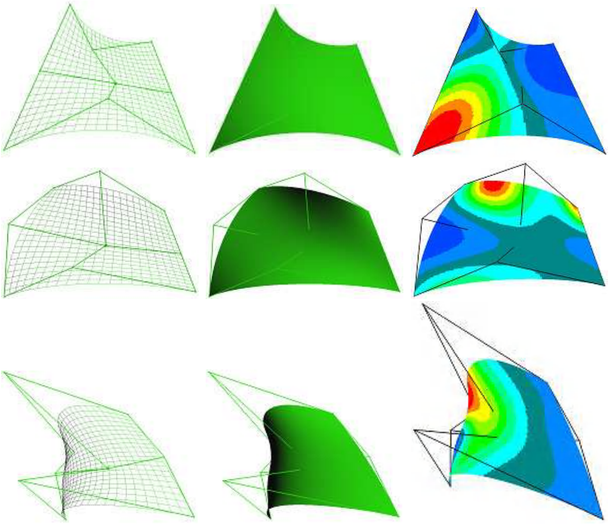


Fig. 11. The rectangular Bézier surfaces associated to different boundary curves with harmonic mask

Table 2. Comparison among different masks

Masks	Example 1		Example 2		Example 3	
	Stretching	Bending	Stretching	Bending	Stretching	Bending
Harmonic Mask	0.8200	1.6320	0.9760	2.4169	1.1927	6.2698
Laplacian Mask	0.8280	1.6080	1.0124	2.3076	1.2696	6.0391
Dirichlet Mask	0.8190	1.7100	0.9714	2.7722	1.1831	7.0194
Bending Energy Mask	0.8242	1.5964	0.9950	2.2545	1.2327	5.9273

such a way that the resulting triangular or rectangular Bézier surface have the minimal energy among all the Bézier surfaces with the same prescribed boundaries.

We propose two new masks related to the two variational problems. One is related to the triangular Bézier surface of minimal bending energy; and the other is associated to the rectangular Bézier surface of minimal bending energy. Experimental results show that the results obtained by using our masks are better than those obtained by using harmonic, Laplacian and Dirichlet masks.

Acknowledgement. This work has been partially supported by Chinese National 973 Fundamental Science Programs (973 program) (2002CB312101) and National Natural Science Foundation (NSFC) (60373036,60333010).

References

1. Arnal, A., Lluch, A., Monterde, J.: Triangular Bézier surfaces of minimal area. Proceedings of the International Workshop on Computer Graphics and Geometric Modeling, CG&GM'2003, Montreal, LNCS 2669, pages 366-375, Springer-Verlag (2003)
2. Brunnett, G., Hagen, H., Santarelli, P.: Variational design of curves and surfaces. Surv. Math. Industry 3(27) (1993) 1-27
3. Cosín, C., and Monterde, J.: Bézier surfaces of minimal area. Proceedings of the International Conference of Computational Science, ICCS'2002, Amsterdam, Eds. Sloot, Kenneth Tan, Dongarra, Hoekstra, LNCS 2330, vol 2, pages 72-81, Springer-Verlag (2002)
4. do Carmo, M. P.: Differential Geometry of Curves and Surfaces. Prentice-Hall International, Englewood Cliffs, NJ (1976)
5. Farin, G., and Hansford, D.: Discrete Coons patches. Computer Aided Geometric Design 16(7) (1999) 691-700
6. Floater, M., Reimers, M.: Meshless parameterization and surface reconstruction. Computer Aided Geometric Design 18(2) (2001) 77-92
7. Gallier, J.: Curves and Surfaces in Geometric Modeling. Morgan Kaufmann publishers, S. Francisco, California (2000)
8. Greiner, G., Loos, J., Wesselink, W.: Data dependent thin plate energy and its use in interactive surface modeling. Computer Graphics Forum 15(3) (1996) 175-186
9. Hagen, H.: Variational principles in curve and surface design. Proceedings of the 5th IMA Conference on the Mathematics of Surfaces, Edinburgh, Eds. Fisher, R. B., Clarendon Press Oxford (1994) 169-190
10. Monterde, J.: The Plateau-Bézier problem. Proceedings of the 10th IMA Conference on the Mathematics of Surfaces, Leeds, LNCS 2768, pages 262-273, Springer-Verlag (2003)
11. Monterde, J.: Bézier surfaces of minimal area: The Dirichlet approach. Computer Aided Geometric Design 21(2) (2004) 117-136
12. Petitjean, S.: A survey of methods for recovering quadrics in triangular meshes. ACM Comput. Surv. 34(2) (2002) 211-262
13. Schneider, R., and Kobbelt, L.: Geometric fairing of irregular meshes for free-form surface design. Computer Aided Geometric Design 18(4) (2001) 359-379
14. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. ACM SIGGRAPH'87 (1987) 205-214
15. Veltkamp, R., and Wesselink, W.: Modeling 3d Curves of minimal energy. Computer Graphics Forum 14(3) (1995) 97-110
16. Veltkamp, R., and Wesselink, W.: Variational modeling of triangular Bézier surfaces. Technical reports of Utrecht University , report no: UU-CS-1996-40 (1996)
17. Welch, W., and Witkin, A.: Variational surface modeling. ACM SIGGRAPH'92 (1992) 157-166
18. Welch, W., and Witkin, A.: Free-Form shape design using triangulated surfaces. ACM SIGGRAPH'94 (1994) 247-256

A Appendix: Proof of Proposition 4.1

For any $a \in \{1, 2, 3\}$:

$$\frac{\partial E(p)}{\partial x_{ij}^a} = 2 \int_\Omega \Big(\langle \frac{\partial P_{uu}}{\partial x_{ij}^a}, P_{uu} \rangle + 2 \langle \frac{\partial P_{uv}}{\partial x_{ij}^a}, P_{uv} \rangle + \langle \frac{\partial P_{vv}}{\partial x_{ij}^a}, P_{vv} \rangle \Big) dudv$$

Similar to the triangular case, we use the difference operators and compute the following partial derivatives:

$$\begin{aligned}
P_{uu}(u,v) &= n(n-1) \sum_{k=0}^{n-2} \sum_{l=0}^{m} B_k^{n-2}(u) B_l^m(v) \Delta^{2,0} P_{k,l} \\
P_{uv}(u,v) &= nm \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} B_k^{n-1}(u) B_l^{m-1}(v) \Delta^{1,1} P_{k,l} \\
P_{vv}(u,v) &= m(m-1) \sum_{k=0}^{n} \sum_{l=0}^{m-2} B_k^{n}(u) B_l^{m-2}(v) \Delta^{0,2} P_{k,l} \\
\frac{\partial P_{uu}}{\partial x_{ij}^a} &= n(n-1) \Big(B_{i-2}^{n-2}(u) - 2B_{i-1}^{n-2}(u) + B_i^{n-2}(u) \Big) B_j^m(v) e^a \\
\frac{\partial P_{uv}}{\partial x_{ij}^a} &= nm \Big(B_{i-1}^{n-1}(u) - B_i^{n-1}(u) \Big) \Big(B_{j-1}^{m-1}(v) - B_j^{m-1}(v) \Big) e^a \\
\frac{\partial P_{vv}}{\partial x_{ij}^a} &= m(m-1) B_i^n(u) \Big(B_{j-2}^{m-2}(v) - 2B_{j-1}^{m-2}(v) + B_j^{m-2}(v) \Big) e^a
\end{aligned}$$

where $e^a (a \in \{1, 2, 3\})$ denotes the ath vector of the canonical basis.

So, the differential of the bending energy functional:

$$\begin{aligned}
\frac{\partial E(p)}{\partial x_{ij}^a} = 2 \Bigg[& n(n-1) \int_\Omega \Big(B_{i-2}^{n-2}(u) - 2B_{i-1}^{n-2}(u) + B_i^{n-2}(u) \Big) B_j^m(v) \langle e^a, P_{uu} \rangle dudv \\
& + 2nm \int_\Omega \Big(B_{i-1}^{n-1}(u) - B_i^{n-1}(u) \Big) \Big(B_{j-1}^{m-1}(v) - B_j^{m-1}(v) \Big) \langle e^a, P_{uv} \rangle dudv \\
& + m(m-1) \int_\Omega B_i^n(u) \Big(B_{j-2}^{m-2}(v) - 2B_{j-1}^{m-2}(v) + B_j^{m-2}(v) \Big) \langle e^a, P_{vv} \rangle dudv \Bigg]
\end{aligned}$$

By applying the following identity of Bernstein Polynomials and integral equation:

$$B_i^n(u) B_j^n(u) = \frac{\binom{n}{i}\binom{m}{j}}{\binom{n+m}{i+j}} B_{i+j}^{n+m}(u), \quad \int_0^1 B_i^n(u) du = \frac{1}{n+1}$$

Then, we can get the condition for the extremals of bending energy functional: $\frac{\partial E(p)}{\partial x_{ij}^a} = 0$. It is:

$$0 = K_1 \sum_{k,l=0}^{n-2,m} \Big[\frac{\binom{n-2}{i-2}}{\binom{2n-4}{i+k-2}\binom{2m}{j+l}} - 2\frac{\binom{n-2}{i-1}}{\binom{2n-4}{i+k-1}\binom{2m}{j+l}} + \frac{\binom{n-2}{i}}{\binom{2n-4}{i+k}\binom{2m}{j+l}}\Big] H_{k,l}^{2,0}$$

$$+K_2 \sum_{k,l=0}^{n-1,m-1} \Big[\frac{\binom{n-1}{i-1}\binom{m-1}{j-1}}{\binom{2n-2}{i+k-1}\binom{2m-2}{j+l-1}} - \frac{\binom{n-1}{i-1}\binom{m-1}{j}}{\binom{2n-2}{i+k-1}\binom{2m-2}{j+l}}\Big] H_{k,l}^{1,1}$$

$$-K_2 \sum_{k,l=0}^{n-1,m-1} \Big[\frac{\binom{n-1}{i}\binom{m-1}{j-1}}{\binom{2n-2}{i+k}\binom{2m-2}{j+l-1}} - \frac{\binom{n-1}{i}\binom{m-1}{j}}{\binom{2n-2}{i+k}\binom{2m-2}{j+l}}\Big] H_{k,l}^{1,1}$$

$$+K_3 \sum_{k,l=0}^{n,m-2} \Big[\frac{\binom{m-2}{j-2}}{\binom{2n}{i+k}\binom{2m-4}{j+l-2}} - 2\frac{\binom{m-2}{j-1}}{\binom{2n}{i+k}\binom{2m-4}{j+l-1}} + \frac{\binom{m-2}{j}}{\binom{2n}{i+k}\binom{2m-4}{j+l}}\Big] H_{k,l}^{0,2}$$

where the coefficients: $K_1 = \frac{n^2(n-1)^2}{(2n-3)(2m+1)}\binom{m}{j}$, $K_2 = \frac{2n^2m^2}{(2n-1)(2m-1)}$, $K_3 = \frac{m^2(m-1)^2}{(2n+1)(2m-3)}\binom{n}{i}$, and the notation $H_{k,l}^{p,q} = \binom{n-p}{k}\binom{m-q}{l}\Delta^{p,q}P_{k,l}$.
If denote:

$$A_{n,i}^k = \frac{nk(n-1)(k-1) + i^2(n-2)(n-3) - i(n-3)(2kn+n-2k-2)}{(n-i)(n-i-1)(2n-i-k-2)(2n-i-k-3)}$$

$$B_{n,i}^k = \frac{ni-nk-i}{(n-i)(2n-k-i-1)}$$

We can prove the equations (3).

Positivity-Preserving Scattered Data Interpolation

Abd. Rahni Mt. Piah[1], Tim N.T. Goodman[2],
and Keith Unsworth[3]

[1] Universiti Sains Malaysia,
School of Mathematical Sciences, Penang, Malaysia
arahni@cs.usm.my
[2] University of Dundee,
Department of Mathematics, Dundee, Scotland
tgoodman@maths.dundee.ac.uk
[3] Lincoln University,
Applied Computing Group, Canterbury, New Zealand
unsworth@lincoln.ac.nz

Abstract. The construction of a C^1 interpolant to scattered data is considered in which the interpolant is positive everywhere if the original data are positive. This study is motivated by earlier work in which sufficient conditions are derived on Bézier points in order to ensure that surfaces comprising cubic Bézier triangular patches are always positive. In the current work, simpler and more relaxed conditions are derived on the Bézier points. The gradients at the data sites are then calculated to ensure that these conditions are satisfied. Each triangular patch of the interpolating surface is formed as a convex combination of three cubic Bézier triangular patches. Its construction is local. A number of examples are presented.

1 Introduction

In recent years, a considerable number of research articles have been published that focus on shape preserving interpolation, both for curves and surfaces. The properties that are most often used to quantify "shape" are convexity, monotonicity (for non-parametric data) and positivity. It is the last of these that is of interest in this paper, namely if all the sampled data are positive, then the interpolating curve or surface should be positive everywhere.

The need to preserve positivity can be readily seen in the area of scientific visualization. Scientific visualization provides a means of understanding various physical phenomena, from limited or incomplete information. The data that are known represent only a sample and may not be sufficient to let one visualize the entire entity. As such one uses interpolation to construct an empirical model which matches the data samples and approximates the unknown entity at intermediate locations. Preserving positivity is particularly important when visualizing a physical entity that cannot possibly be negative. An example is

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 336–349, 2005.

included later in this paper that interpolates rainfall data. Examples of studies in the area of positivity-preservation include [3] which considers curves and [2] which considers surfaces. More references on this topic are included in these studies.

The problem that we are concerned with is the interpolation of scattered data. This problem occurs in many practical situations where data are gathered experimentally or via simulation studies. A review of this area of research may be found in [11]. One approach to this problem uses the idea of "meshless" surfaces such as radial basis functions and Shepard-type methods. A second approach triangulates the data points, leading to a piecewise construction of the surface. Constraining such surfaces to be positive everywhere has received less attention compared with preserving positivity for curves and for surfaces that interpolate gridded data. However, the reader may refer to [14], [15] with regard to meshless surfaces, and [1], [4], [12], [13] for triangulated surfaces. It is the latter approach that is of interest in this paper in which we follow a similar approach to the work described in [4].

Thus we may describe the problem as follows: given functional data

$$(x_i, y_i, z_i),\ i = 1, \dots N,\ \text{where } z_i \geq 0\ \forall\ i,$$

we wish to construct a C^1 surface $z = F(x, y)$ such that

$$z_i = F(x_i, y_i),\ i = 1, \dots N,\ \text{and } F(x, y) \geq 0\ \forall\ x, y.$$

In both this paper and that of [4], the interpolating surface comprises cubic Bézier triangular patches with sufficient conditions imposed on the ordinates of the Bézier control points in each triangle to guarantee preservation of positivity. The derivatives at the data points are specified to be consistent with these conditions. The main difference between this work and that of [4] is the way in which the Bézier ordinates are constrained. Compared with the work of [4], we offer more relaxed sufficient conditions that are easier to compute. These are derived in detail in Section 2 of this paper.

Details of the algorithm for generating the surface are given in Section 3. A brief summary is as follows:

1. triangulate the domain
2. specify derivatives at the data points
3. assign Bézier ordinate values for each triangular patch
4. generate the triangular patches of the surface.

Section 4 discusses results obtained using the scheme. The paper finishes with a summary and a proposal for future work.

The reader should note that in general we use the term 'positivity' to refer to 'greater than or equal to zero', rather than the more rigorous but somewhat awkward 'non-negativity'. However, note that the conditions that are derived in Section 2 consider the case in which the given data are *strictly* positive. The way in which the scheme deals with general positive data is mentioned at the end of Section 2.

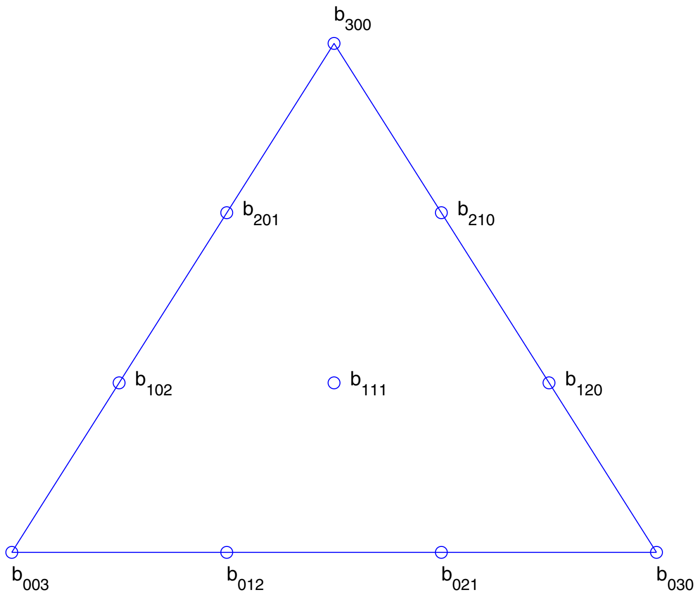


Fig. 1. Relative locations of Bézier ordinates for $P(u, v, w)$

2 Sufficient Positivity Conditions for a Cubic Bézier Triangular Patch

Consider a triangle T, with vertices A, B, C, and barycentric coordinates u, v, w such that any point V on the triangle can be expressed as

$$V = uA + vB + wC$$

where

$$u + v + w = 1 \quad \text{and} \quad u, v, w \geq 0.$$

A cubic Bézier triangular patch P on T is defined as

$$\begin{aligned} P(u, v, w) = {} & u^3 b_{300} + v^3 b_{030} + w^3 b_{003} + 3u^2 v b_{210} + 3u^2 w b_{201} + 3v^2 u b_{120} \\ & + 3v^2 w b_{021} + 3w^2 u b_{102} + 3w^2 v b_{012} + 6uvw b_{111} \end{aligned} \tag{1}$$

where b_{rst} are the Bézier ordinates of P (see Figure 1).

We assume that the Bézier ordinates at the vertices are strictly positive, i.e. $b_{300},\ b_{030},\ b_{003} > 0$. We shall derive sufficient conditions on the remaining Bézier ordinates for the entire Bézier patch to be positive. In [4], such sufficient conditions are based on the univariate results on positivity in [8]. The following is a proposition from [4].

Proposition 1. *Consider the cubic Bezier triangular patch* $P(u, v, w)$ *with* $b_{300} = \alpha l, b_{030} = \beta l, b_{003} = l, l > 0$ *and* $\alpha \geq \beta \geq 1$. *If* $b_{210}, b_{201}, b_{120}, b_{021}, b_{102}$, $b_{012}, b_{111} \geq -l/3a$, *where* a *is the unique solution in* $(1, 8/3]$ *of the equation*

$$16 - 8\alpha + (72\alpha - 27\alpha^2)a + 54\alpha^2 a^2 - 27\alpha^2 a^3 = 0$$

then $P(u, v, w) \geq 0 \ \forall \ u, v, w \geq 0, \ u + v + w = 1$.

The scheme constructs an interpolating surface comprising cubic Bézier triangular patches such that the Bézier ordinates satisfy Proposition 1 in each triangle.

Now suppose $l = 1, \alpha = \beta > 1$, and suppose $\alpha \to \infty$. In order to guarantee positivity, using Proposition 1, the common lower bound of the seven Bézier ordinates approaches $-1/3$. It will never be less than this value. We will see that by choosing this lower bound in a different and simpler way, the magnitude of this lower bound can become infinitely large in the above case. We now consider this alternative scheme.

Let $A = b_{300}, \ B = b_{030}, \ C = b_{003}$, and $A, B, C > 0$. Our approach is to find the minimum value $F(A, B, C)$ such that if all the Bézier ordinates, apart from $b_{300}, \ b_{030}, \ b_{003}$, have the value $F(A, B, C)$, then $P(u, v, w) = 0$.

Apart from the Bézier points at the vertices, we assume that all the Bézier ordinates have the same value $-t < 0$ (where $t > 0$). Thus, (1) becomes

$$\begin{aligned} P(u, v, w) &= Au^3 + Bv^3 + Cw^3 \\ &\qquad -3t(u^2 v + u^2 w + v^2 u + v^2 w + w^2 u + w^2 v + 2uvw) \\ &= Au^3 + Bv^3 + Cw^3 - t(1 - u^3 - v^3 - w^3) \\ &= (A + t)u^3 + (B + t)v^3 + (C + t)w^3 - t \end{aligned} \tag{2}$$

Clearly when $t = 0, \ P(u, v, w) > 0$. As t increases, $P(u, v, w)$ decreases. We are interested in finding the value $t = t_0$ when the minimum value of $P(u, v, w) = 0$.

Assume t is fixed. At the minimum value of $P(u, v, w)$ we know that:

$$\frac{\partial P}{\partial u} - \frac{\partial P}{\partial v} = 0 \quad \text{and} \quad \frac{\partial P}{\partial u} - \frac{\partial P}{\partial w} = 0$$

Thus

$$\frac{\partial P}{\partial u} = \frac{\partial P}{\partial v} = \frac{\partial P}{\partial w} \tag{3}$$

From (2) we have

$$\frac{\partial P}{\partial u} = 3(A + t)u^2, \quad \frac{\partial P}{\partial v} = 3(B + t)v^2, \quad \frac{\partial P}{\partial w} = 3(C + t)w^2.$$

Thus, substituting into (3) we obtain:

$$\frac{u^2}{v^2} = \frac{B + t}{A + t} \quad \text{and} \quad \frac{u^2}{w^2} = \frac{B + t}{C + t}$$

Hence:

$$u^2 : v^2 : w^2 = \frac{1}{A + t} : \frac{1}{B + t} : \frac{1}{C + t}.$$

Remembering that $u + v + w = 1$ we obtain:

$$u = \frac{\frac{1}{\sqrt{A+t}}}{\frac{1}{\sqrt{A+t}} + \frac{1}{\sqrt{B+t}} + \frac{1}{\sqrt{C+t}}}$$
$$v = \frac{\frac{1}{\sqrt{B+t}}}{\frac{1}{\sqrt{A+t}} + \frac{1}{\sqrt{B+t}} + \frac{1}{\sqrt{C+t}}}$$
$$w = \frac{\frac{1}{\sqrt{C+t}}}{\frac{1}{\sqrt{A+t}} + \frac{1}{\sqrt{B+t}} + \frac{1}{\sqrt{C+t}}}.$$

From the above, and (2), the minimum value of $P(u,v,w)$ is

$$P(u,v,w) = \frac{1}{(\frac{1}{\sqrt{A+t}} + \frac{1}{\sqrt{B+t}} + \frac{1}{\sqrt{C+t}})^2} - t$$

We now need to choose a value of $t = t_0$ so that this minimum value is zero. Noting that $t > 0$, $P(u,v,w) = 0$ when

$$\frac{1}{\sqrt{\frac{A}{t}+1}} + \frac{1}{\sqrt{\frac{B}{t}+1}} + \frac{1}{\sqrt{\frac{C}{t}+1}} = 1$$

If we define $s_0 = \frac{1}{t_0}$, it follows immediately that s_0 is the solution of

$$G(s) = 1, \quad s \geq 0 \tag{4}$$

where

$$G(s) = \frac{1}{\sqrt{As+1}} + \frac{1}{\sqrt{Bs+1}} + \frac{1}{\sqrt{Cs+1}} \tag{5}$$

Recalling that $A, B, C > 0$, it is easy to show that for $s \geq 0$, $G'(s) > 0$, $G''(s) < 0$. In addition, let $X = \max(A,B,C)$ and $Y = \min(A,B,C)$. Clearly

$$\frac{3}{\sqrt{Xs+1}} \leq G(s) \leq \frac{3}{\sqrt{Ys+1}}$$

In particular

$$G(\frac{8}{X}) \geq 1 \text{ and } G(\frac{8}{Y}) \leq 1.$$

Figure 2 shows the form of $G(s)$, $s \geq 0$. Also shown are the relative locations of $8/X$, $8/Y$ and s_0.

We now have the following proposition, as an alternative to Proposition 1.

Proposition 2. *Consider the cubic Bézier triangular patch $P(u,v,w)$ with $b_{300} = A, b_{030} = B, b_{003} = C$, $A, B, C > 0$. If $b_{210}, b_{201}, b_{120}, b_{021}, b_{102}, b_{012}$, $b_{111} \geq -t_0$, where $t_0 = \frac{1}{s_0}$ is the unique solution of* (4), (5) *then $P(u,v,w) \geq 0 \ \forall \ u,v,w \geq 0, \ u+v+w = 1$.*

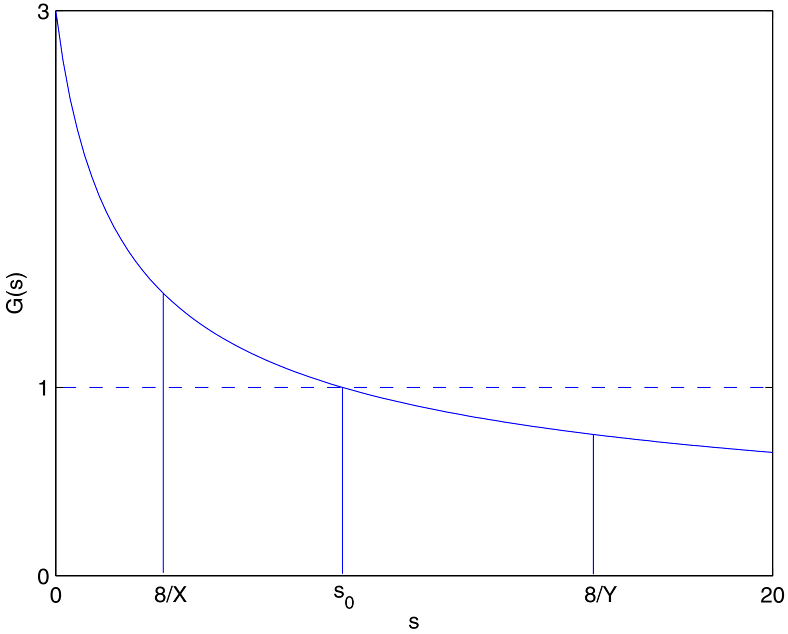


Fig. 2. Function $G(s)$ for $s \geq 0$

These more relaxed sufficient conditions, compared to those in Proposition 1, prescribe lower bounds for the Bézier ordinates. To obtain the value of s_0 for given values of A, B, C we can use a simple iterative scheme. Since we need to calculate the root of an equation that will act as a lower bound on the Bézier ordinates, in choosing an iterative scheme we must ensure one-sided convergence, ie. that s_0 is approached from above. The convexity of $G(s)$ (see Figure 2) means that this can be achieved using the method of false-position [5]. An initial estimate for the root will be the value of s for which the line joining $8/X$ and $8/Y$ has the value 1.

We suggest that this is simpler than calculating the roots of the cubic expression in Proposition 1. In addition, the scheme in [4] was considered earlier in cases in which $l = 1, \alpha = \beta > 1$ and $\alpha \to \infty$. It was noted that the lower bound for the Bézier ordinates approaches $-1/3$. Consider the same case with this alternative scheme. Let $A = B > C = 1$. We have,

$$G(s) = \frac{2}{\sqrt{As+1}} + \frac{1}{\sqrt{s+1}}$$

Therefore, as $A \to \infty$,

$$G(s) \to \frac{1}{\sqrt{s+1}}.$$

Hence, $s_0 \to 0$ and therefore $t_0 \to \infty$. Thus the ordinate values are unbounded.

Note that, in practice, if any of the values of A, B or C are zero (i.e. the given data are positive but not strictly positive), t_0 is assigned the value zero for that triangle.

3 Construction of Positivity Preserving Interpolating Surface

Having established sufficient conditions on the Bézier points as described above, we are now able to construct the interpolating surface. Our approach is very similar to that of [4] and a brief overview is given in the introduction. For the sake of completeness we give more details in this section, but the reader can refer to [4] for a fuller explanation.

As already noted, the surface comprises cubic Bézier triangular patches each of which is guaranteed to remain positive. Essentially, the construction process consists of the following steps:

1. Triangulate the domain data $(x_i, y_i), i = 1, \dots N$.
2. Initialise partial derivatives at each data point, then modify them (if necessary) to be consistent with the positivity constraints imposed on the Bézier ordinates.
3. For each domain triangle:
   - Assign boundary Bézier ordinate values using the partial derivative values from step 2.
   - Calculate three triangular patches using the boundary ordinates from step 3, and an inner Bézier ordinate value that satisfies the positivity constraint and ensures C^1 continuity across one of the triangle boundaries.
   - Calculate final triangular patch as a blend of these three patches

We now give further details of each of these steps.

3.1 Triangulation

Given the data values to be interpolated, let D be the convex hull of the points $U_i = (x_i, y_i), i = 1, \dots N$. Delaunay triangulation [6] is used to triangulate D, with each $U_i,\ 1 \le i \le N$ the vertex of a triangle.

3.2 Derivative Specification

For the interpolating surface $F(x, y)$, values for the partial derivatives F_x, F_y are required at $(x_i, y_i), i = 1, \dots N$. Estimation of these derivatives is done by using the method proposed in [9]. Using these values, the derivative *along a triangle edge* can be derived. For example, suppose e_{jk} is the edge connecting (x_j, y_j) to (x_k, y_k) for triangular patch P, then:

$$\frac{\partial P}{\partial e_{jk}} = (x_k - x_j)\frac{\partial F}{\partial x} + (y_k - y_j)\frac{\partial F}{\partial y}$$

The derivatives are used to assign initial values to the Bézier ordinates located on each triangle boundary. Thus, for example, for an arbitrary triangular patch P we have:

$$b_{300} = F(V_1), \quad b_{210} = F(V_1) + \frac{1}{3}\frac{\partial P}{\partial e_3}(V_1)$$

where e_3 is the edge from vertex V_1 to vertex V_2. However, for a triangle P, the initial estimate for each edge ordinate may not satisfy the positivity condition for P. If it does not, the magnitudes of F_x, F_y at the vertices need to be reduced so that the condition is satisfied. Reduction of the derivatives at a vertex V is achieved by multiplying each derivative at that vertex by a scaling factor α, where $0 < \alpha < 1$. The value of α is obtained by considering all triangles that meet at vertex V, and obtaining the smallest value of α that will guarantee satisfaction of the positivity condition for all these triangles: i.e. so that, for example,

$$(b_{210})_j = F(V_1) + \frac{\alpha}{3}\frac{\partial P}{\partial (e_3)_j}(V_1) \geq -(t_0)_j$$

where subscript j represents quantities corresponding to triangle j. Having adjusted these derivatives, if necessary, the Bézier ordinates are recalculated using the formulae above.

For each triangle, the inner Bézier point remains to be calculated. This needs to be done in order to guarantee preservation of positivity and to ensure C^1 continuity across patch boundaries. We adopt the approach presented in [7] which guarantees C^1 continuity and has cubic precision. In this scheme each triangular surface patch is a convex combination of three cubic Bézier patches. The work of [4] also adopts this approach, with a small difference in the form of the convex combination used. The boundary ordinates for all three patches are the same and calculated as above.

Suppose therefore that each triangular surface patch P is a convex combination of the triangular patches $P^i,\ i = 1, 2, 3$. Each P^i may have a different inner Bézier ordinate $b^i_{111},\ i = 1, 2, 3$. Initial values for $b^i_{111},\ i = 1, 2, 3$ are calculated from a system of equations that relates the Bézier ordinates for two triangles that share a common edge. See equations (9)-(12) of [7], or (3.4)-(3.7) of [4]. Modifications to this approach are required for triangles that have an edge on the boundary of the domain D: these are documented in both [7] and [4].

If an inner Bézier point $b^i_{111},\ i = 1, 2, 3$ fails to satisfy the positivity condition it is modified according to Proposition 2. The inner Bézier point of the adjacent triangle is also adjusted in order to maintain C^1 continuity and cubic precision.

3.3 Triangular Patch Generation

For each patch P, once all the Bézier ordinates have been assigned for the triangular patches $P^i,\ i = 1, 2, 3$, the final surface can be generated. In particular, for barycentric coordinates u, v, w:

$$P = c_1 P^1 + c_2 P^2 + c_3 P^3$$

where

$$c_1 = \frac{vw}{vw + wu + uv}, \quad c_2 = \frac{wu}{vw + wu + uv}, \quad c_3 = \frac{uv}{vw + wu + uv}$$

This is the same formulation as used in [7], although slightly different to that used in [4].

4 Examples

We present output from three examples in this section. The first example considers a situation in which there is a significant difference in the bounds that must be satisfied in order to preserve positivity, compared with the method of [4]. The data to be interpolated comprises four data points, namely:

x	y	z
0.1	0.1	1×10^8
0.1	0.9	1×10^8
0.9	0.1	0.1
0.3	0.35	1.0

As noted in Section 2, for the triangle with the first two data points as vertices, the current scheme allows its Bézier ordinates to be relatively large and negative while still guaranteeing positivity of the surface. The linear interpolant to the data is shown in Figure 3 and the corresponding positivity preserving surface in Figure 4.

For this particular data set the Bézier ordinates referred to above are bounded below by -183.65. In the case of [4], the ordinate values would be bounded below by $-1/3$.

The second example interpolates one day's rainfall data from the Arthur's Pass region of the South Island of New Zealand. The data to be interpolated comprises five data points, namely:

Longitude	Latitude	Rainfall(mm)
171.567	-42.950	4.6
171.672	-43.470	0.1
171.946	-43.529	0.2
171.800	-43.800	0.4
171.208	-44.035	0.2

The linear interpolant to the data is shown in Figure 5 and the corresponding positivity preserving surface in Figure 6.

The final example comprises 36 data points sampled from the well-known data set taken from [10].

$$S(x,y) = \begin{cases} 1.0 & \text{if } (y-x) \geq 0.5 \\ 2(y-x) & \text{if } 0.5 \geq (y-x) \geq 0.0 \\ \frac{cos(4\pi\sqrt{(x-1.5)^2+(y-0.5)^2})+1}{2} & \text{if } (x-1.5)^2 + (y-0.5)^2 \leq \frac{1}{16} \\ 0 & \text{otherwise} \end{cases}$$

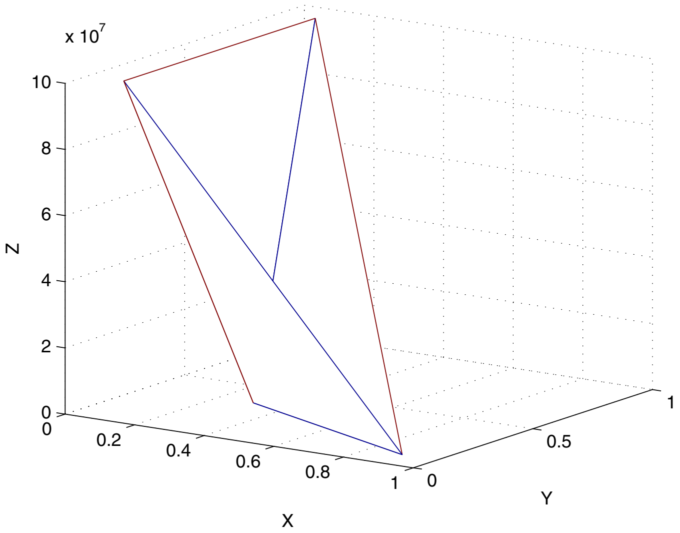


Fig. 3. Linear interpolant for data with large differences in vertex ordinates

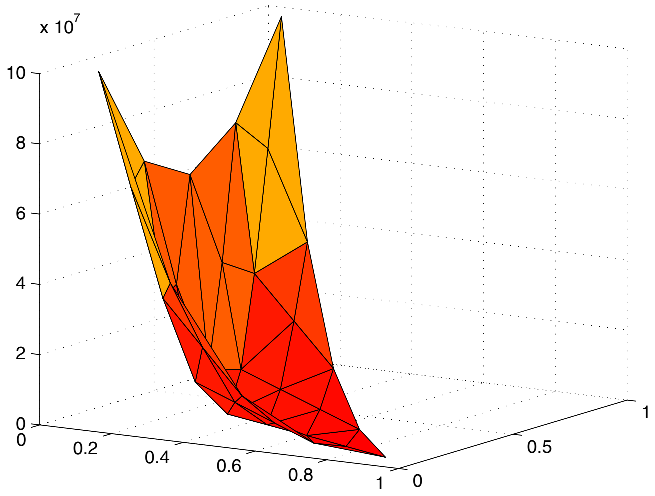


Fig. 4. Positivity-preserving surface for data with large differences in vertex ordinates

The function $S(x, y)$ is shown in Figure 7. Notice that it has areas where it is exactly zero, and a peak and upper shelf where it has a value of 1.0. A linear interpolant to the sampled data is shown in Figure 8. Output from the positivity-preserving scheme of this study is shown in Figure 9. It clearly shows that the surface remains positive everywhere. However the output also shows unwanted oscillations in the region of the upper shelf. There is nothing in the current

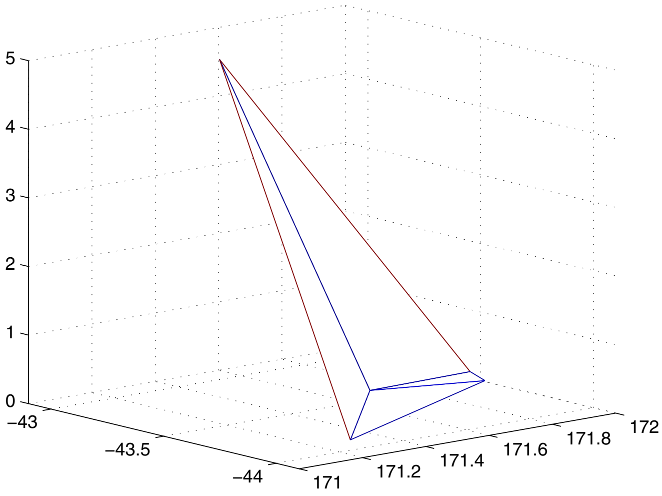


Fig. 5. Linear interpolant to rainfall data

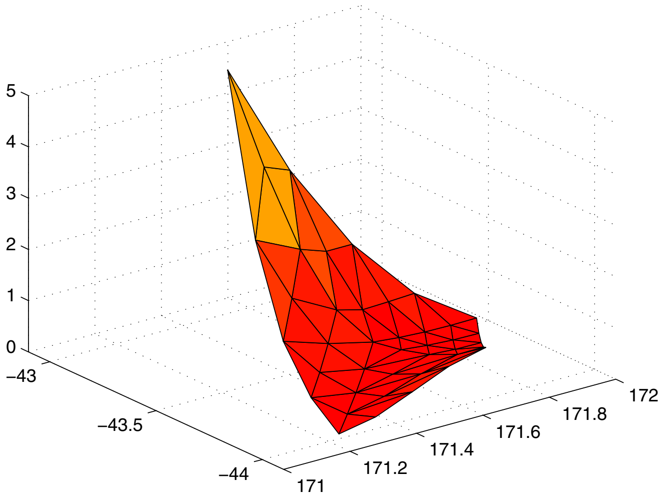


Fig. 6. Positivity-preserving interpolant to rainfall data

scheme to deal with this, however it would be a straightforward extension of the method (as is done in [4]) to prevent the surface exceeding the upper bound of 1.0 in this region.

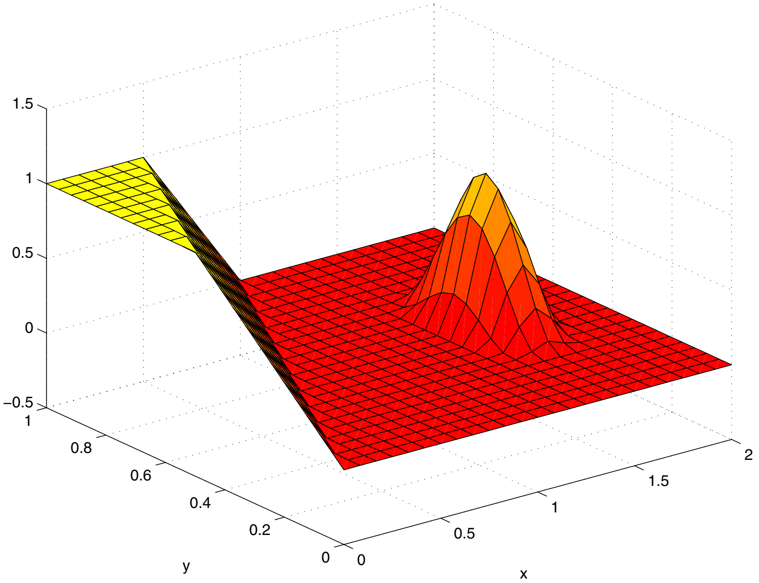


Fig. 7. Lancaster and Salkauskas function

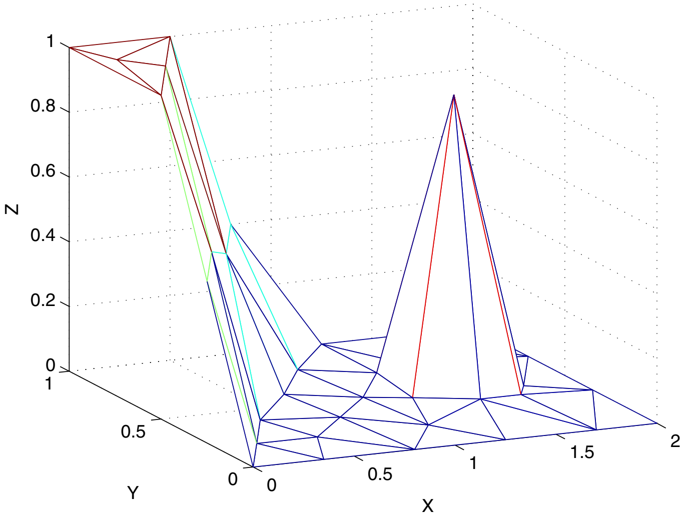


Fig. 8. Linear interpolant for data sampled from Lancaster and Salkauskas function

5 Conclusions

In this study, we have considered the generation of non-parametric surfaces that interpolate positive scattered data. The surfaces comprise piecewise cubic Bézier

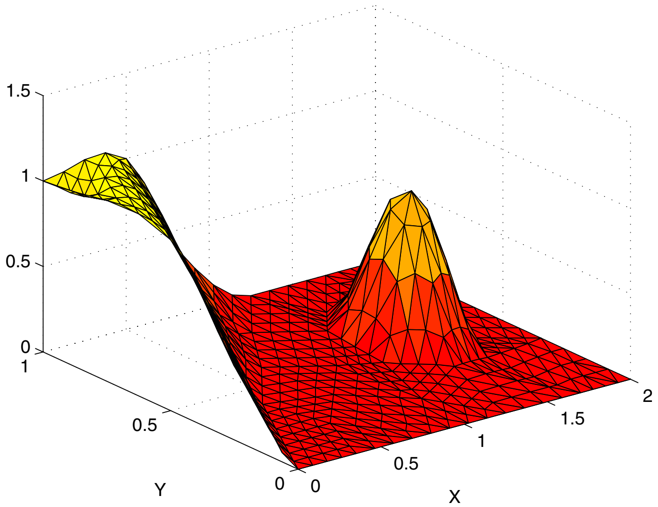


Fig. 9. Positivity-preserving interpolant for data sampled from Lancaster and Salkauskas function

triangles. The approach is similar to that of [4], but imposes more relaxed and simpler conditions on the Bézier ordinates. This paper considers positivity only, however it could easily be extended to range restricted scattered data interpolation in an analogous manner to that of [4].

Both this scheme (and that of [4]) suffer from the constraint that in order to prevent a surface patch becoming negative all Bézier points within the corresponding triangle (except those at the vertices) may be assigned the same value according to Proposition 2. This makes the problem tractable and achieves the desired outcomes. More flexibility would be achieved however if they could be adjusted independently while still ensuring positivity. Preliminary investigations indicate that this approach leads to an optimisation problem, however the overall solution procedure is significantly more complex than that described in this paper. A study of this approach is ongoing.

Acknowledgements. Much of this work was carried out while Professor Goodman and Dr. Abd. Rahni Mt. Piah were visiting Lincoln University, on sabbatical leave from University of Dundee and Universiti Sains Malaysia respectively. They are both extremely grateful to Lincoln University for the warmth of the hospitality, the stimulating environment and financial support that allowed the study to be undertaken.

Dr. Rahni would also like to extend his gratitude to Universiti Sains Malaysia for supporting his attendance at the 11th Mathematics of Surfaces Conference, under its Fundamental Research Grant, Account No. 203/PMATHS/670055.

References

1. Asim, M.R.: Visualization of data subject to positivity constraint, PhD Thesis. School of Computing, University of Leeds (2000)
2. Brodlie, K.W., Butt, S., Mashwama, P.: Visualization of surface data to preserve positivity and other simple constraints. Comput. Graph. **19** (1995) 585–594
3. Butt S., Brodlie, K.W. Preserving positivity using piecewise cubic interpolation. Comput. Graph. **17** (1993) 55–64
4. Chan, E.S., Ong, B.H.: Range restricted scattered data interpolation using convex combination of cubic Bézier triangles. J. Comp. Appl. Math. **136** (2001) 135–147
5. Conte, S.D., de Boor, C.: Elementary Numerical Analysis, McGraw-Hill, Tokyo (1972)
6. Fang, T.P., Piegl, L.A.: Algorithms for Delaunay triangulation and convex-hull computation using a sparse matrix. Computer Aided Design, **24** (1992) 425–436
7. Foley, T.A., Opitz, K.: Hybrid cubic Bézier triangle patches. In Mathematical Methods in Computer Aided Geometric Design II, Lyche, T., Schumaker, L.L. (eds.) Academic Press, New York (1992) 275–286
8. Goodman, T.N.T., Ong, B.H., Unsworth, K.: Constrained interpolation using rational cubic splines. In NURBS for Curve and Surface Design, Farin, G. (ed.), SIAM, Philadelphia (1991) 59–74
9. Goodman, T.N.T., Said, H.B., Chang, L.H.T.: Local derivative estimation for scattered data interpolation. Appl. Math. Comp. **80** (1994) 1–10
10. Lancaster, P., Salkauskas, K.: Curve and Surface Fitting: An Introduction, Academic Press, New York (1986)
11. Lodha, S.K., Franke R.: Scattered Data Techniques for Surfaces. In Scientific Visualization, Dagstuhl 97 Proceedings, IEEE Computer Society Press, (2000) 189–230
12. Mulansky B., Schmidt J.W., Powell-Sabin splines in range restricted interpolation of scattered data. Computing **53** (1994) 137–154
13. Ong, B.H., Wong, H.C.: A C^1 positivity preserving scattered data interpolation scheme. In Advanced Topics in Multivariate Approximation, Fontanella F., Jetter K., Laurent P.J. (eds.) World Scientific, Singapore (1996) 259–274
14. Utreras, F.I.: Positive Thin Plate Splines. J. Approximation Theory and its Applications, **1** (1985) 77–108
15. Xiao Y., Woodbury C.: Constraining Global Interpolation Methods for Sparse Data Volume Visualization. International Jnl of Comp. and Applic., **21** (1999) 59–64

Artifacts in Box-Spline Surfaces

Malcolm A. Sabin, Ursula H. Augsdörfer, and Neil A. Dodgson

Computer Laboratory, University of Cambridge,
15 J.J Thomson Ave., Cambridge CB3 0FD
malcolm@geometry.demon.co.uk, {uha20, nad}@cl.cam.ac.uk

Abstract. Certain problems in subdivision surfaces have provided the incentive to look at artifacts. Some of these effects are common to all box-spline surfaces, including the tensor product B-splines widely used in the form of NURBS, and these are worthy of study. Although we use the subdivision form of box- and B-splines as the mechanism for this study, and also apply the same mechanism to the subdivision schemes which are not box-splines, we are looking at problems which are not specific to subdivision surfaces, but which afflict all Box- and B-splines.

1 Introduction

The paper starts with introductory material: in the remainder of this section terms used are defined, and prior knowledge identified; the broad approach is set out, and notation defined. Then follows a section on the longitudinal artifacts of curves, looking in detail at the effects of even and odd numbers of B-spline factors, and then at the way the same techniques can be applied to subdivision schemes which are not B-splines. Finally a similar section on surfaces, showing that longitudinal and lateral artifacts are parts of the same story.

1.1 Definitions

An *artifact* is defined to be any feature of the surface which cannot be removed by movement of the control points. Whether it is desirable or not is not the question.

This implies that the surface contains spatial frequencies above the Shannon [6] limit relative to the density of the control mesh, because features of spatial frequency below this limit are removable by movement of the control points.

It is convenient to consider two sub-classes, *longitudinal artifacts* where the artifact spatial frequencies have the same direction across the surface as the intended shape, and *lateral artifacts* where the artifact spatial frequencies have a different direction. For curves only the longitudinal artifacts can occur.

Longitudinal artifacts are generally quite mild in effect, merely slightly altering the shape of the profiles involved. In Figure 1, the two loops are exactly the same shape—a cyclic cubic B-spline with four control points at the corners of a square—but one has been rotated through 45 degrees so that the non-circularity is readily evident.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 350–363, 2005.

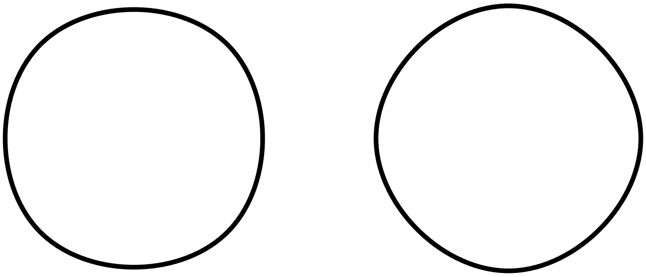


Fig. 1. Two copies of a cyclic cubic B-spline with just four control points. The difference between the two shapes is due to the different phase of the longitudinal artifact because one of the shapes is rotated through 45 degrees

Lateral artifacts are best known as the ‘dinosaur back’ produced when a more-or-less extruded feature is run in a direction not aligned with the underlying grid. (Or when the grid is not aligned with the local principal curvatures.)

Both were identified first in the context of subdivision surfaces, (see Sabin and Barthe [4]) and we shall use the subdivision approach to splines to analyse them, but they are equally applicable to B-spline surfaces used explicitly.

1.2 Prior Knowledge

Longitudinal artifacts are present in all splines. They reduce with control point density, at a rate which depends on the degree. They are more familiarly thought of in terms of the approximation error, which is of order $O(h^{d+1})$ where d is the degree of the polynomial pieces, meeting with continuity of $d-1^{\text{th}}$ derivative at the knots. This is closely related to the fact that the splines contain the polynomials of degree d as their precision set.

Thus although the approximation error drops quickly with the control point density, especially for high degrees, some artifact component is always present. In the curve and surface design context we do not want to be forced to use dense control points, and so the factor multiplying the $O(h^{d+1})$ is also highly relevant.

Lateral artifacts again afflict all spline surfaces, except that there can be certain directions in which features intended to be more or less extruded can be run safely, without generating lateral artifacts. Using subdivision-coloured spectacles, we can see that these directions are exactly those in which the subdivision mask has at least one $1+z$ factor. The Peters-Shiue 4-3 subdivision scheme [3] is a box-spline with two $1+z$ factors in each of the main parametric directions and one in each of the two diagonal directions, and the fact that it can support extruded features running diagonally is one of its main advantages.

If there are directions with two $1+z$ factors, then the cross-section of an extruded feature can grow or shrink linearly along the feature without corruption. Features in directions with three or more can vary quadratically etc.

This has all been obvious enough in the tensor product B-spline context that no analysis seems to have been published. This has not stopped attempts at the building of surface editing packages which can introduce features running skew to the safe directions by modifying control points using some 'hyper basis' function.

It is also reasonably widely understood that when the surface is constructed as the limit surface of a subdivision process, most of the damage is done at the first step.

1.3 The Fourier Approach

The approach we take here is to examine the response of the configuration (i.e. the polygon or polyhedron) after one step to that before the step. Because we assume linearity of the system we can separate the initial polyhedron into components of different spatial frequency, and look at the response of each component as a function of its frequency. This is in some ways equivalent to looking at the approximation error as a function of the maximum spacing between two data points: an early exploration of this route appears in [5].

1.4 Notation

The notation $T^x, \mathbb{R} \to \mathbb{C}$ denotes the function $e^{i\pi x} = \cos(\pi x) + i\sin(\pi x)$. Superscript notation retains the concept of exponentiation. Although this could be interpreted as $T^x = e^{i\pi x} = (e^{i\pi})^x$ so that $T = e^{i\pi} = -1$ and $T^x = -1^x$, it is more relevant to think of T as a function, than as a number.

The properties of T^x which are used are

$$\begin{aligned}
T^0 &= 1,\\
T^{1/2} &= i,\\
T^{-1/2} &= -i,\\
T^1 = T^{-1} &= -1,\\
T^{x+y} &= T^x T^y,\\
T^x + T^{-x} &= 2\cos(\pi x),\\
T^x - T^{-x} &= 2i\sin(\pi x).
\end{aligned}$$

2 Curves

2.1 Matrices and z-Transforms

We have two ways of looking at univariate subdivision schemes. One is via the z-transform, the other via subdivision matrices. Analysis of longitudinal artifacts has exposed a relationship between the two.

The subdivision matrix is applied to a vector of points at old places, which (in a binary scheme) are at alternate new places, say the even places. When we convolve with the mask, we can regard the summation of masks times old points as actually being summation of masks with old points at the even places, and with zero values at the intervening odd places.

We can then look at the matrix as being the product of a circulant matrix with unit slope, with a standard matrix with unit values on a diagonal having slope $= -2$:

$$\begin{bmatrix} \ddots & & & & \\ & c & a & & \\ & b & b & & \\ & a & c\,a & & \\ & & b\,b & & \\ & & a\,c & a & \\ & & & b & b \\ & & & a & c \\ & & & & \ddots \end{bmatrix} \equiv \begin{bmatrix} \ddots & & & & & & & \\ & c & b & a & & & & \\ & b & c & b & a & & & \\ & a & b & c & b & a & & \\ & & a & b & c & b & a & \\ & & & a & b & c & b & a \\ & & & & a & b & c & b \\ & & & & & a & b & c \\ & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \ddots & & & & \\ & 1 & & & \\ & 0 & 0 & & \\ & 0 & 1 & & \\ & & 0 & 0 & \\ & & & 1 & \\ & & & 0 & 0 \\ & & & & 1 \\ & & & & \ddots \end{bmatrix} .$$

We call the circulant matrix the *filter matrix* and the steep diagonal the *sampling matrix.* In fact it is more convenient to multiply the sampling matrix by 2 and divide the filter matrix by 2, so that the latter has the property that its rows sum to unity.

All circulant matrices have a unique (up to a scaling) factorisation into circulant factors (see Davis [1–page 68]), and these factors turn out to have rows (or columns) which are exactly the factors of the z-transform. If we also require that all factors have the property that the rows sum to unity the factorisation is unique. These factors commute, and so we can take them in any sequence.

Because of the significance of the $(1+z)$ factors seen through the z-transform telescope, we see that the filter matrix can be factorised into a number of factors equal to

$$\frac{1}{2}\begin{bmatrix} \ddots & & & & & & \\ & 1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & 1 & 1 & & \\ & & & & 1 & 1 & \\ & & & & & 1 & 1 \\ & & & & & & \ddots \end{bmatrix}$$

which we call the *smoothing matrices*, multiplied by some further factor, which we call the *kernel*, which may or may not have a further factorisation, but certainly has no further $1+z$ factors. The kernel and the smoothing matrices all have the property that the rows sum to unity. B-splines have only smoothing factors, the kernel being an identity matrix.

This set of matrices can then be applied (actually or theoretically) to the result of multiplying the configuration by the sampling matrix in any order. We can choose whether to regard the kernel as the leftmost factor (applied last, to the result of smoothing the sampled original data) of the circulant part, or as the rightmost (applied first to the sampled data which is then smoothed). It does not matter: these matrices commute.

We can now address the two important questions concerning the longitudinal artifact. How rapidly does the longitudinal artifact die away as the sampling density is increased? How large is it at any given sampling density? The latter

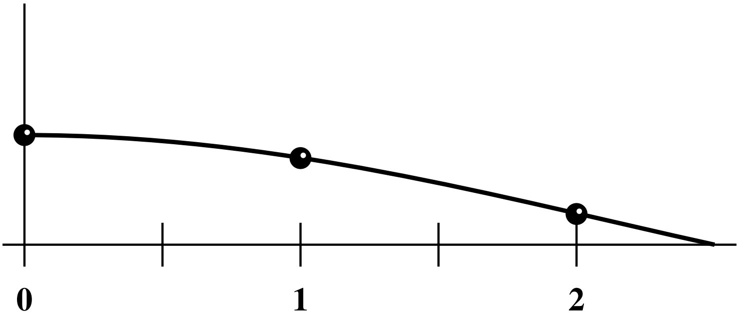


Fig. 2. Input values D_j with a frequency ω

is particularly important in practice, because we explicitly want to be able to sample as sparsely as possible.

2.2 The Effect of the Sampling Matrix

This is very simple. The result of multiplying the original sequence by the sampling matrix is to produce a new sequence with the original values doubled and zero values inserted in between.

In order to analyse the behaviour we let the original data, D_j, where j counts through the original control points, be sampled from a sine wave of frequency ω, measured in units of complete cycles per original vertex (the reciprocal of vertices per cycle). An actual data set can be regarded as the sum of such components, and by linearity the effect on the total is the sum of the effects on the separate spatial frequency components.

The magnitude of ω is less than 1/2 by the Shannon limit, because we need at least two control points per cycle to define the variation, and typically $\omega \leq 1/4$, assuming that there are reasonably about four control points per cycle. (Current NURBS practice uses far more and so ω is even smaller.) Then multiplication by the sampling matrix gives the sequence S, where

$$\begin{aligned} S_{2j} &= 2D_j = 2e^{2i\pi\omega j} = 2\left(\cos(2\pi\omega j) + i\sin(2\pi\omega j)\right) = 2T^{2\omega j)} \\ S_{2j+1} &= 0 \end{aligned}$$

which can be written more concisely as

$$\begin{aligned} S_p &= 2T^{\omega p}(1 + cos(\pi p))/2, \quad p \in \mathbb{Z} \\ \text{or as} \quad S_p &= T^{\omega p}\left(1 + (T^p + T^{-p})/2\right) \end{aligned}$$

where p counts through the control points after refinement.

In general we expect, by the symmetry of the situation, to see that after any amount of smoothing the result R will be of the form

$$\begin{aligned} R_p &= T^{2\omega p}\left(\alpha + \beta(T^p + T^{-p})/2\right) \\ &= \alpha T^{2\omega p} + \beta(T^{(2\omega+1)p} + T^{(2\omega-1)p})/2 \end{aligned}$$

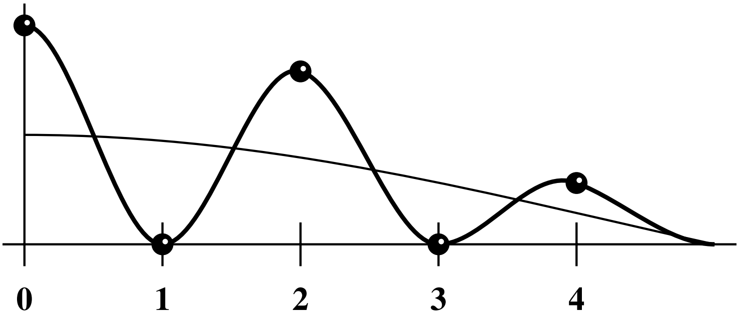


Fig. 3. Samples S_p with intermediate zeroes

where the symmetry of the β terms is derived from the mirror symmetry of the mask. We call the first term the *signal term*, the second the *artifact term*, because the change in α can be compensated for by just scaling the original polygon; the presence of non-zero β cannot. For the unfiltered signal, $R_p = S_p$ so that initially $\alpha = \beta = 1$. Each factor in the filter matrix then affects the values of α and β by multiplying them by some factors depending on the value of ω.

The question is then to see what effect a single [1 1]/2 matrix has on the values of α and β. However, because it turns out that the square of this factor has a simpler effect, we look first at the effect of the square, a [1 2 1]/4 matrix.

B-splines of odd degree have an even number of [1 1]/2 factors and can be understood solely in terms of [1 2 1]/4 matrices. Those of even degree can be understood in terms of a number of [1 2 1]/4 matrices followed by a single [1 1]/2 matrix.

2.3 The Effect of a [1 2 1]/4 Matrix

Our original samples, with the zeroes in between, can be described as

$$S_p = T^{\omega p} + T^{\omega p}\left(T^p + T^{-p}\right)/2$$

Consider first the signal term, $T^{\omega p}$. The convolution of this with [1 2 1]/4 is

$$\begin{aligned}
&\left(T^{\omega(p-1)} + 2T^{\omega p} + T^{\omega(p+1)}\right)/4 \\
&= \left(T^{\omega p}T^{-\omega} + 2T^{\omega p} + T^{\omega p}T^{+\omega}\right)/4 \\
&= T^{\omega p}\left(T^{-\omega} + 2 + T^{+\omega}\right)/4 \\
&= T^{\omega p}\left(2 + 2\cos(\pi\omega)\right)/4 \\
&= T^{\omega p}\cos^2(\pi\omega/2).
\end{aligned}$$

Thus α is multiplied by $\cos^2(\pi\omega/2)$.

Now consider the artifact term, $T^{\omega p}\left(T^p + T^{-p}\right)/2$. The convolution of this with [1 2 1]/4 is

$$\begin{aligned}
&\left(T^{\omega(p-1)}(T^{p-1}+T^{-(p-1)}) + 2T^{\omega(p)}(T^p+T^{-(p)}) + T^{\omega(p+1)}(T^{p+1}+T^{-(p+1)})\right)/8\\
&= \left(T^{\omega p}T^{-\omega}(T^{p-1}+T^{-(p-1)}) + 2T^{\omega p}(T^p+T^{-(p)}) + T^{\omega p}T^{\omega}(T^{p+1}+T^{-(p+1)})\right)/8\\
&= T^{\omega p}\left(T^{-\omega}(T^{p-1}+T^{-(p-1)}) + 2(T^p+T^{-(p)}) + T^{\omega}(T^{p+1}+T^{-(p+1)})\right)/8\\
&= T^{\omega p}\left(T^{-\omega}(T^pT^{-1}+T^{-p}T^{-1}) + 2(T^p+T^{-(p)}) + T^{\omega}(T^pT^{+1}+T^{-p}T^{-1})\right)/8
\end{aligned}$$

Substituting for $T^1 = T^{-1} = -1$ this becomes

$$\begin{aligned}
&T^{\omega p}\left(T^{-\omega}(-T^p - T^{-p}) + 2(T^p+T^{-(p)}) + T^{\omega}(-T^p - T^{-p})\right)/8\\
&= T^{\omega p}\left(+T^p(2-T^{\omega}-T^{-\omega}) + T^{-p}(2-T^{\omega}-T^{-\omega})\right)/8\\
&= T^{\omega p}(T^p+T^{-p})(2-T^{\omega}-T^{-\omega})/8\\
&= T^{\omega p}(T^p+T^{-p})(2-2\cos(\pi\omega))/8\\
&= T^{\omega p}(T^p+T^{-p})\sin^2(\pi\omega/2)/2
\end{aligned}$$

Thus β is multiplied by $\sin^2(\pi\omega/2)$.

This is all consistent with the classical results that for B-splines of odd degree, which have an even number $d+1$ of $1+z$ factors in their mask, the approximation errors from the artifact part reduce as $O(\omega^{d+1})$, while those from the signal part remain at $O(\omega^2)$. If ω is measured in samples per cycle, it is exactly analogous to the classical h (see [4] for an earlier exploration of this).

2.4 The Effect of a [1 1]/2 Matrix

Our original samples, with the zeroes in between, can be described as

$$S_p = T^{\omega p} + T^{\omega p}\left(T^p + T^{-p}\right)/2$$

Consider first the signal term, $T^{\omega p}$. The convolution of this with [1 1]/2 is

$$\begin{aligned}
&\left(T^{\omega(p-1/2)} + T^{\omega(p+1/2)}\right)/2\\
&= \left(T^{\omega p}T^{-\omega/2)} + T^{\omega p}T^{+\omega/2)}\right)/2\\
&= T^{\omega p}\left(T^{-\omega/2)} + T^{+\omega/2)}\right)/2\\
&= T^{\omega p}\cos(\pi\omega/2)
\end{aligned}$$

Thus α is multiplied by $\cos(\pi\omega/2)$.

Now consider the artifact term, $T^{\omega p}\left(T^p + T^{-p}\right)/2$. The convolution of this with [1 1]/2 is

$$\begin{aligned}
&\left(T^{\omega(p-1/2)}(T^{p-1/2}+T^{-(p-1/2)}) + T^{\omega(p+1/2)}(T^{p+1/2}+T^{-(p+1/2)})\right)/4\\
&= \left(T^{\omega p}T^{-\omega/2}(T^pT^{-1/2}+T^{-p}T^{1/2}) + T^{\omega p}T^{\omega/2}(T^pT^{1/2}+T^pT^{-1/2})\right)/4\\
&= T^{\omega p}\left(T^{-\omega/2}T^pT^{-1/2} + T^{-\omega/2}T^{-p}T^{1/2} + T^{\omega/2}T^pT^{1/2} + T^{\omega/2}T^pT^{-1/2}\right)/4
\end{aligned}$$

Substituting for $T^{1/2} = -T^{-1/2} = i$ this becomes

$$\begin{aligned} & iT^{\omega p}\left(-T^{-\omega/2}T^{p} + T^{-\omega/2}T^{-p} + T^{\omega/2}T^{p} - T^{\omega/2}T^{p}\right)/4 \\ &= iT^{\omega p}(T^{\omega/2} - T^{-\omega/2})(T^{p} - T^{-p})/4 \\ &= i^{2}T^{\omega p}((T^{p} - T^{-p})/2)\sin(\pi\omega/2) \\ &= -T^{\omega p}((T^{p} - T^{-p})/2)\sin(\pi\omega/2) \end{aligned}$$

Thus the artifact magnitude is multiplied by $-\sin(\pi\omega/2)$ but, because the second factor is $(T^{p} - T^{-p})$ instead of $(T^{p} + T^{-p})$, the artifact phase is also altered, so that the actual error in the new control point positions is along the direction of the curve, rather than perpendicular to it, thus accounting for the apparent anomaly noted in [4] that the geometric artifacts from B-splines of even degree shrank at the same rate as those of the next degree higher. The quantitative analysis of this is beyond the scope of this paper.

2.5 The Effect of the Kernel

It is quite possible to determine the effect of the kernel in exactly the same way as we determined the effect of [1 2 1]/4, but there is a significantly simpler way, which leads to more insight, and even to a way of designing kernels to have desirable properties.

First observe that the kernel is palindromic, and that, because we have divided out all $1 + z$ factors it has no factors of $1 + z$. It must therefore have an odd number of entries, because if it had an even number it would be divisible by $1 + z$.

Divide by an appropriate power of z so that the central value is the one corresponding to z^0, and do likewise with the [1 2 1]/4 generating polynomial. (This process is legitimate, because multiplying or dividing by z merely shifts our labelling along the polygon. If it is carried out with the mask expressed just as a sequence of numbers you never even notice that it is happening.)

$$\begin{aligned} K(z) &= \ldots cz^{-2} + bz^{-1} + az^{0} + bz^{1} + cz^{2} \ldots \\ S(z) &= [z^{-1} + 2z^{0} + z^{1}]/4 \end{aligned}$$

Now express $K(z)$ as a polynomial in $S(z)$. This will always be a finite polynomial, because $K(z)$ is itself finite.

For example, the kernel of the four-point scheme [2] is [-1 4 -1]/2, which can be expressed as

$$\begin{aligned} [-1\ 4\ -1] &= -[1\ 2\ 1] + 6[0\ 1\ 0] \\ [-1\ 4\ -1]/2 &= -[1\ 2\ 1]/2 + 3[0\ 1\ 0] \\ &= -2[1\ 2\ 1]/4 + 3[0\ 1\ 0] \\ &= 3 - 2S[z] \end{aligned}$$

More complex (larger) kernels will be a polynomial in S of higher degree.

We can now identify the effects that each term of this polynomial has on the signal and the artifact, and then just sum those effects. The constant term multiplies both signal and artifact by the same constant: the linear term multiplies the signal by $\cos^2(\pi\omega/2)$ and the artifact by $\sin^2(\pi\omega/2)$ etc. We can therefore write down polynomials in these factors to get the effect of the kernel.

In the four-point case we have a net factor of $3 - 2\cos^2(\pi\omega/2)$ for the signal and $3 - 2\sin^2(\pi\omega/2)$ for the artifact. As ω tends to zero we find that the factor tends to 1 for the signal and 3 for the artifact. These are then multiplied by the factors from the $1 + z$ factors to give the overall artifact and signal sizes for the scheme.

An exercise for the reader is to determine the Taylor series for the signal and artifact in terms of ω. It will then be seen that the total effect of the four-point scheme on the signal varies as $1 + O(\omega^4)$ rather than as $1 + O(\omega^2)$ (as might be expected from the cosine terms from the [1 2 1]/4 factors, and that the actual difference from 1 is equal in magnitude to the effect on the artifact. This is exactly what would be expected from an interpolating scheme.

We can now deliberately design kernels to have specific effects. The first option is that the kernel could be designed to have a signal factor exactly compensating for the smoothing factors, so that the mean limit curve would be a circle passing through the data points, with the artifact giving equal perturbations inside it and outside. This would be exact only for one specific spatial frequency.

The other possibility is that the artifact could be cancelled out exactly for a specific spatial frequency, by choosing the kernel polynomial to have roots within the 0..1 range, so that when ω was such that $\sin^2(\pi\omega/2)$ lay on a root the artifact factor would be exactly zero. Those circle-preserving subdivision schemes which need to know the number of vertices initially on a circle, can usefully be looked at in this light.

3 Surfaces

The analysis here follows closely the univariate analysis. The first notational difference is that abscissa positions and spatial frequencies are now bivariate objects. We use upper case, P and Ω, respectively to denote them. The important operator on them is the inner product, which gives a scalar result.

The bivariate spatial frequency Ω is represented as a vector with two components, which we call ω_x and ω_y. In fact ω_x measures the number of complete cycles per point along the x axis, so that it is zero when Ω is pointing along the y-axis.

The second notational difference is that we have to denote the shifts within the grid: it is no longer adequate to use just +1 and -1 as shifts. We use the symbols X and Y to denote the unit shifts in the abscissa plane along the two grid directions of a rectangular grid. Thus $X + Y$ and $X - Y$ become shifts in diagonal directions.

We assume without proof the validity of the partitioning of the subdivision process into first a sampling, with insertion of zero elements at all new positions, followed by a series of convolutions with factors of the mask. This is highly plausible for tensor product constructions. The same partitioning is believed to be valid also for schemes which are not tensor products, but matrix notation becomes a clumsy tool for bivariate subdivision, when rows and columns (**stencils** and *masks*, respectively) are both themselves arrays of coefficients.

On a rectangular grid the sampling process may be viewed as a tensor product:

$$\begin{aligned}
S_P &= T^{\Omega.P}(T^{X.P}+2+T^{-X.P})(T^{Y.P}+2+T^{-Y.P})/4\\
&= T^{\Omega.P}\\
&\quad +T^{\Omega.P}(T^{X.P}+T^{-X.P})/2\\
&\quad +T^{\Omega.P}(T^{Y.P}+T^{-Y.P})/2\\
&\quad +T^{\Omega.P}(T^{(X+Y).P}+T^{-(X+Y).P})/4\\
&\quad +T^{\Omega.P}(T^{(X-Y).P}+T^{-(X-Y).P})/4.
\end{aligned}$$

There therefore exist a signal component, and four potential artifact components, each of the form $T^{\Omega.P}(T^{V.P}+T^{-V.P})/2$, V being directed along a grid edge or along a diagonal of the grid.

Thus when we consider a signal which is not oriented along or perpendicular to the grid edges, the artifacts, which remain aligned relative to the grid can no longer be described as purely longitudinal or purely lateral.

Consider first the signal term, $T^{\Omega.P}$. The convolution of this with [1 2 1]/4 in direction D, (where D is X, Y, $X+Y$ or $X-Y$) is

$$\begin{aligned}
&\left(T^{\Omega.[P-D]}+2T^{\Omega.P}+T^{\Omega.[P+D]}\right)/4\\
&= T^{\Omega.P}\left(T^{-\Omega.D}+2+T^{\Omega.D}\right)/4\\
&= T^{\Omega.P}\cos^2(\pi\Omega.D/2)
\end{aligned}$$

This behaviour is exactly analogous to a curve's longitudinal signal behaviour if Ω and D lie in the same direction, while no scaling happens if Ω is perpendicular to D.

Consider next one of the artifact components. Each of these can be expressed as $T^{\Omega.P}(T^{V.P}+T^{-V.P})/2$ (where V is X, Y, $X+Y$ or $X-Y$). The convolution of this with [1 2 1]/4 in direction D is

$$\begin{aligned}
&\Big(T^{\Omega.[P-D]}(T^{V.[P-D]}+T^{-V.[P-D]})+2T^{\Omega.P}(T^{V.P}+T^{-V.P})\\
&\quad +T^{\Omega.[P+D]}(T^{V.[P+D]}+T^{-V.[P+D]})\Big)/8\\
= \;& T^{\Omega.P}\left(T^{-\Omega.D}(T^{V.P}T^{-V.D}+T^{-V.P}T^{V.D})+2(T^{V.P}+T^{-V.P})\right.\\
&\quad \left.+T^{\Omega.D}(T^{V.P}T^{V.D}+T^{-V.P}T^{-V.D})\right)/8
\end{aligned}$$

Because both V and D are taken from the set $\{X, Y, X+Y, X-Y\}$, there are now only two cases to consider. In one case, when $V.D=0$ or $V.D=2$, $T^{V.D}=T^{-V.D}=1$. In the other case, when $V.D=1$, $T^{V.D}=T^{-V.D}=-1$.

In the first case the above expression simplifies to

$$
\begin{aligned}
&T^{\Omega.P}\left(T^{-\Omega.D}(T^{V.P}+T^{-V.P})+2(T^{V.P}+T^{-V.P})+T^{\Omega.D}(T^{V.P}+T^{-V.P})\right)/8\\
&=T^{\Omega.P}(T^{V.P}+T^{-V.P})\left(T^{-\Omega.D}+2+T^{\Omega.D}\right)/8\\
&=T^{\Omega.P}(T^{V.P}+T^{-V.P})\cos^2(\pi\Omega.D/2)/2
\end{aligned}
$$

so that the artifact component is multiplied by $\cos^2(\pi\Omega.D/2)$.

In the second case similar manipulation shows that the artifact component is multiplied by $\sin^2(\pi\Omega.D/2)$.

These results can now be collected together into a table. To make the table fit the page we write $\sin(\pi\omega_x/2)$ as S_x and $\cos(\pi\omega_x/2)$ as C_x:

	$V=X$	$V=Y$	$V=X+Y$	$V=X-Y$
$D=X$	S_x^2	C_y^2	S_{x+y}^2	S_{x-y}^2
$D=Y$	C_x^2	S_y^2	S_{x+y}^2	S_{x-y}^2
$D=X+Y$	S_x^2	S_y^2	C_{x+y}^2	C_{x-y}^2
$D=X-Y$	S_x^2	S_y^2	C_{x+y}^2	C_{x-y}^2

3.1 Examples

The values from the table above are now used to determine the artifact magnitudes for some of the standard quad-grid schemes. The Bicubic box spline has two [1 2 1]/4 factors in each of the X and Y directions; the box-spline used in Velho's 4-8 scheme [7] has one [1 2 1]/4 factor in each of the four directions X, Y, $X+Y$ and $X-Y$; and the box-spline used in Peters and Shiue's 4-3 scheme [3] has one such factor in each of X and Y and a [1 1]/2 factor in each of the diagonal directions. We can therefore determine the artifact magnitudes by just multiplying together the appropriate number of factors from the above table.

	$V=X$	$V=Y$	$V=X+Y$	$V=X-Y$
Bicubic	$S_x^4C_x^4$	$S_y^4C_y^4$	S_{x+y}^8	S_{x-y}^8
$4-8$	$S_x^6C_x^2$	$S_y^6C_y^2$	$S_{x+y}^4C_{x+y}^4$	$S_{x-y}^4C_{x-y}^4$
$4-3$	$S_x^4C_x^2$	$S_y^4C_y^2$	$S_{x+y}^4C_{x+y}^2$	$S_{x-y}^4C_{x-y}^2$

We plot the results of the above computations by using hatching of various densities on an abscissa of Ω. Each of the figures shows the regions where the sum of the four artifacts is greater than 0.1 (dense hatching), between 0.01

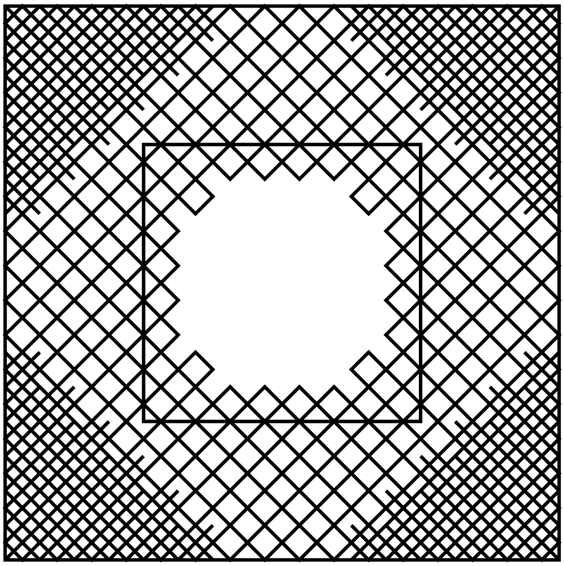


Fig. 4. Total artifacts as a function of spatial frequency for the bicubic B-spline

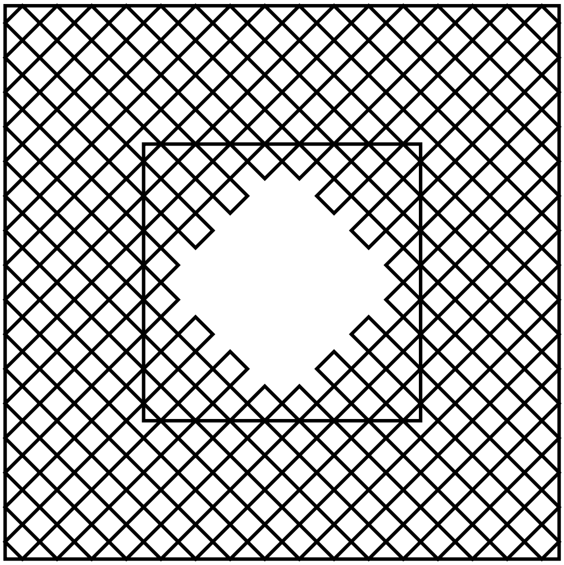


Fig. 5. Total artifacts as a function of spatial frequency for the Four-direction box-spline with two shifts in each direction

and 0.1 (lighter hatching) and less than 0.01 (unhatched). The sum is a crude measure, but serves as a general measure of quality. Each square covers the region $-1/2 \leq \omega_x, \omega_y \leq 1/2$, but the realistic region is the central quarter of this area, where $-1/4 \leq \omega_x, \omega_y \leq 1/4$.

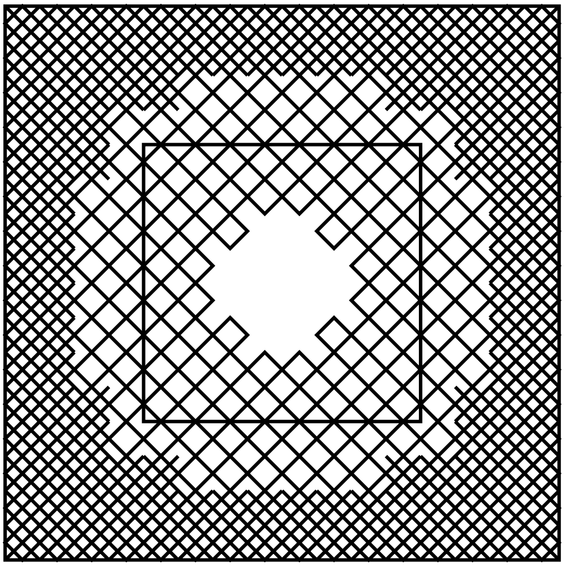


Fig. 6. Total artifacts as a function of spatial frequency for the Four-direction box-spline with two shifts in axis directions and one shift in each diagonal direction. It is possible that the phase-shift effect from [1 1] factors, mentioned above, which reduces the geometric artifact for dual schemes, may also apply in this case, making this plot a pessimistic one, but this is far from certain, because this scheme is primal

4 Conclusions

The analysis of longitudinal and lateral artifacts in spline surfaces and in other subdivision surfaces has been shown to be systematic and straightforward over a regular rectangular grid. This is closely related to the spectral analysis of wavelets, but has its own quirks resulting from the symmetry of the systems involved.

The known results about approximation orders have been confirmed through a different route, which gives actual artifact sizes from the number of vertices per cycle, not just rates of convergence.

Indeed, the only surprise emerging from this work is that it has not been standard material in courses on subdivision for many years.

4.1 Further Work

We have not studied here either general arities or splines over triangulations. We expect the same techniques to be applicable, and similar results to emerge. For the B-spline and box-spline cases the ternary and higher arities should, of course, give exactly the same results as for the binary case, because the artifact is a property of the relationship between the control polygon and the limit curve, not of the subdivision by which it is implemented.

Acknowledgements. The work described above was supported by EPSRC under grant number GR/S67173/01.

References

1. P.Davis, *Circulant Matrices.* Wiley Interscience, 1979.
2. N.Dyn, J.Gregory and D.Levin, *A four-point interpolatory subdivision scheme for curve design*, CAGD 4, pp 257–268, 1987.
3. J.Peters and L-J.Shiue,*4-3 Directionally Ripple-free Subdivision*, ACM ToG, 23(4), pp980–1003, 2004.
4. M.Sabin and L.Barthe, *Artifacts in Recursive Subdivision Surfaces*, pp 353–362 in *Curve and Surface Fitting:St.Malo 2003*, ed. Cohen, Merrien and Schumaker, Nashboro Press, 2003.
5. M.Sabin, *ω-convergence, A criterion for linear approximation*, pp 415–420 in *Curves and Surfaces* ed. Laurent, Le Méhauté and Schumaker. Academic Press, 1991.
6. C.Shannon and W.Weaver,*The Mathematical Theory of Communication*, University of Illinois Press, 1949.
7. L.Velho and D.Zorin, *4-8 subdivision*, CAGD 18, pp 397–427, 2001
8. J.Warren and H.Weimer *Subdivision Methods for Geometric Design*, Morgan Kaufmann, 2002.

Spatial Pythagorean Hodograph Quintics and the Approximation of Pipe Surfaces

Zbyněk Šír and Bert Jüttler

Johannes Kepler University, Institute of Applied Geometry,
Altenberger Str. 69, 4040 Linz, Austria
{zbynek.sir, bert.juettler}@jku.at
http://www.ag.jku.at/

Abstract. As observed by Farouki et al. [9], any set of C^1 space boundary data (two points with associated first derivatives) can be interpolated by a Pythagorean hodograph (PH) curve of degree 5. In general there exists a two dimensional family of interpolants.

In this paper we study the properties of this family in more detail. We introduce a geometrically invariant parameterization of the family of interpolants. This parameterization is used to identify a particular solution, which has the following properties. Firstly, it preserves planarity, i.e., the interpolant to planar data is a planar PH curve. Secondly, it has the best possible approximation order (4). Thirdly, it is symmetric in the sense that the interpolant of the "reversed" set of boundary data is simply the "reversed" original interpolant. These observations lead to a fast and precise algorithm for converting any (possibly piecewise) analytical curve into a piecewise PH curve of degree 5 which is globally C^1.

Finally we exploit the rational frames associated with any space PH curve (the Euler-Rodrigues frame) in order to obtain a simple rational approximation of pipe surfaces with a piecewise analytical spine curve and we analyze its approximation order.

1 Introduction

Pythagorean hodograph (PH) curves (see the survey [11] and the references cited therein), form a remarkable subclass of polynomial parametric curves. They have a piecewise polynomial arc length function and, in the planar case, rational offset curves. These curves provide an elegant solution of various difficult problems occurring in applications, in particular in the context of CNC (computer-numerical-control) machining.

In the planar case, the properties and various constructions of PH curves have been thoroughly studied, e.g., [1, 6, 8, 7, 18, 23]. Due to the constrained nature of PH curves, all constructions – which are linear in the case of polynomial curves – become *nonlinear* in the PH case. Consequently, they may have more than one solution, and the problem of choosing the 'best' solution has to be addressed, e.g. by analyzing the approximation order or using the rotation index [15, 18, 20, 21, 22].

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 364–380, 2005.

Spatial PH curves were introduced by Farouki and Sakkalis in 1994 [5], and they have later been characterized using results about Pythagorean quadruples in the ring of polynomials and quaternion calculus [2, 4, 10]. Spatial PH curves can be equipped with rational frames, which were studied in [3, 13, 17].

Various constructions were also given, e.g. a global method for C^2 interpolation of point data by quintic splines has been presented in [12]. Hermite interpolation of G^1 boundary data was addressed in [17], and C^1 Hermite interpolation by PH quintics was discussed in [9]. In the latter case, the authors identify a family of interpolants to any C^1 Hermite data which depends on two free parameters, and a heuristic choice for them is given. Later, this has also been related to helical interpolants [14].

The present paper is devoted to the problem of C^1 Hermite interpolation by spatial PH quintics, and to the approximation of pipe surfaces and sweeping surfaces. We study the family of interpolants and identify the solution which has the best approximation order, preserves planarity, and is symmetric with respect to the reversion $t \mapsto (1-t)$ of the parameter interval $[0, 1]$.

The remainder of the paper is organized as follows. First we recall some basic facts about quaternion algebra and PH curves. The first part of Section 3 summarizes the approach taken in [9] to the problem of C^1 Hermite interpolation by PH quintics. In the second part we introduce a parameterization of the family of interpolants with respect to a standard position. We prove that this parameterization is geometrically invariant and symmetric.

Section 4 provides a qualitative analysis of the solutions. We give an asymptotical analysis, including approximation order, and we identify the parameter values which preserve planarity. Based on these results, we use optimal solution for converting analytical curves into piecewise PH quintic curves and for the approximation of pipe surfaces. Finally we conclude the paper.

2 Preliminaries

In order to make this paper self–contained, we recall some basic facts about quaternions and Pythagorean Hodograph curves.

2.1 Quaternions

Quaternions (see e.g. [19] for an elementary introduction) are elements

$$\mathcal{A} = a + a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k} \tag{1}$$

of 4–dimensional real linear space $\mathbb{Q}$ with basis $1, \mathbf{i}, \mathbf{j}, \mathbf{k}$. The space $\mathbb{Q}$ has the structure of a non-commutative field, where the multiplication is defined by the relations

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{2}$$

of the basis elements, which imply

$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \quad \mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \quad \mathbf{ki} = -\mathbf{ik} = \mathbf{j}. \tag{3}$$

The conjugate of any quaternion (1) is defined as $\mathcal{A}^* = a - a_x\mathbf{i} - a_y\mathbf{j} - a_z\mathbf{k}$, and its absolute value is the non-negative real number

$$|\mathcal{A}|^2 = \sqrt{\mathcal{A}\mathcal{A}^*} = \sqrt{\mathcal{A}^*\mathcal{A}} = \sqrt{a^2 + a_x^2 + a_y^2 + a_z^2}. \qquad (4)$$

Unit quaternions, which are characterized by $|\mathcal{A}| = 1$, form a multiplicative group. Pure quaternions are distinguished by having a vanishing scalar part.

Quaternions are traditionally used in classical mechanics. Any vector $\mathbf{c} = [c_x, c_y, c_z]^\top \in \mathbb{R}^3$ is identified with the pure quaternion $c_x\mathbf{i} + c_y\mathbf{j} + c_z\mathbf{k}$. Any unit quaternion $\mathcal{U}$ can be expressed in the form

$$\mathcal{U} = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2}, \quad \theta \in [-\pi, \pi), \qquad (5)$$

where $\mathbf{u}$ is a unit pure quaternion. Then the mapping

$$U : \mathbb{R}^3 \to \mathbb{R}^3 : U(\mathbf{c}) = \mathcal{U}\,\mathbf{c}\,\mathcal{U}^*, \qquad (6)$$

represents a rotation through the angle θ about the axis spanned by the direction vector $\mathbf{u}$.

In the sequel we will use the the abbreviation

$$\mathcal{Q}(\phi) = (\cos\phi + \mathbf{i}\sin\phi) \qquad (7)$$

for unit quaternions with vanishing $\mathbf{j}$ and $\mathbf{k}$ components.

For the construction of PH Hermite interpolants which is described below, the following Lemma proved in [9–section 3.2] is essential.

Lemma 1. *For a given pure quaternion* $\mathbf{c}$*, which is not a negative multiple of* $\mathbf{i}$*, all solutions of the equation*

$$\mathcal{A}\,\mathbf{i}\,\mathcal{A}^* = \mathbf{c} \qquad (8)$$

are expressed as

$$\mathcal{A}(\phi) = \sqrt{|\mathbf{c}|}\,\frac{\frac{\mathbf{c}}{|\mathbf{c}|} + \mathbf{i}}{\left|\frac{\mathbf{c}}{|\mathbf{c}|} + \mathbf{i}\right|}\,\mathcal{Q}(\phi), \quad \phi \in [0, 2\pi). \qquad (9)$$

If $\mathbf{c}$ is a negative multiple of $\mathbf{i}$, then a suitable limit of the formula (9) must be taken.

2.2 Pythagorean Hodograph Curves

The *hodograph* of a space curve $\mathbf{p}(t) = [x(t), y(t), z(t)]^\top$ of degree n is the curve $\mathbf{h}(t) = [x'(t), y'(t), z'(t)]^\top$ of degree $n - 1$, where $'$ denotes the first derivative. Recall that a polynomial curve is called *Pythagorean Hodograph (PH)*, if the length of its tangent vector depends in a (piecewise) polynomial way on the parameter. In particular $\mathbf{p}(t) = [x(t), y(t), z(t)]^\top$ is called *space PH curve* if there exists a polynomial $\sigma(t)$ such that

$$x'(t)^2 + y'(t)^2 + z'(t)^2 = \sigma^2(t). \qquad (10)$$

If $\gcd(x'(t), y'(t), z'(t))$ is a square, then equation (10) holds if and only if there exist polynomials $u(t), v(t), p(t), q(t)$ such that

$$\begin{aligned} x'(t) &= u^2(t) + v^2(t) - p^2(t) - q^2(t), \\ y'(t) &= 2u(t)q(t) + 2v(t)p(t), \\ z'(t) &= 2v(t)q(t) - 2u(t)p(t), \\ \sigma(t) &= u^2(t) + v^2(t) + p^2(t) + q^2(t), \end{aligned} \tag{11}$$

see [4]. This result can be reformulated using quaternions [2, 10]. Any spatial polynomial curve $\mathbf{p}(t) = [x(t), y(t), y(t)]^\top$ is identified with the pure–quaternion–valued function $\mathbf{p}(t) = x(t)\mathbf{i}+y(t)\mathbf{j}+z(t)\mathbf{k}$. The PH curves are then characterized as follows.

Lemma 2. *Let* $\mathbf{p}(t) = x(t)\mathbf{i}+y(t)\mathbf{j}+z(t)\mathbf{k}$ *be a space polynomial curve, such that* $\gcd(x'(t), y'(t), z'(t))$ *is the square of a polynomial*[1]. *Then* $\mathbf{p}(t)$ *is PH if and only if there exists a quaternion-valued polynomial* $\mathcal{A}(t) = u(t) + v(t)\mathbf{i} + p(t)\mathbf{j} + q(t)\mathbf{k}$ *such that*

$$\mathbf{h}(t) = \mathcal{A}(t)\,\mathbf{i}\,\mathcal{A}^*(t). \tag{12}$$

The arc length function of the PH curve is a polynomial obtained by integrating $|\mathcal{A}(t)|^2 = \mathcal{A}(t)\mathcal{A}^*(t)$.

Consequently, the construction of a PH curve is reduced to the construction of a suitable curve $\mathcal{A}(t)$. This curve will be called the *preimage.*

3 C^1 Hermite Interpolation by Space Quintics

Following [9], we construct a spatial PH curve $\mathbf{p}(t)$ which matches given C^1 Hermite boundary data. More precisely, the curve is to interpolate the end points $\mathbf{p}_0$, $\mathbf{p}_1$ and the tangent vectors (or derivation vectors) $\mathbf{t}_0$, $\mathbf{t}_1$. The cases $\mathbf{t}_0 = 0$ or $\mathbf{t}_1 = 0$, and $\mathbf{t}_0 = -\mathbf{t}_1$, which correspond to singular points at the segment end points, and to antiparallel tangent vectors of the same lengths, will be excluded.

3.1 Construction of the Interpolants

Two curves $\mathbf{p}(t)$, $\tilde{\mathbf{p}}(t)$ share the same hodograph if and only if they differ only by translation. Consequently a space PH curve $\mathbf{p}(t)$ is fully determined by the preimage $\mathcal{A}(t)$ and by the location of its starting point $\mathbf{p}(0)$.

The position of $\mathbf{p}_0$ can be matched by a suitable choice of the integration constant. The remaining $3 \cdot 3 = 9$ conditions must be satisfied by choosing the control points of the preimage $\mathcal{A}(t)$. Hence, the degree of $\mathcal{A}(t)$ has to be at least 2, yielding $3 \cdot 4 = 12$ free parameters. As shown in [10], the representation *preimage* → *hodograph* (12) has one dimensional fibers. Therefore one can expect that there will be a two dimensional system of PH interpolants of degree $2 \cdot 2 + 1 = 5$.

[1] This includes the generic case $\gcd(x'(t), y'(t), z'(t)) = 1$.

We will use the Bernstein-Bézier representation [16] of the hodograph $\mathbf{h}(t) = \mathbf{p}'(t)$ and the preimage $\mathcal{A}(t)$:

$$\mathbf{h}(t) = \sum_{i=0}^{4} \mathbf{h}_i B_i^4(t), \quad \mathcal{A}(t) = \sum_{i=0}^{2} \mathcal{A}_i B_i^2(t), \quad t \in [0,1], \tag{13}$$

where $\mathbf{h}_i$ (pure quaternions) and $\mathcal{A}_i$ (quaternions) are the control points and $B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j}$ are the Bernstein polynomials. The interpolation conditions lead to the equations

$$\mathbf{h}_0 = \mathbf{t}_0, \quad \mathbf{h}_4 = \mathbf{t}_1, \text{ and } \frac{1}{5}\sum_{i=0}^{4} \mathbf{h}_i = (\mathbf{p}_1 - \mathbf{p}_0), \tag{14}$$

which have to be satisfied by the control points of the hodograph. After expressing them in terms of the control points of the preimage curve, and a suitable re–arranging, one arrives at the following system of equations [9]:

$$\mathcal{A}_0 \mathbf{i} \mathcal{A}_0^* = \mathbf{t}_0, \quad \mathcal{A}_2 \mathbf{i} \mathcal{A}_2^* = \mathbf{t}_1, \tag{15}$$

and

$$\begin{aligned}(3\mathcal{A}_0 + 4\mathcal{A}_1 + 3\mathcal{A}_2)\mathbf{i}(3\mathcal{A}_0 + 4\mathcal{A}_1 + 3\mathcal{A}_2)^* &= \\ 120(\mathbf{p}_1 - \mathbf{p}_0) - 15(\mathbf{t}_1 + \mathbf{t}_0) &+ 5(\mathcal{A}_0 \mathbf{i} \mathcal{A}_2^* + \mathcal{A}_2 \mathbf{i} \mathcal{A}_0^*).\end{aligned} \tag{16}$$

These three equations have the form (8). From (15) we get two 1–parametric systems of solutions $\mathcal{A}_0(\phi_0)$ and $\mathcal{A}_2(\phi_2)$ of the form (9) (step 1). After substituting them into (16), we get (step 2) a 3-parametric system of solutions $\mathcal{A}_1(\phi_0, \phi_1, \phi_2)$.

Summing up, we arrive at a three-parametric system of suitable preimages

$$\mathcal{A}(t) = \mathcal{A}_0(\phi_0) B_0^2(t) + \mathcal{A}_1(\phi_0, \phi_1, \phi_2) B_1^2(t) + \mathcal{A}_2(\phi_2) B_2^2(t). \tag{17}$$

However, as observed in [9], the resulting PH curve depends only on the differences of the angular parameters ϕ_0, ϕ_1, ϕ_2, and therefore the preimages with a fixed value of ϕ_1 still give all possible PH interpolants. We fix $\phi_1 = 0$ and denote the system of preimages as $\mathcal{A}_{\phi_0,\phi_2}(t)$. [2]

The quintic PH interpolants are obtained from

$$\mathbf{p}_{\phi_0,\phi_2}(\tau) = \mathbf{p}_0 + \int_0^{\tau} \mathcal{A}_{\phi_0,\phi_2}(t) \mathbf{i} \mathcal{A}^*_{\phi_0,\phi_2}(t) \, \mathrm{d}t. \tag{18}$$

As a first example, Figure 1 shows some representatives of the system of all PH quintic interpolants to the data

$$\mathbf{p}_0 = (0,0,0)^\top, \quad \mathbf{p}_1 = (1,0,0)^\top, \quad \mathbf{t}_0 = (3,3,0)^\top, \quad \mathbf{t}_1 = (3,3,0)^\top. \tag{19}$$

Note, that while this data are in fact planar (since it is contained in the xy plane), most interpolants are truly spatial curves.

[2] Any fixed value of ϕ_1 gives equivalent results. In [9] authors choose $\phi_1 = -\pi/2$.

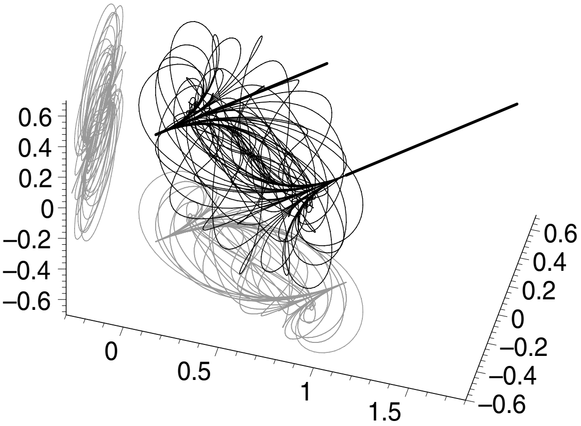


Fig. 1. The system of space PH quintic interpolants to given data. 64 representatives are plotted, along with their projections into the xy and yz planes (gray lines). The end-point tangent vectors are also shown, scaled by 1/4

3.2 Invariance of Interpolants

For any given Hermite data $\mathbf{p}_0, \mathbf{p}_1, \mathbf{t}_0, \mathbf{t}_1$, the system $\{\mathbf{p}_{\phi_0,\phi_2}(t) \,|\, \phi_0, \phi_2 \in [0, 2\pi)\}$ represents all PH Hermite interpolants. Therefore, it is invariant [10] with respect to orthogonal transformations (including reflections).

More precisely, if we apply an orthogonal transformation Ξ to the Hermite data, we get modified data $\tilde{\mathbf{p}}_0$, $\tilde{\mathbf{p}}_1$, $\tilde{\mathbf{t}}_0$, $\tilde{\mathbf{t}}_1$. The associated systems of interpolants then satisfy

$$\{\tilde{\mathbf{p}}_{\tilde{\phi}_0,\tilde{\phi}_2}(t) \,|\, \tilde{\phi}_0, \tilde{\phi}_2 \in [0, 2\pi)\} = \Xi(\{\mathbf{p}_{\phi_0,\phi_2}(t) \,|\, \phi_0, \phi_2 \in [0, 2\pi)\}).$$

On the other hand, this transformation does not preserve the parameterization of the solutions: In general

$$\tilde{\mathbf{p}}_{\tilde{\phi}_0,\tilde{\phi}_2}(t) = \Xi(\mathbf{p}_{\phi_0,\phi_2}(t)) \tag{20}$$

is not valid for $\tilde{\phi}_0 = \phi_0$, $\tilde{\phi}_2 = \phi_2$.

The relation between ϕ_0, ϕ_2 and $\tilde{\phi}_0, \tilde{\phi}_2$ ensuring (20) is rather complicated [9]. Still, it can be formulated easily in the following cases.

Lemma 3. *For any* ϕ_0, ϕ_2*:*

1. *If* Ξ *is a rotation about the* $\mathbf{i}$*-axis, then*

$$\tilde{\mathbf{p}}_{\phi_0,\phi_2}(t) = \Xi(\mathbf{p}_{\phi_0,\phi_2}(t)).$$

2. *If* Ξ *is a reflection with respect to a plane containing the* $\mathbf{i}$*-axis, then*

$$\tilde{\mathbf{p}}_{\phi_0,\phi_2}(t) = \Xi(\mathbf{p}_{-\phi_0,-\phi_2}(t)).$$

Proof. Consider fixed values of ϕ_0, ϕ_2 and let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ denote the control points of the preimage for some data $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{t}_0$, $\mathbf{t}_1$ and $\tilde{\mathcal{A}}_0, \tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2$ for the transformed data $\tilde{\mathbf{p}}_0 = \Xi(\mathbf{p}_0)$, $\tilde{\mathbf{p}}_1 = \Xi(\mathbf{p}_1)$, $\tilde{\mathbf{t}}_0 = \Xi(\mathbf{t}_0)$, $\tilde{\mathbf{t}}_1 = \Xi(\mathbf{t}_1)$.

1) Any rotation can be expressed using formula (6). If Ξ is rotation about the $\mathbf{i}$-axis through angle θ, then for any vector $\mathbf{c}$ we have

$$\Xi(\mathbf{c}) = \mathcal{Q}(\frac{\theta}{2})\mathbf{c}\mathcal{Q}(-\frac{\theta}{2}). \tag{21}$$

The right-hand side of the previous equation allows to extend the transformation Ξ from pure quaternions to all quaternions. Because of the form of the equations (9) and the fact that $\mathcal{Q}(\phi)\mathcal{Q}(\frac{\theta}{2}) = \mathcal{Q}(\frac{\theta}{2})\mathcal{Q}(\phi)$,

$$\tilde{\mathcal{A}}_0 = \Xi(\mathcal{A}_0), \text{ and } \tilde{\mathcal{A}}_2 = \Xi(\mathcal{A}_2). \tag{22}$$

Now using (21) and (22)

$$(\tilde{\mathcal{A}}_0\mathbf{i}\tilde{\mathcal{A}}_2^* + \tilde{\mathcal{A}}_2\mathbf{i}\tilde{\mathcal{A}}_0^*) = \Xi(\mathcal{A}_0\mathbf{i}\mathcal{A}_2^* + \mathcal{A}_2\mathbf{i}\mathcal{A}_0^*). \tag{23}$$

Consequently, in step 2, the right-hand side of equation (16) for the transformed data is equal to the transformed right-hand side of this equation for the original data. Hence,

$$\tilde{\mathcal{A}}_1 = \Xi(\mathcal{A}_1) \text{ and thus for the whole preimage curve } \tilde{\mathcal{A}}(t) = \Xi(\mathcal{A}(t)). \tag{24}$$

Finally

$$\tilde{\mathbf{p}}(\tau) = \tilde{\mathbf{p}}_0 + \int_0^\tau \tilde{\mathcal{A}}(t)\mathbf{i}\tilde{\mathcal{A}}^*(t)\,\mathrm{d}t = \Xi(\mathbf{p}_0) + \int_0^\tau \Xi(\mathcal{A}(t)\mathbf{i}\mathcal{A}^*(t))\,\mathrm{d}t = \Xi(\mathbf{p}(\tau)). \tag{25}$$

2) Due to the first part of the lemma, it suffices to consider only the reflection $\Xi_\mathbf{k}$ with respect to the $\mathbf{i}, \mathbf{j}$ plane. Indeed, any other reflection with respect to a plane containing the $\mathbf{i}$ axis can be obtained as a composition of $\Xi_\mathbf{k}$ and two rotations about the $\mathbf{i}$ axis.

$\Xi_\mathbf{k}$ can be extended to all quaternions setting

$$\Xi_\mathbf{k}(a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}) = -a + b\mathbf{i} + c\mathbf{j} - d\mathbf{k}. \tag{26}$$

A direct computation confirms, that formulas (22)-(25) are still valid, if the control points $\tilde{\mathcal{A}}_i$ are constructed with parameters $-\phi_0, -\phi_2$, while the control points $\mathcal{A}_i$ are constructed with the parameters ϕ_0, ϕ_2. □

A fully invariant parameterization of interpolants is obtained by considering a standard position.

Definition 1. *The C^1 spatial Hermite data are said to be in a standard position, if $\mathbf{t}_0 + \mathbf{t}_1$ is a positive multiple of $\mathbf{i}$, and $\mathbf{p}_0 = 0$.*

From now on, we will use the following parameterization of the system of interpolants.

Definition 2. *The system of interpolants is parameterized by the two parameters ϕ_0, ϕ_2, as follows. First we transform data to a standard position, where we construct the interpolants $\mathbf{p}_{\phi_0,\phi_2}(t)$, as described before. Finally, we transform the solution back to the original position.*

Note that parameterization is well–defined, since Lemma 3 ensures that the particular choice of a standard position (which may vary by a rotation about $\mathbf{i}$ axis) does not matter.

Theorem 1. *The parameterization of the solutions, according to Definition 2, is invariant with respect to rigid body motions (special orthogonal transformations), whereas reflections change the signs of both parameters. Consequently, the solution $\mathbf{p}_{0,0}(t)$ is invariant with respect to all orthogonal transformations.*

The proof results from Lemma 3.

In addition, the parameterization of the solutions is symmetric in the following sense.

Theorem 2. *Let $\mathbf{p}_{\phi_1,\phi_2}(t)$ be the interpolants of data $\mathbf{p}_0, \mathbf{p}_1, \mathbf{t}_0, \mathbf{t}_1$ and $\bar{\mathbf{p}}_{\phi_0,\phi_2}(t)$ the interpolants of the "reversed" data $\bar{\mathbf{p}}_0 = \mathbf{p}_1, \bar{\mathbf{p}}_1 = \mathbf{p}_0, \bar{\mathbf{t}}_0 = -\mathbf{t}_1, \bar{\mathbf{t}}_1 = -\mathbf{t}_0$. Then for any ϕ_0, ϕ_2*

$$\bar{\mathbf{p}}_{\phi_0,\phi_2}(1-t) = \mathbf{p}_{-\phi_2,-\phi_0}(t). \tag{27}$$

Proof. Suppose that the given data to be in a standard position, i.e. $\mathbf{t}_0 + \mathbf{t}_1$ is a positive multiple of $\mathbf{i}$ and $\mathbf{p}_0 = 0$. The reversed data can be transformed into a new standard position by the translation of the vector $-\mathbf{p}_1$ and the rotation given as composition of the the symmetry $S : \mathbf{c} \to -\mathbf{c}$ and reflection $\Xi_\mathbf{k}$ (26). In fact, S already transforms the reversed data into a standard position, but it is not a rotation ($\det(S) = -1$).

The standard position associated with the reversed data differs from the original data only by swapping $\mathbf{t}_0$ and $\mathbf{t}_1$ and by the reflection $\Xi_\mathbf{k}$. The theorem then follows from the symmetry of the system of equations (15)-(16) with respect to $\mathcal{A}_0$ and $\mathcal{A}_2$, and from the second part of Lemma 3. □

4 Qualitative Analysis of the Interpolants

In this section we give a qualitative analysis of the system of PH quintic interpolants, in order to identify the 'best' values of the parameters ϕ_0, ϕ_2. These parameters yield the interpolants suitable for applications.

4.1 Asymptotic Behavior

In order to fix the free parameters ϕ_0, ϕ_2, we will now study the asymptotic behavior of the solutions $\mathbf{p}_{\phi_0,\phi_2}(t)$. More precisely, we assume that the C^1 Hermite data are taken from a small segment of an analytical curve, and we investigate the asymptotic behavior of the solutions for decreasing step-size.

We assume that the curve is given by its Taylor expansion. Without loss of generality,

$$\mathbf{C}(T) = (T + \sum_{i=2}^{\infty} \frac{x_i}{i!} T^i,\ \sum_{i=2}^{\infty} \frac{y_i}{i!} T^i,\ \sum_{i=2}^{\infty} \frac{z_i}{i!} T^i)^\top \tag{28}$$

with arbitrary coefficients $x_2, x_3, \ldots$, $y_2, y_3, \ldots$ and $z_2, z_3, \ldots$.

For any step–size h, we pick the segment $\mathbf{c}(t) = \mathbf{C}(ht)$, $t \in [0,1]$. This segment has the expansion

$$\mathbf{c}(t) = (th + \sum_{i=2}^{\infty} \frac{x_i}{i!} t^i h^i,\ \sum_{i=2}^{\infty} \frac{y_i}{i!} t^i h^i, \sum_{i=2}^{\infty} \frac{z_i}{i!} t^i h^i)^\top. \tag{29}$$

Now we interpolate the C^1 Hermite boundary data at the points $\mathbf{c}(0) = \mathbf{C}(0)$ and $\mathbf{c}(1) = \mathbf{C}(h)$. Depending on the interval size h, different PH curves interpolating the data behave as described in the following Theorem.

Theorem 3. *The error of the PH interpolation*

$$\max_{t \in [0,1]} ||\mathbf{c}(t) - \mathbf{p}_{\phi_0,\phi_2}(t)|| \tag{30}$$

converges to 0 *as* $\mathcal{O}(h^4)$ *if and only if* $\phi_0 = \phi_2 = 0$. *Otherwise it converges to* 0 *only as* $\mathcal{O}(h^1)$.

Proof. The proof consists in evaluating power series of all quantities occurring in the interpolation process with respect to the step size h. This can be done by a suitable computer algebra tool. Due to the space limitation and the complexity of the expressions, we show only the leading terms of certain quantities, in order to illustrate the idea of our approach.

First, we derive the Taylor expansions of the Hermite boundary data at $t = 0$ and $t = 1$ of the curve (29),

$$\begin{aligned} \mathbf{p}_0 &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \ \mathbf{p}_1 = \begin{pmatrix} h + \frac{1}{2} x_2 h^2 + \frac{1}{6} x_3 h^3 + \ldots \\ \frac{1}{2} y_2 h^2 + \frac{1}{6} y_3 h^3 + \ldots \\ \frac{1}{2} z_2 h^2 + \frac{1}{6} z_3 h^3 + \ldots \end{pmatrix} \\ \mathbf{t}_0 &= \begin{pmatrix} h \\ 0 \\ 0 \end{pmatrix} \ \mathbf{t}_1 = \begin{pmatrix} h + x_2 h^2 + \frac{1}{2} x_3 h^3 + \ldots \\ y_2 h^2 + \frac{1}{2} y_3 h^3 + \ldots \\ z_2 h^2 + \frac{1}{2} z_3 h^3 + \ldots \end{pmatrix}. \end{aligned} \tag{31}$$

This data can be transformed into a standard position by a rotation

$$U = \begin{pmatrix} 1 - \frac{{y_2}^2 + {z_2}^2}{8} h^2 + \ldots & \frac{y_2}{2} h + \frac{y_3 - y_2 x_2}{4} h^2 + \ldots & \frac{z_2}{2} h + \frac{z_3 - z_2 x_2}{4} h^2 + \ldots \\ -\frac{y_2}{2} h - \frac{y_3 - y_2 x_2}{4} h^2 + \ldots & 1 - \frac{{y_2}^2}{8} h^2 + \ldots & 0 \\ -\frac{z_2}{2} h - \frac{z_3 - z_2 x_2}{4} h^2 + \ldots & -\frac{z_2 y_2}{4} h^2 + \ldots & 1 - \frac{{z_2}^2}{8} h^2 + \ldots \end{pmatrix}.$$

Then we compute the Taylor expansions of the control points of the preimage for the transformed data $U(\mathbf{p}_0)$, $U(\mathbf{p}_1)$, $U(\mathbf{t}_0)$, $U(\mathbf{t}_1)$. Using (9) we obtain

$$\begin{aligned}\mathcal{A}_0(\phi_0) &= -\sqrt{h}[\sin\phi_0 + \dots] + \sqrt{h}[\cos\phi_0 + \dots]\mathbf{i} - \sqrt{h}[\tfrac{y_2\cos\phi_0 + z_2\sin\phi_0}{4}h + \dots]\mathbf{j} \\ &\quad -\sqrt{h}[\tfrac{z_2\cos\phi_0 - y_2\sin\phi_0}{4}h + \dots]\mathbf{k}, \\ \mathcal{A}_2(\phi_2) &= -\sqrt{h}[\sin\phi_2 + \tfrac{x_2\sin\phi_2}{2}h + \dots] + \sqrt{h}[\cos\phi_2 + \tfrac{x_2\cos\phi_2}{2}h + \dots]\mathbf{i} \\ &\quad +\sqrt{h}[\tfrac{y_2\cos\phi_2 + z_2\sin\phi_2}{4}h + \dots]\mathbf{j} + \sqrt{h}[\tfrac{z_2\cos\phi_2 - y_2\sin\phi_2}{4}h + \dots]\mathbf{k}\end{aligned} \tag{32}$$

(step 1). In step 2 (with the fixed choice $\phi_1 = 0$) we obtain the expansion of $\mathcal{A}_1$, involving both ϕ_0, ϕ_2. We omit it here since even the leading terms are rather complicated.

Finally we are able to express the Taylor expansion of the PH interpolant $\mathbf{p}_{\phi_1,\phi_2}(t)$, which is again a long expression. Still, the leading term of its x component is equal to

$$\begin{aligned}\Big[&t + \tfrac{1}{2}\left(\cos(\phi_0)\sqrt{10\cos(\phi_2-\phi_0)+90} - 3\cos(\phi_2-\phi_0) - 7\right)t^2 \\ &-\tfrac{1}{2}\left([3\cos(\phi_0) + \cos(\phi_2)]\sqrt{10\cos(\phi_2-\phi_0)+90} - 12\cos(\phi_2-\phi_0) - 28\right)t^3 \\ &+\tfrac{1}{2}\left([3\cos(\phi_0) + 2\cos(\phi_2)]\sqrt{10\cos(\phi_2-\phi_0)+90} - 15\cos(\phi_2-\phi_0) - 35\right)t^4 \\ &-\tfrac{1}{2}\left([\cos(\phi_0) + \cos(\phi_2)]\sqrt{10\cos(\phi_2-\phi_0)+90} - 6\cos(\phi_2-\phi_0) - 14\right)t^5\Big]\,h.\end{aligned} \tag{33}$$

Comparing this series with (29), we see that the coefficients at t^2, t^3, t^4, t^5 in (33) must be zero if the interpolant $\mathbf{p}_{\phi_1,\phi_2}(t)$ should match the the shape of $\mathbf{c}(t)$ and the error (30) should converge to 0 faster then $\mathcal{O}(h)$. Solving the system of trigonometric equations it can be shown, that this is achieved if and only if $\phi_0 = \phi_2 = 0$. Using these values, the Taylor expansion of $\mathbf{p}_{0,0}(t)$ simplifies enormously, and matches the Taylor expansion of $\mathbf{c}(t)$ up to h^3. □

Remark 1. Though the proof Theorem 3 seems to be complicated, it is in fact a straightforward computation. It would be interesting to proof a more general result stating that any C^1 Hermite interpolation satisfying certain conditions always leads to approximation order 4, as it is the case for Hermite interpolation by Bézier cubics and PH quintics $\mathbf{p}_{0,0}(t)$. This may be a subject of future research.

Remark 2. In Theorem 3 we considered only constant values of ϕ_0 and ϕ_2, which do not depend on the the step size h. It may also be interesting to choose these parameters depending on the step-size. Thus $\phi_0(h), \phi_2(h)$ would be functions of h. In this more general setting we still can study the asymptotic behavior. It is only necessary to replace the constants ϕ_0, ϕ_2 by abstract Taylor series of the functions $\phi_0(h), \phi_2(h)$. We obtained the following result about the approximation order.

The error of the PH interpolation

$$\max_{t\in[0,1]} ||\mathbf{c}(t) - \mathbf{p}_{\phi_0(h),\phi_2(h)}(t)||$$

is equal to $\mathcal{O}(h^4)$ if and only if

1. $\lim_{h\to 0}\phi_1(h) = \lim_{h\to 0}\phi_2(h) = 0$ and
2. $\phi_2'(0) = -\phi_0'(0)$.

If only the first condition holds, the error of the approximation equals $\mathcal{O}(h^3)$. If 1. is not satisfied, then the error equals $\mathcal{O}(h^1)$.

Remark 3. The interpolant $\mathbf{p}_{0,0}(t)$ has the following interesting property. If $\phi_0 = \phi_1 = \phi_2 = 0$, then the control points $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_2$ are pure quaternions, and therefore the whole preimage $\mathcal{A}(t)$ is pure-quaternion valued. This corresponds to setting $u(t) = 0$ in the representation formula (11), which then becomes

$$\begin{aligned} x'(t) &= v^2(t) - p^2(t) - q^2(t), \\ y'(t) &= 2v(t)p(t), \\ z'(t) &= 2v(t)q(t), \\ \sigma(t) &= v^2(t) + p^2(t) + q^2(t). \end{aligned} \tag{34}$$

This incomplete description of PH curves was used first in [5]. The optimal solution $\mathbf{p}_{0,0}(t)$ is therefore given by transforming the curve into standard position and constructing one of the interpolants of the form (34).

4.2 Preservation of Planarity

As a natural question, one may ask which interpolants $\mathbf{p}_{\phi_0,\phi_2}$ are planar for planar input data.

C^1 Hermite interpolation by PH quintics in the plane is well understood. It has been shown, that for any non-degenerated planar data there are four planar PH quintic interpolants [6], which have been in [20] labeled as $(++)$, $(+-)$, $(-+)$ and $(--)$. In the following new result we identify them among the two parameter family of spatial interpolants.

Theorem 4. *For any input data $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{t}_0$, $\mathbf{t}_1$ lying in a plane, the four interpolants $\mathbf{p}_{0,0}(t)$, $\mathbf{p}_{0,\pi}(t)$, $\mathbf{p}_{\pi,0}(t)$, $\mathbf{p}_{\pi,\pi}(t)$ are planar. They can be identified in the following way with the interpolants in [20]:*

$$\mathbf{p}_{0,0}(t) = (++),\ \mathbf{p}_{0,\pi}(t) = (+-),\ \mathbf{p}_{\pi,0}(t) = (-+),\ \textit{and}\ \mathbf{p}_{\pi,\pi}(t) = (--). \tag{35}$$

Proof. We suppose that the input data lie in the $\mathbf{i},\mathbf{j}$ plane, which we will denote with $\mathbb{Q}_{\mathbf{ij}}$. We define on the quaternions the commutative multiplication

$$\mathcal{U} \star \mathcal{V} := \frac{1}{2}(\mathcal{U}\,\mathbf{i}\,\mathcal{V}^* + \mathcal{V}\,\mathbf{i}\,\mathcal{U}^*). \tag{36}$$

One can verify directly, that $(\mathbb{Q}_{\mathbf{ij}}, \star) \simeq \mathbb{C}$. Under this isomorphism, the equations (15)-(16) become the complex equations characterizing PH interpolation in plane (equations (3)-(4) of [20]).

Obviously, for the choice $\phi_0 = 0$, resp. $\phi_0 = \pi$, the control point $\mathcal{A}_0$ is in $\mathbb{Q}_{\mathbf{ij}}$ and correspond to the complex square root of $\mathbf{t}_0$ with positive, resp. negative real part, which is precisely the principle of the labeling used in [20]. Similarly for $\mathcal{A}_2$ and ϕ_2. Then also $\mathcal{A}_1 \in \mathbb{Q}_{\mathbf{ij}}$ and the correspondence (35) holds. □

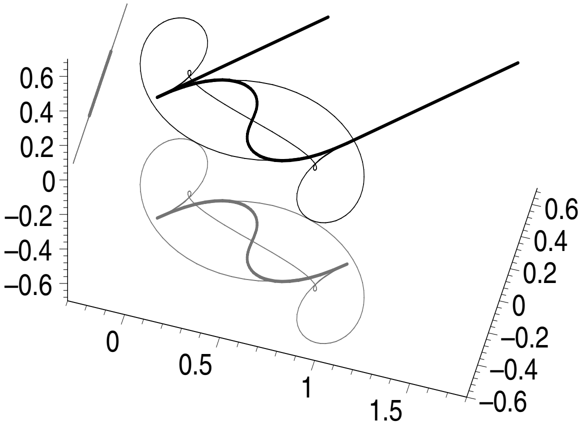


Fig. 2. Planar interpolants of Hermite data (19). The interpolant $\mathbf{p}_{0,0}$ is plotted in bold. Projections to xy and yz plane are plotted in grey

Figure 2 shows the four planar interpolants $\mathbf{p}_{0,0}(t)$, $\mathbf{p}_{0,\pi}(t)$, $\mathbf{p}_{\pi,0}(t)$, $\mathbf{p}_{\pi,\pi}(t)$ to the planar data (19) (see also the previous figure). Note that the projections (grey) into the yz plane collapse into line segments.

5 Applications

We apply the previous results in order to design an algorithm converting any analytical curve into a piecewise PH quintic curve. This conversion is then used for approximation of pipe surfaces.

5.1 Conversion of Analytical Curves

The result described in Theorem 3 allows us to design an algorithm for the conversion of any analytical curve into a piecewise PH curve. Let the parameter domain of the analytical curve be $[0, 1]$. We split this interval into the 2^n subintervals $[\frac{i}{2n}, \frac{i+1}{2n}]$, $i = 0..2n - 1$. For each subinterval, we construct the PH Hermite interpolant $\mathbf{p}_{0,0}(t)$ and obtain a C^1 continuous piecewise PH curve of degree 5. If the error from the original analytical curve is not sufficiently small, we continue the subdivision. Due to the Proposition 3, the error will converge to 0 as $\mathcal{O}\left(\frac{1}{16^n}\right)$ under subdivision.

The relatively high rate of convergence is demonstrated by the following example.

Figure 3 shows the segment of the analytical curve

$$\mathbf{c}(t) = (1.5\sin(7.2t), \cos(9t), e^{\cos(1.8t)})^\top, t \in [0, 1]. \tag{37}$$

We construct the PH Hermite interpolant for the whole segment and the piecewise PH interpolants obtained after splitting the parameter into 2, 4, 8, ..., 512

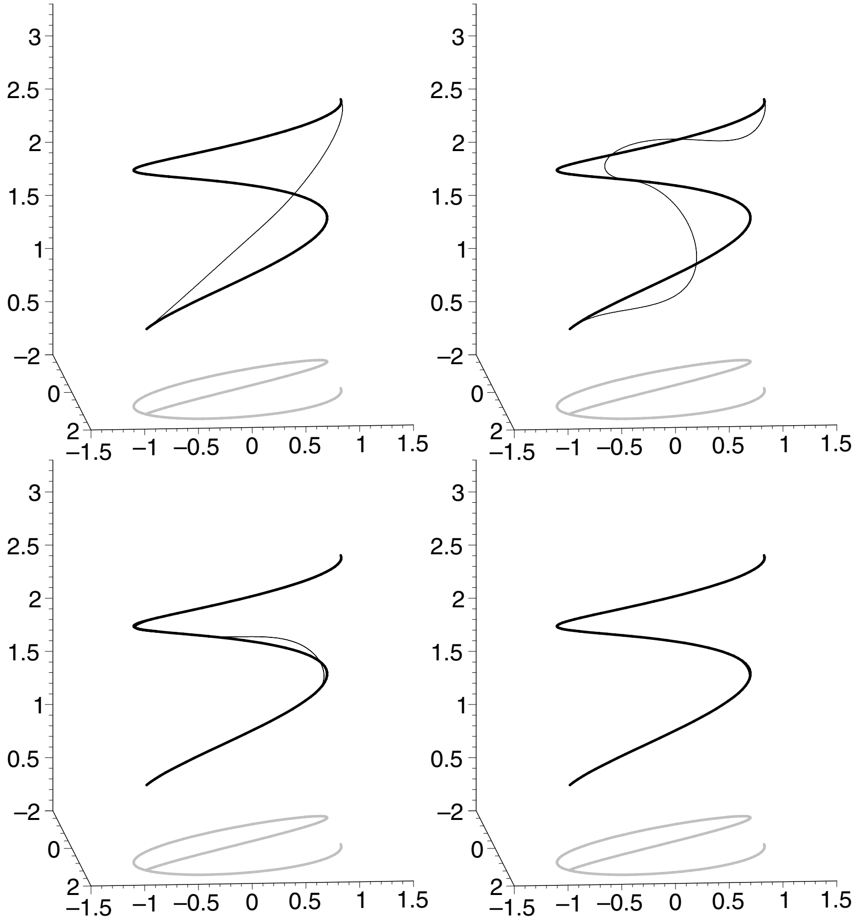


Fig. 3. Approximate conversion of an analytical curve (bold line) via C^1 Hermite interpolation by PH curves, obtained after splitting the parameter domain into 1, 2, 4 and 8 segments. The difference between the curves is almost invisible in the last case. In addition to the curve its projection into xy plane is plotted (gray line)

subintervals. The maximal approximation error and its improvement (ratio) in each step are shown in Table 1.

Clearly, instead of the simple uniform subdivision, using an adaptive subdivision scheme would reduce the number of segments.

5.2 Approximation of Pipe Surfaces

PH curves possess a simple low degree rational adapted frame, which has been called the Euler-Rodrigues frame in [3]. Based on this construction, Farouki

Table 1. Error of piecewise quintic PH approximation via Hermite interpolation

#Segments	Error	Ratio	#Segments	Error	Ratio
1	2.429		32	$1.941\,10^{-4}$	10.67×
2	1.384	1.76×	64	$1.337\,10^{-5}$	14.52×
4	$1.553\,10^{-1}$	8.91×	128	$8.523\,10^{-7}$	15.68×
8	$2.399\,10^{-2}$	6.48×	256	$5.376\,10^{-8}$	15.85×
16	$2.070\,10^{-3}$	11.59×	512	$3.361\,10^{-9}$	16.00×

proposed a rational approximation of the rotation minimizing frame for any space PH curve [13].

Both frames can be used for the approximation of pipe surfaces, or – more generally – of sweep surfaces. First we convert a given analytical curve into a piecewise PH curve using the algorithm of section 5.1 and then we construct a pipe surface for this PH curve.

The approximation error can be defined in the following way. The exact pipe surface with radius r can be understood as a union of circles $S(t)$, with centers $\mathbf{c}(t)$ and lying in the normal plane of $\mathbf{c}$ at point t. Similarly, our approximation of the pipe surface can be seen as collection of the circles $\tilde{S}(t)$.

For each parameter value t, we define $E(t)$ as the Hausdorff distance of the circles $S(t)$ and $\tilde{S}(t)$. Then the global error of the approximation is defined as

$$E = \max_{t\in[0,1]} E(t). \tag{38}$$

Theorem 5. *The approximation error E behaves as $\mathcal{O}(h^3)$ for $h \to 0$.*

Proof. Using the triangle inequality, the Hausdorff distance between circles $S(t)$ and $\tilde{S}(t)$ can be bounded as

$$E(t) \leq r \left\| \frac{\mathbf{c}'(t)}{||\mathbf{c}'(t)||} - \frac{\mathbf{p}'_{0,0}(t)}{||\mathbf{p}'_{0,0}(t)||} \right\| + \|\mathbf{c}(t) - \mathbf{p}_{0,0}(t)\|, \tag{39}$$

where the first term represents the Hausdorff distance between two circles with a common center and the second one is the distance between the centers. According to Theorem 3

$$\max_{t\in[0,1]} ||\mathbf{c}(t) - \mathbf{p}_{0,0}(t)|| = \mathcal{O}(h^4). \tag{40}$$

In a similar way one can prove that

$$\max_{t\in[0,1]} \left\| \frac{\mathbf{c}'(t)}{||\mathbf{c}'(t)||} - \frac{\mathbf{p}'_{0,0}(t)}{||\mathbf{p}'_{0,0}(t)||} \right\| = \mathcal{O}(h^3), \tag{41}$$

which concludes the proof. □

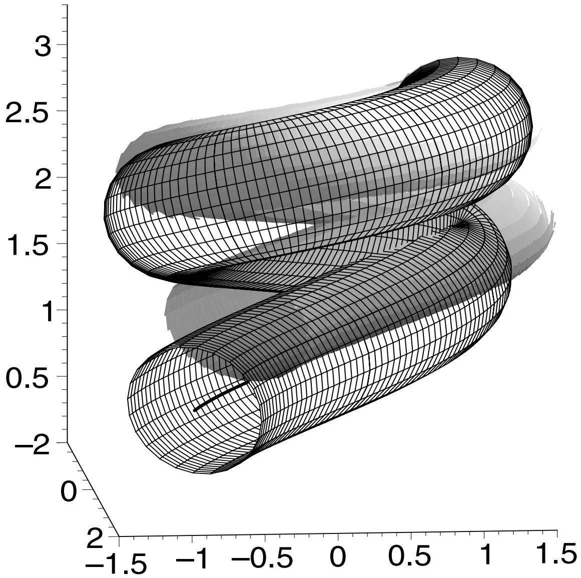


Fig. 4. Approximation of a pipe surface

As an example, Figure 4 shows an approximation of a pipe surface associated with the curve (37), constructed using a piecewise PH curve composed of 8 segments (see last figure of Fig. 3).

6 Conclusion

Starting from previous results about quintic PH curves in two and three dimensions [9, 20], we analyzed the system of solutions and identified one of them, which is suitable for applications. More precisely, it is invariant under orthogonal transformations (rigid body transformations and reflections), preserves planarity, it is invariant with respect to reversion of the parameter interval, and it has the optimum approximation order. We used this solution for approximately converting general curves into PH form and for approximation of pipe surfaces.

It should be noted that choosing this solution leads to a significant improvement of the approximation order. In order to achieve the same error with other solutions, 8 times as many intervals would be needed. Also, the shape might be less pleasing.

As a matter of future work, we will investigate the problem of C^2 Hermite interpolation in three–dimensional space. According to our experience in the planar case, the use of geometric Hermite data (points, tangent directions, curvatures) always produces problems with specific points, such as inflections, while these difficulties are not present for analytical data (points, first and second derivatives). Consequently, the use of analytical data seems to be more appropriate.

Acknowledgment. The first author was supported through grant P17387-N12 of the Austrian Science Fund (FWF).

References

1. M.-H. Ahn, G.-I. Kim and C.-N. Lee, Geometry of root-related parameters of PH curves. Appl. Math. Lett. **16** (2003), no. 1, 49–57.
2. H.I. Choi, D.S. Lee and H.P. Moon (2002), Clifford algebra, spin representation, and rational parameterization of curves and surfaces. Adv. Comput. Math. 17, 5–48.
3. H.I. Choi, C.Y. Han, Euler Rodrigues frames on spatial Pythagorean-hodograph curves, Comput. Aided Geom. Design 19 (2002) 603–620.
4. R. Dietz, J. Hoschek, B. Jüttler, An algebraic approach to curves and surfaces on the sphere and on other quadrics, Comp. Aided Geom. Design 10 (1993), 211–229.
5. R.T. Farouki and T. Sakkalis, Pythagorean-hodograph space curves, Adv. Comput. Math. 2 (1994) 41–66.
6. R.T. Farouki and C.A. Neff (1995), Hermite interpolation by Pythagorean-hodograph quintics. Math. Comput. 64, 1589–1609.
7. R. T. Farouki and K. Saitou and Y-F. Tsai (1998), Least-squares tool path approximation with Pythagorean-hodograph curves for high-speed CNC machining. The Mathematics of Surfaces VIII, Information Geometers, Winchester, 245–264.
8. R.T. Farouki, B.K. Kuspa, C. Manni, and A. Sestini (2001), Efficient solution of the complex quadratic tridiagonal system for C^2 PH quintic splines, Numer. Alg. 27, 35–60.
9. R. T. Farouki, M. al-Kandari and T. Sakkalis (2002), Hermite interpolation by rotation-invariant spatial Pythagorean-hodograph curves., Adv. Comput. Math. 17, 369–383.
10. R. T. Farouki, M. al-Kandari and T. Sakkalis (2002), Structural invariance of spatial Pythagorean hodographs, Comp. Aided Geom. Design 19, 395–407.
11. R.T. Farouki (2002), Pythagorean hodograph curves, in G. Farin, J. Hoschek and M.-S. Kim (eds.), Handbook of Computer Aided Geometric Design, North-Holland, Amsterdam, 405–427.
12. R.T. Farouki, C. Manni, A. Sestini (2003), Spatial C^2 PH quintic splines. Curve and surface design (Saint-Malo, 2002), 147–156, Mod. Methods Math., Nashboro Press.
13. R.T. Farouki, C. Y. Han (2003), Rational approximation schemes for rotation-minimizing frames on Pythagorean- hodograph curves, Computer Aided Geometric Design 20, 435–454.
14. R.T. Farouki, C.Y. Han, C. Manni, A. Sestini (2004), Characterization and construction of helical polynomial space curves, J. Comput. Appl. Math. 162, no. 2, 365–392.
15. R. Feichtinger (2005), Hermite–Interpolation mit PH–Quintiken, Diplomarbeit (MSc. thesis), Johannes Kepler University, Linz, Austria, in progress.
16. J. Hoschek and D. Lasser (1996), Fundamentals of Computer Aided Geometric Design, AK Peters, Wellesley MA.

17. B. Jüttler and C. Mäurer (1999), Cubic Pythagorean Hodograph Spline Curves and Applications to Sweep Surface Modeling, Comp.–Aided Design 31, 73–83.
18. B. Jüttler (2001), Hermite interpolation by Pythagorean hodograph curves of degree seven. Math. Comp. 70, 1089–1111.
19. J.B. Kuipers (1999), Quaternions and rotation sequences. Princeton University Press.
20. H.P. Moon, R.T. Farouki and H.I. Choi (2001), Construction and shape analysis of PH quintic Hermite interpolants, Comp. Aided Geom. Design 18, 93–115.
21. Z. Šír and B. Jüttler (2005), Constructing acceleration continuous tool paths using pythagorean hodograph curves. Mech. Mach. Theory. In press.
22. D.S. Meek and D.J. Walton (1997), Geometric Hermite interpolation with Tschirnhausen cubics, Journal of Computational and Applied Mathematics 81, 299–309.
23. D.J. Walton and D.S. Meek (2004), A generalisation of the Pythagorean hodograph quintic spiral. J. Comput. Appl. Math. 172, no. 2, 271–287.

Modelling Surface Normal Distribution Using the Azimuthal Equidistant Projection

William A.P. Smith and Edwin R. Hancock

Department of Computer Science, The University of York,
York, YO1 5DD, UK
{wsmith, erh}@cs.york.ac.uk

Abstract. This paper describes how surface shape, and in particular facial shape, can be modeled using a statistical model that captures variations in surface normal direction. To construct this model we make use of the azimuthal equidistant projection to map surface normals from the unit sphere to points on a local tangent plane. The variations in surface normal direction are captured using the covariance matrix for the projected point positions. This allows us to model variations in surface shape using a standard point distribution model. We show how this model can be trained using surface normal data acquired from range images. We fit the model to intensity data using constraints on the surface normal direction provided by Lambert's law. We demonstrate the utility of the method on the recovery of 3D surface shape from 2D images.

1 Introduction

Shape-from-shading provides an alluring yet somewhat elusive route to recovering 3D surface shape from single 2D intensity images [1]. Unfortunately, the method has proved ineffective in recovering the realistic shape of complex surfaces, such as the human face, because of local convexity-concavity instability caused by the bas-relief ambiguity [1]. This is of course a well known effect which is responsible for a number of illusions, including Gregory's famous inverted mask [2]. It is for this reason that methods such as photometric stereo [3] have proved to be more effective.

One way of overcoming this problem with single view shape-from-shading is to use domain specific constraints. For the specific case of faces, several authors [4, 5, 6, 7, 8] have shown that, at the expense of generality, the accuracy of recovered shape information can be greatly enhanced by restricting a shape-from-shading algorithm to a particular class of objects. For instance, both Prados and Faugeras [8] and Castelan and Hancock [7] use the location of singular points to enforce convexity on the recovered surface. Zhao and Chellappa [5], on the other hand, have introduced a geometric constraint which exploited the approximate bilateral symmetry of faces. This 'symmetric shape-from-shading' was used to correct for variation in illumination. They employed the technique for recognition by synthesis. However, the recovered surfaces were of insufficient quality to synthesise novel viewpoints. Moreover, the symmetry constraint

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 381–394, 2005.

is only applicable to frontal face images. Atick et al. [4] proposed a statistical shape-from-shading framework based on a low dimensional parameterisation of facial surfaces. Principal components analysis was used to derive a set of 'eigenheads' which compactly captures 3D facial shape. Unfortunately, it is surface orientation and not depth which is conveyed by image intensity. Therefore, fitting the model to an image equates to a computationally expensive parameter search which attempts to minimise the error between the rendered surface and the observed intensity. This is similar to the approach adopted by Samaras and Metaxas [6] who incorporate reflectance constraints derived from shape-from-shading into a deformable model.

Unfortunately, the construction of a statistical model for the distribution of surface normal directions is not a straightforward task. The reason for this is that the statistical representation of directional data has proved to be considerably more difficult than that for Cartesian data [9]. Surface normals can be viewed as residing on a unit sphere and may be specified in terms of the elevation and azimuth angles. This representation makes the computation of distance difficult. For instance, if we consider a short walk across one of the poles of the unit sphere, then although the distance traversed is small, the change in azimuth angle is large. Likewise, the significance of a change in azimuth angle is dependent on the magnitude of the elevation angle. In other words, elevation and azimuth are not comparable. Hence, constructing a statistical model that can capture the statistical distribution of directional data is not a straightforward task. Heap and Hogg [9] provided a partial solution to the angle discontinuity problem for the circular case by attempting to place the discontinuity such that no boundary crossings occur in a given distribution. Unfortunately, the method does not extend in a simple way to data on the sphere.

To overcome the problem, in this paper we draw on ideas from cartography. Our starting point is the *azimuthal equidistant* or Postel projection [10]. This projection has the important property that it preserves the distances between the centre of projection and all other locations on the sphere. It is used in cartography for path planning tasks. Another useful property of this projection is that straight lines on the projected plane through the centre of projection correspond to great circles on the sphere. The projection is constructed by selecting a reference point on the sphere and constructing the tangent plane to the reference point. Locations on the sphere are projected onto the tangent plane in a manner that preserves arc-length on the sphere.

We exploit this property to generate a local representation of the field of surface normals. Our idea is as follows. We commence with a set of needle-maps, i.e. fields of surface normals which in practice are obtained from range images. We begin by computing the mean field of surface normals. The surface normals are represented using elevation and azimuth angles on a unit sphere. At each image location the mean-surface normal defines a reference direction. We use this reference direction to construct an azimuthal equidistant projection for the distribution of surface normals at each image location. The distribution of points on the projection plane preserves the distances of the surfaces normals on the unit sphere with respect to the mean surface normal, or reference direction. We then construct a deformable model over the set of surface normals by applying the Cootes and Taylor [11] point distribution model to the co-ordinates that result from transforming the surface normals from the unit sphere to the tangent plane under azimuthal equidistant projection. On the tangent projection plane, the points associated with the surface normals are allowed to move in a manner which is determined by the principal component directions of the covariance matrix for the point-distribution. Once we have computed the allowed deformation movement on the tangent plane, we recover surface normal directions by using the inverse transformation onto the unit sphere.

We fit the model to 2D intensity images using ideas drawn from shape-from-shading. We couple the model to the raw image brightness using the geometric shape-from-shading framework of Worthington and Hancock [12]. According to this framework, when the surface reflectance follows Lambert's law, then the surface normal is constrained to fall on a cone whose axis is in the light source direction and whose opening angle is the inverse cosine of the normalised image brightness. This method commences from an initial configuration in which the surface normals reside on the irradiance cone and point in the direction of the local image gradient. The statistical model is fitted to recover a revised estimate of the surface normal directions. The best-fit surface normals are projected onto the nearest location on the irradiance cones. This process is iterated to convergence, and the height map for the surface recovered by integrating the final field of surface normals. We explore the utility of the resulting model for a number of face analysis tasks.

2 A Statistical Model for Surface Normals

Fields of surface normals provide an important source of information from which a statistical surface shape model can be constructed. The surface normals can be obtained in a number of ways, including capturing range images or using surface orientation information extracted from intensity data using processes such as shape-from-shading. A field of surface normals, or needle map, provides a more detailed description of an object than a corresponding brightness image. Surface normals are invariant to changes in illumination and surface reflectance. Moreover, topographic information such as surface curvature can be computed from a field of surface normals [13]. Using shape-from-shading [12], the field of surface normals is also more easily recovered from an image than the underlying surface height function, since it is orientation and not depth information which is conveyed by variations image intensity.

2.1 Azimuthal Equidistant Projection

A 'needle map' describes a surface $z(x, y)$ as a set of local surface normals $\mathbf{n}(x, y)$ projected onto the view plane. Let $\mathbf{n}_k(i, j) = (n_k^x(i, j), n_k^y(i, j), n_k^z(i, j))^T$ be the unit surface normal at the pixel indexed (i, j) in the k^{th} training image. If there are T images in the training set, then at the location (i,j) the mean-surface normal direction is

$$\hat{\mathbf{n}}(i,j) = \frac{\bar{\mathbf{n}}(i,j)}{||\bar{\mathbf{n}}(i,j)||} \tag{1}$$

where

$$\bar{\mathbf{n}}(i,j) = \frac{1}{T}\sum_{k=1}^{T}\mathbf{n}_k(i,j) \tag{2}$$

On the unit sphere, the surface normal $\mathbf{n}_k(i,j)$ has elevation angle $\theta_k(i,j) = \frac{\pi}{2} - \arcsin n_k^z(i,j)$ and azimuth angle $\phi_k(i,j) = \arctan\frac{n_k^y(i,j)}{n_k^x(i,j)}$, while the mean surface normal at the location (i,j) has elevation angles $\hat{\theta}(i,j) = \frac{\pi}{2} - \arcsin \hat{n}^z(i,j)$ and azimuth angle $\hat{\phi}(i,j) = \arctan\frac{\hat{n}^y(i,j)}{\hat{n}^x(i,j)}$.

To construct the azimuthal equidistant projection we proceed as follows. We commence by constructing the tangent plane to the unit-sphere at the location corresponding to the mean-surface normal. We establish a local co-ordinate system on this tangent plane. The origin is at the point of contact between the tangent plane and the unit sphere. The x-axis is aligned parallel to the local circle of latitude on the unit-sphere.

Under the azimuthal equidistant projection at the location (i,j), the surface normal $\mathbf{n}_k(i,j)$ maps to the point with coordinates $\mathbf{v}_k(i,j) = (x_k(i,j), y_k(i,j))^T$. The transformation equations between the unit-sphere and the tangent-plane coordinate systems are

$$x_k(i,j) = k'\cos\theta_k(i,j)\sin[\phi_k(i,j) - \hat{\phi}(i,j)] \tag{3}$$

$$y_k(i,j) = k'\left\{\cos\hat{\theta}(i,j)\sin\phi_k(i,j) - \sin\hat{\theta}(i,j)\cos\theta_k(i,j)\cos[\phi_k(i,j) - \hat{\phi}(i,j)]\right\} \tag{4}$$

where $\cos c = \sin\hat{\theta}(i,j)\sin\theta_k(i,j) + \cos\hat{\theta}(i,j)\cos\theta_k(i,j)\cos[\phi_k(i,j) - \hat{\phi}(i,j)]$ and $k' = c/\sin c$.

Thus, in Figure 1, CP' is made equal to the arc CP for all values of θ. The projected position of P, namely P', therefore lies at a distance θ from the

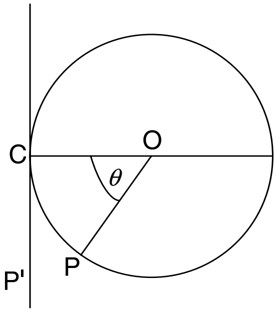


Fig. 1. The azimuthal equidistant projection

centre of projection and the direction of P' from the centre of the projection is true. The equations for the inverse transformation from the tangent plane to the unit-sphere are

$$\theta_k(i,j) = \sin^{-1}\left\{\cos c \sin\hat{\theta}(i,j) - \frac{1}{c} y_k(i,j) \sin c \cos\hat{\theta}(i,j)\right\} \tag{5}$$

$$\phi_k(i,j) = \hat{\phi}(i,j) + \tan^{-1}\psi(i,j) \tag{6}$$

where

$$\psi(i,j) = \tag{7}$$
$$\begin{cases} x_k(i,j)\sin c/(c\cos\hat{\theta}(i,j)\cos c - y_k(i,j)\sin\hat{\theta}(i,j)\sin c) & \text{if } \hat{\theta}(i,j) \neq \pm\frac{\pi}{2} \\ -x_k(i,j)/y_k(i,j) & \text{if } \hat{\theta}(i,j) = \frac{\pi}{2} \\ x_k(i,j)/y_k(i,j) & \text{if } \hat{\theta}(i,j) = -\frac{\pi}{2} \end{cases}$$

and $c = \sqrt{x^2 + y^2}$.

2.2 Point Distribution Model

For each image location the transformed surface normals from the T different training images are concatenated and stacked to form two long-vectors of length T. For the pixel location indexed (i,j), the first of these is the long vector with the transformed x-co-ordinates from the T training images as components, i.e.

$$\mathbf{V}_x(i,j) = (x_1(i,j), x_2(i,j), ..., x_T(i,j))^T \tag{8}$$

and the second long-vector has the y co-ordinate as its components, i.e.

$$\mathbf{V}_y(i,j) = (y_1(i,j), y_2(i,j), ..., y_T(i,j))^T \tag{9}$$

As the equidistant azimuthal projection involves centering the local co-ordinate system, the mean long-vectors over the training images are zero. If the data is of dimensions M rows and N columns, then there are $M \times N$ pairs of such long-vectors. The long vectors are ordered according to the raster scan (left-to-right and top-to-bottom) and are used as the columns of the $(2MN) \times T$ data-matrix

$$\mathbf{D} = (\mathbf{V}_x(1,1)|\mathbf{V}_y(1,1)|\mathbf{V}_x(1,2)|\mathbf{V}_y(1,2)|\ldots|\mathbf{V}_x(M,N)|\mathbf{V}_y(M,N))^T \tag{10}$$

The covariance matrix for the long-vectors is the $(2MN) \times (2MN)$ matrix $\mathbf{L} = \frac{1}{T}\mathbf{D}\mathbf{D}^T$. We use the numerically efficient *snap-shot* method of Sirovich [14] to compute the eigenvectors of $\mathbf{L}$. Accordingly, we construct the matrix $\hat{\mathbf{L}} = \frac{1}{K}\mathbf{D}^T\mathbf{D}$. The eigenvectors $\hat{\mathbf{e}}_i$ of $\hat{\mathbf{L}}$ can be used to find the eigenvectors $\mathbf{e}_i$ of $\mathbf{L}$ using $\mathbf{e}_i = \mathbf{D}\hat{\mathbf{e}}_i$. We deform the equidistant azimuthal point projections in the directions defined by the $2MN \times K$ matrix $\mathbf{P} = (\mathbf{e}_1|\mathbf{e}_2|\ldots|\mathbf{e}_K)$ formed from the leading K principal eigenvectors. This deformation displaces the transformed surface normals on the local tangent planes in the directions

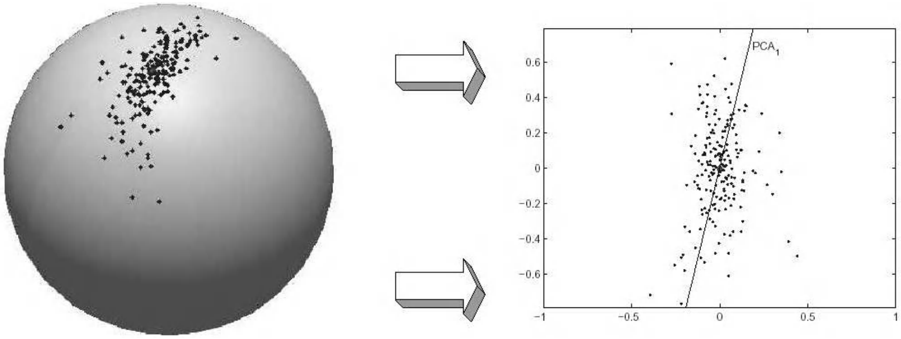


Fig. 2. Projection of points on the unit sphere to points on the tangent plane at the mean point

defined by the eigenvectors $\mathbf{P}$. If $\mathbf{b} = (b_1, b_2, ..., b_K)^T$ is a vector of parameters of length K, then since the mean-vector of co-ordinates resulting from the equidistant azimuthal projection is zero, the deformed vector of projected co-ordinates is $\mathbf{v}_b = \mathbf{Pb}$. Suppose that $\mathbf{v}_o$ is the vector of co-ordinates obtained by performing the azimuthal equidistant projection on an observed field of surface normals. We seek the parameter vector $\mathbf{b}$ that minimises the squared error $\mathcal{E}(\mathbf{b}) = (\mathbf{v}_o - \mathbf{P}^T\mathbf{b})^T(\mathbf{v}_o - \mathbf{P}^T\mathbf{b})$. The solution to this least-squares estimation problem is $\mathbf{b}^* = \mathbf{P}^T\mathbf{v}_o$. The best fit field of surface normals allowed by the model is $\mathbf{v}_o^* = \mathbf{PP}^T\mathbf{v}_o$. The deformed vector of azimuthal equidistant projection co-ordinates can be transformed back into a surface normal on the unit sphere using the inverse azimuthal equidistant projection equations given above.

Figure 2 illustrates this process. On the left a distribution of surface normals at one pixel in a model is shown as points on the unit sphere. The mean direction is shown as a red point. On the right the azimuthal equidistant projection of the points is shown with the mean point as the centre of projection. The first PCA axis is shown by the black line labelled PCA$_1$. This line corresponds to a great circle on the sphere through the mean direction which minimises the spherical distance to each point.

3 Fitting the Model to Intensity Images

Once trained, the statistical model represents the space of valid surface shapes. We can exploit this prior knowledge in order to help resolve the ambiguity in the shape-from-shading process. We do this using an iterative approach which can be posed as that of recovering the best-fit field of surface normals from the statistical model, subject to constraints provided by the image irradiance equation. According to Worthington and Hancock [12], when the surface reflectance follows Lambert's law, then the surface normal is constrained to fall on a cone whose axis is in the light source direction and whose opening angle is the inverse cosine of the normalised image brightness. This method commences from an initial configuration in which the surface normals reside on the irradiance cone and point in the direction of the local image gradient. The statistical model is fitted to recover a revised estimate of the surface normal directions. The best-fit surface normals are projected onto the nearest location on the irradiance cones. This process is iterated to convergence, and the height map for the surface recovered by integrating the final field of surface normals using the method of Frankot and Chellappa [15].

In the remainder of this subsection, we begin by briefly introducing the geometric shape-from-shading approach of Worthington and Hancock [12]. We then show how our statistical model may be integrated into this framework by providing a statistical update process for the field of surface normals.

3.1 Geometric Shape-from-Shading

If I is the measured image brightness, then according to Lambert's law $I = \mathbf{n}.\mathbf{s}$, where $\mathbf{s}$ is the light source direction. In general, the surface normal $\mathbf{n}$ can not be recovered from a single brightness measurement since it has two degrees of freedom corresponding to the elevation and azimuth angles on the unit sphere. In the Worthington and Hancock [12] iterative shape-from-shading framework, data-closeness is ensured by constraining the recovered surface normal to lie on the reflectance cone whose axis is aligned with the light-source vector $\mathbf{s}$ and whose opening angle is $\alpha = \arccos I$. At each iteration the surface normal is free to move to an off-cone position subject to smoothness or curvature consistency constraints. However, the hard irradiance constraint is re-imposed by rotating each surface normal back to its closest on-cone position. This process ensures that the recovered field of surface normals satisfies the image irradiance equation after every iteration.

Suppose that $(\mathbf{n}')^l(i,j)$ is an off-cone surface normal at iteration l of the algorithm, then the update equation is

$$\mathbf{n}^{l+1}(i,j) = \Theta(\mathbf{n}')^l(i,j) \tag{11}$$

where Θ is a rotation matrix computed from the apex angle α and the angle between $(\mathbf{n}')^l(i,j)$ and the light source direction $\mathbf{s}$. To restore the surface normal to the closest on-cone position it must be rotated by an angle $\theta = \alpha - \arccos\left[(\mathbf{n}')^l(i,j).\mathbf{s}\right]$ about the axis $(u,v,w)^T = (\mathbf{n}')^l(i,j) \times \mathbf{s}$. Hence, the rotation matrix is

$$\Theta = \begin{pmatrix} c + u^2c' & -ws + uvc' & vs + uwc' \\ ws + uvc' & c + v^2c' & -us + vwc' \\ -vs + uwc' & us + vwc' & c + w^2c' \end{pmatrix} \tag{12}$$

where $c = cos(\theta)$, $c' = 1 - c$ and $s = \sin(\theta)$.

The framework is initialised by placing the surface normals on their reflectance cones such that they are aligned in the direction opposite to that of the local image gradient. We use the irradiance cone constraint to fit our statistical model of surface normal variation to image brightness data.

3.2 Combining the Statistical Model and Geometric SFS

Our approach to fitting the model to intensity images uses the fields of surface normals estimated using the geometric shape-from-shading method described above. This is an iterative process in which we interleave the process of fitting the statistical model to the current field of estimated surface normals, and then re-enforcing the data-closeness constraint provided by Lambert's law by mapping the surface normals back onto their reflectance cones. The algorithm can be summarised as follows:

1. Calculate an initial estimate of the field of surface normals $\mathbf{n}$ by placing each normal on its reflectance cone in the direction of the negative local intensity gradient.
2. Each normal in the estimated field $\mathbf{n}$ undergoes an azimuthal equidistant projection (Equations (3) and (4)) to give a vector of transformed coordinates $\mathbf{v}_o$.
3. The vector of best fit model parameters is given by $\mathbf{b} = \mathbf{P}^T\mathbf{v}_o$.
4. The vector of transformed coordinates corresponding to the best-fit parameters is given by $\mathbf{v}' = (\mathbf{P}\mathbf{P}^T)\mathbf{v}_o$.
5. Using the inverse azimuthal equidistant projection (Equations (5) and (6)) find the off-cone best fit surface normal $\mathbf{n}'$ from $\mathbf{v}'$.
6. Find the on-cone surface normal $\mathbf{n}''$ by rotating the off-cone surface normal $\mathbf{n}'$ using Equation (11).
7. Test for convergence. If $\sum_{i,j} \arccos\left[\mathbf{n}(i,j).\mathbf{n}''(i,j)\right] < \epsilon$, where ϵ is a predetermined threshold, then stop and return $\mathbf{b}$ as the estimated model parameters and $\mathbf{n}''$ as the recovered needle map.
8. Make $\mathbf{n} = \mathbf{n}''$ and return to step 2.

4 Experiments

In this Section we present experiments with our method. There are four elements to this study. We commence by comparing the models obtained when range images are used to provide the training data. Second, we show the results of fitting the model to intensity data, and show the surface height data that can be reconstructed from the fitted fields of surface normals. Third, we illustrate how the fitted models can be used synthesise novel facial views.

4.1 Training the Model

In this section we describe how our model is constructed from real-world data. We commence by building a "ground truth" model using fields of surface normals extracted from range data. This allows us to show the utility of the model in capturing facial shape in a compact manner when trained on relatively 'clean' data.

The training set consisted of high resolution 3D scans of 100 male and 100 female subjects with a neutral expression. The scans were collected using a

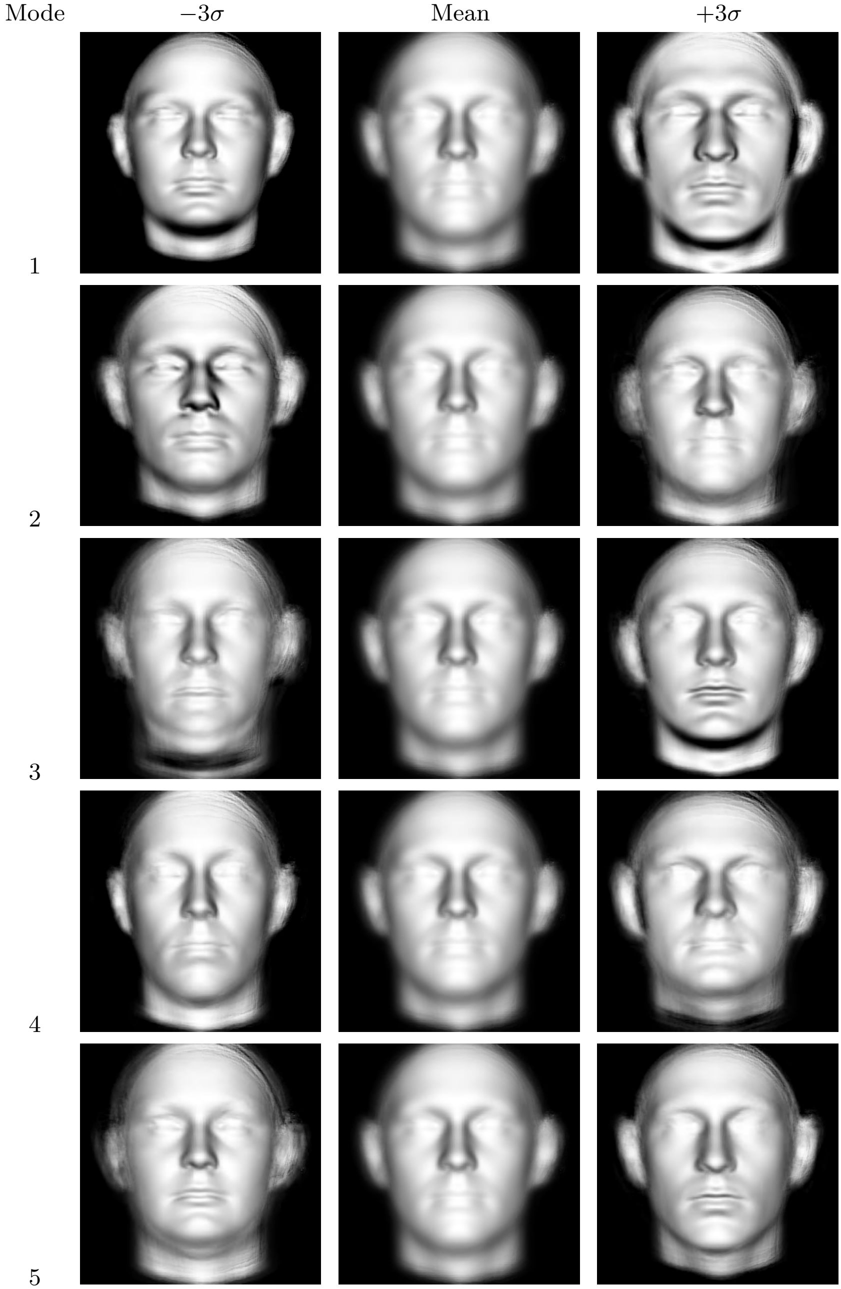


Fig. 3. The first five modes of variation of a statistical surface normal model trained on a set of facial needle maps extracted from range data. The mean face is shown in the central column and ± 3 standard deviations along each of the first 5 principal modes of variation are shown in the left and right columns

Cyberware$^{\text{TM}}$ 3030PS laser scanner. Each facial surface was aligned with a reference face using an off-the-shelf package (Delta$^{\text{TM}}$ from FarField Technology). This software uses a two-step, non-rigid registration algorithm which uses manually placed landmarks to produce an initial coarse registration. Subsequently the surface data itself is used to complete a fine registration. Fields of surface normals were extracted by orthographically projecting the 3 surface normal components onto a view plane positioned fronto-parallel to the aligned faces. These needle maps were used to investigate a statistical surface normal model trained on ground truth range data.

In Figure 3 we show the first 5 modes of variation of this model. In each case we deform the points under azimuthal equidistant projection by ±3 standard deviations along each of the first 5 principal axes. We then perform the inverse azimuthal equidistant projection before reilluminating the resulting needle maps with a point light source situated at the viewpoint and Lambertian reflectance.

The modes encode shape only, since the needle maps are invariant to illumination conditions and the training set contained no variation in expression. A number of the modes appear to correspond to obvious facial characteristics, for example mode 1 encodes head size and also seems to be correlated with gender. This is manifested in the broader jaw, brow and nose in the positive direction, all of which are masculine features. The fourth mode encodes the difference between long narrow faces and short wide faces, whereas the fifth mode encodes fat versus thin with a 'double chin' evident in the negative direction.

Figure 4 shows the eigenvalue associated with each of the decreasingly significant principal components. This shows that much of the model variance is

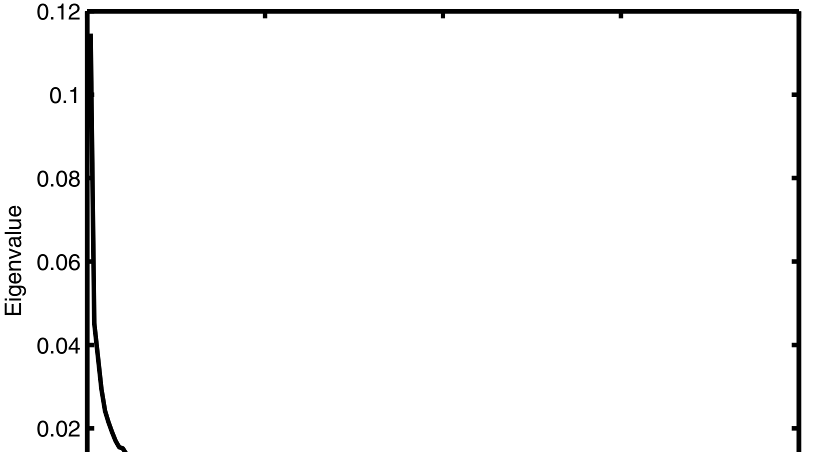


Fig. 4. Plot of eigenvalue versus eigenmode

captured by the first few modes. These principal components provide a low dimensional parameterisation of facial needle maps. In addition, because they only model the underlying shape they capture appearance in a sufficiently flexible manner to be able to generalise to 'out-of-sample' faces. This means needle maps not included in the training set can be accurately represented in a low dimensional space. A similar observation was made by Atick et al. [4] for facial surfaces. Notably the same is not true for 'black box' intensity based approaches such as Eigenfaces [16] or Active Appearance Models [17], which jointly model variation in illumination and reflectance properties as well as in identity.

In order to quantify this generalisation ability, we re-trained the model using a subset of the complete training set (180 needle maps). We then calculated the reconstruction error in representing the remaining 20 needle maps using this reduced model and found that the average angular percentage relative error in surface normal direction was 3.66% with a variance of 0.13%.

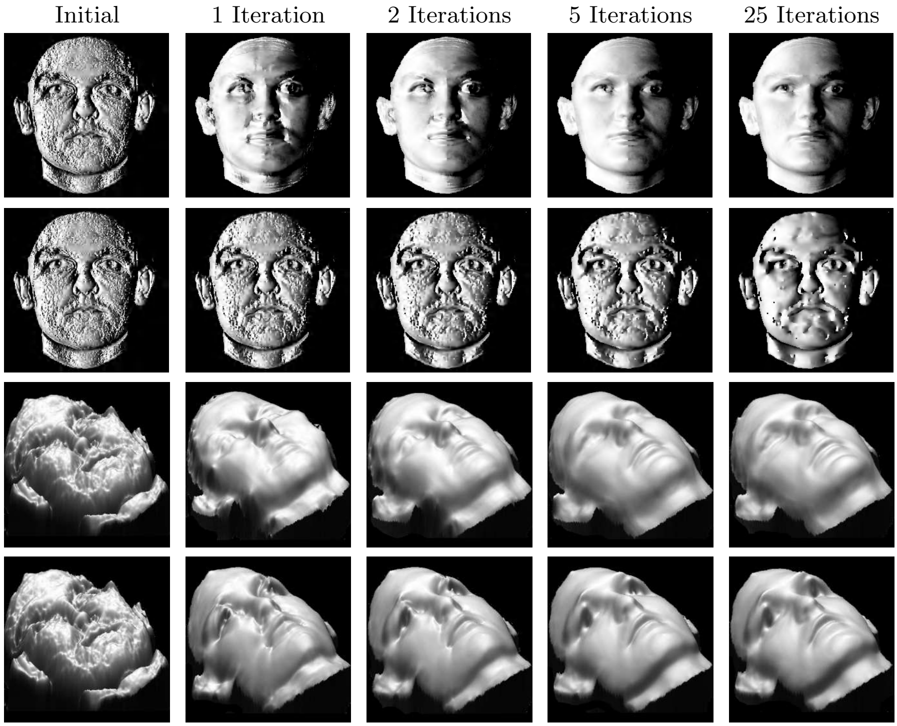


Fig. 5. Behaviour of the iterative fitting process over 25 iterations. The first row shows the recovered needle maps reilluminated by a light source with direction [-1 0 1]^T. For comparison the second row shows similarly reilluminated needle maps recovered by the Worthington and Hancock algorithm. The third and fourth rows show the surfaces recovered from $\mathbf{n}'$ (third row) and $\mathbf{n}''$ (fourth row)

4.2 Fitting the Model to Data

In this section we show how the statistical model may be fitted to intensity data using the method outlined in Section 3. We commence by considering the iterative behaviour of the algorithm. The top row of Figure 5 shows how a needle map develops over 25 iterations of the algorithm. Since the needle maps satisfy data-closeness at every iteration, they would all appear identical when rendered with a light source from the original direction ($[0\ 0\ 1]^T$). For this reason in the top row we show the needle maps reilluminated with a light source moved along the negative x-axis to subtend an angle of 45° with the viewing direction. After one iteration there is a significant global improvement in the recovered needle map. Subsequent iterations make more subtle improvements, helping to resolve convex/concave errors and sharpening defining features. For comparison the second row shows the corresponding needle maps recovered using the original curvature consistency constraint of Worthington and Hancock [12] reilluminated in the same manner. Although there is a steady improvement in the quality of the recovered normals, there are gross global errors as well as feature implosions around features such as the nose.

In Figure 5 we also show the surfaces recovered from the current best fit needle maps, $\mathbf{n}'$, (third row) and the needle maps which satisfy data-closeness, $\mathbf{n}''$, (bottom row) as the algorithm iterates. Surface recovery is effected using the method of Frankot and Chellappa [15]. As one would expect, the imposition of data-closeness results in errors in the recovered surface where there is variation in albedo, most notably around the eyes and eye-brows. In both sets there is a clear improvement in the recovered surface as the algorithm iterates. The implosion of the nose is corrected, the surface becomes smoother and finer details become evident, for example around the lips.

4.3 Synthesising Novel Views

In Figure 6 we show the surfaces recovered from the best fit needle maps. In the first and third rows the surfaces are shown rotated 30° about the vertical axis. The surfaces are rendered with Lambertian reflectance and an estimated albedo map. The light source remains fronto-parallel with respect to the face (i.e. from the original direction). The resulting synthesised images are near photo-realistic under a large change in viewpoint. Certainly, the results are comparable with those of Georghiades et al. [3] in which 7 input images were required per subject. Rows 2 and 4 of Figure 6 show the meshes of the recovered surfaces to allow inspection of the recovered shape alone. In Figure 7 we demonstrate that the recovered surface is sufficiently stable to synthesise images in both novel pose and novel illumination. We show the surface of subject 8 rendered as in the previous figure, except that the light source is circled from left profile to right profile.

5 Conclusions

We have shown how a statistical model of shape may be constructed from fields of surface normals using the azimuthal equidistant projection. We demonstrated

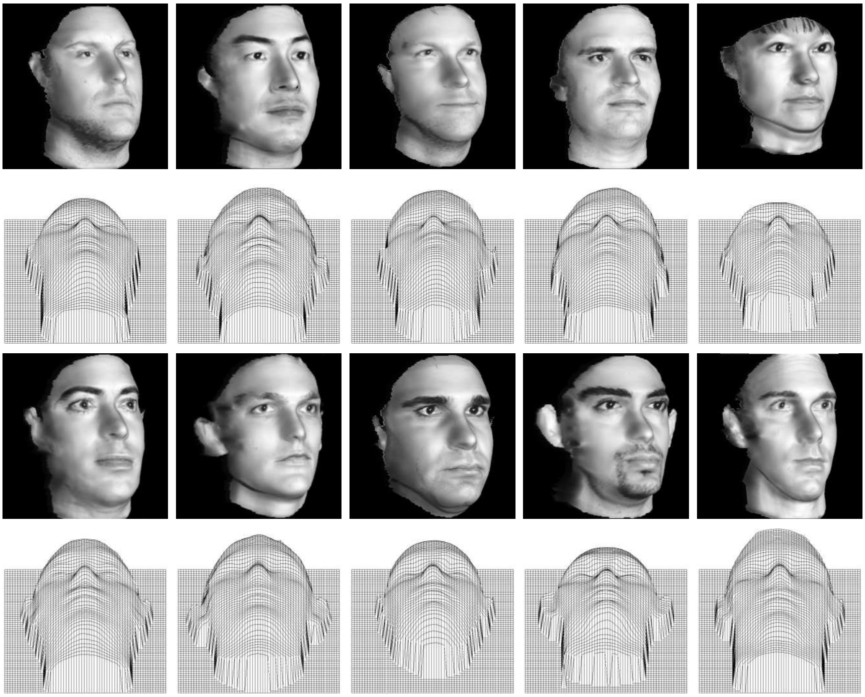


Fig. 6. Surfaces recovered from the ten subjects in the Yale B database. In the first and third rows, the surfaces are rendered with Lambertian reflectance and are shown rotated 30° about the vertical axis. The light source remains fronto-parallel with respect to the face. In the second and fourth rows the surface meshes are shown rotated 40° about the horizontal axis

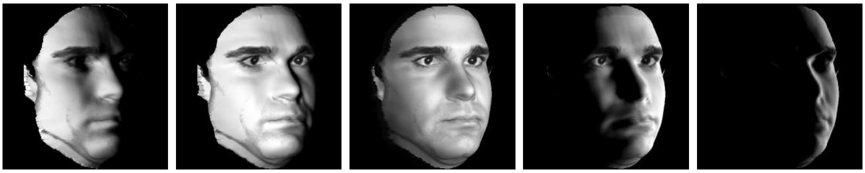


Fig. 7. Surface recovered from subject 8 of the Yale B database. The surface is again rendered with Lambertian reflectance and rotated 30° about the vertical axis. The light source is circled from full left profile to full right profile with respect to the face

that such a model trained on facial needle maps captures facial shape in a compact manner and is capable of generalising to out-of-sample faces.

The model may be fitted to image brightness data using an iterative method. The method alternates between surface normal estimation using a geometrical shape-from-shading method and fitting a statistical model to the field of surface normals. This process can be posed as that of recovering the best-fit field of surface normals from the statistical model, subject to constraints provided by the image irradiance equation. The method proves rapid to converge, and delivers realistic surfaces when the fields of surface normals are integrated.

Our future plans revolve around placing the iterative process in a statistical setting using the EM algorithm and a von-Mises distribution to model the likelihood for the surface normal data. We also plan to develop ways of aligning the model with images which are not in a frontal pose.

References

1. Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M.: Shape-from-shading: a survey. IEEE Trans. PAMI **21** (1999) 690–706
2. Gregory, R.L.: Knowledge in perception and illusion. Phil. Trans. R. Soc. Lond. B **352** (1997) 1121–1128
3. Georghiades, A., Belhumeur, P., Kriegman, D.: From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Trans. PAMI **23** (2001) 643–660
4. Atick, J.J., Griffin, P.A., Redlich, A.N.: Statistical approach to SFS: Reconstruction of 3D face surfaces from single 2D images. Neural Comp. **8** (1996) 1321–1340
5. Zhao, W.Y., Chellappa, R.: Illumination-insensitive face recognition using symmetric SFS. In: Proc. CVPR. (2000)
6. Samaras, D., Metaxas, D.: Illumination constraints in deformable models for shape and light direction estimation. IEEE Trans. PAMI **25** (2003) 247–264
7. Castelán, M., Hancock, E.R.: Acquiring height maps of faces from a single image. In: Proc. 3DPVT. (2004) 183–190
8. Prados, E., Faugeras, O.: A rigorous and realistic shape from shading method and some of its applications. Technical Report RR-5133, INRIA (2004)
9. Heap, T., Hogg, D.: Extending the point distribution model using polar coordinates. Image and Vision Computing **14** (1996) 589–599
10. Snyder, J.P.: Map Projections–A Working Manual, U.S.G.S. Professional Paper 1395. United States Government Printing Office, Washington D.C. (1987)
11. Cootes, T.F., J.Taylor, C., Cooper, D., Graham, J.: Training models of shape from sets of examples. In: Proc. BMVC. (1992) 9–18
12. Worthington, P.L., Hancock, E.R.: New constraints on data-closeness and needle map consistency for shape-from-shading. IEEE Trans. PAMI **21** (1999) 1250–1267
13. Worthington, P.L., Hancock, E.R.: Surface topography using shape-from-shading. Pattern Recognition **34** (2001) 823–840
14. Sirovich, L.: Turbulence and the dynamics of coherent structures. Quart. Applied Mathematics **XLV** (1987) 561–590
15. Frankot, R.T., Chellappa, R.: A method for enforcing integrability in shape from shading algorithms. IEEE Trans. PAMI **10** (1988) 439–451
16. Turk, M., Pentland, A.: Face recognition using eigenfaces. In: IEEE Conf. CVPR. (1991) 586–591
17. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. In: Proc. ECCV. (1998) 484–498

New Trends in Digital Shape Reconstruction

Tamás Várady[1,2] and Michael A. Facello[1]

[1] Raindrop Geomagic, Inc., P.O. Box 12219, Research Triangle Park,
NC 27709, USA
[2] Raindrop Geomagic Hungary, Infopark sétány. 1. Budapest 1117 Hungary
varady@geomagic.com
http://http://www.geomagic.com

Abstract. There are various segmentation and surfacing methods to create CAD models from measured data. First the difficulties of creating a good surface structure over a polygonal mesh are investigated, followed by investigating the most important approaches according to the amount of user interaction, computational efficiency and surface quality. References to commercial systems are also added. The focus of the paper is to present (i) automatic surfacing and (ii) functional decomposition. New demands and emerging technologies are also identified to trace out current trends in digital shape reconstruction.

1 Introduction

There is a rapidly growing demand to represent 3D objects in digital form. Complex computer models can be defined by a sequence of commands in CAD systems, but in many applications 'appropriate' digital replicas of existing physical objects are needed. The technological discipline that converts large measured point clouds into CAD models is called '3D Digital Photography' or 'Digital Shape Reconstruction' (DSR). We consciously avoid using the term 'Reverse Engineering' due to its negative connotation and its use in other areas.

We will investigate state-of-the-art solutions to this conversion process, and in particular, segmentation and surface generation. We assume that after data capture, the partial scans have been aligned and merged. The subsequent preprocessing operations include mesh repair, noise reduction, hole filling, smoothing and decimation. As a result, a *triangular polygonal mesh* is obtained that is the first (piecewise linear) approximation of the object. For reviewing recent advances in data acquisition and mesh processing, see [6, 13, 25].

Polygonal representations are—in fact—adequate for many applications; take *natural (organic) objects* formed by the laws of nature, or *artistic objects* created by human artists. There are several operations where the discretized representation is sufficient—graphical rendering, stereo-lithography or finite-element analysis. In these cases the structure of the object and the surface quality are not particularly important.

Engineering objects are created by human engineers, driven by different requirements of form and function. These objects obey physical laws (e.g. turbine

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 395–412, 2005.

blades); and/or satisfy special aesthetic requirements dictated by fashion (e.g. car bodies or consumer goods); and/or unite a set of mechanical constraints (e.g. a multi-axis gear box). For high-end engineering applications both the structure and surface quality are important, since these models will be exported into CAD/CAM systems for redesign, manufacture and analysis.

There is large group of organic, artistic, and engineering objects, where smooth and compact surface representations are needed, but rapid surfacing has a much higher priority than perfectly reconstructing the structure. In these cases, surface meshes of *quadrilateral tiles* are generated in an automatic manner.

In this paper, we deal with the problems of segmenting and approximating polygonal meshes. Approaches differ by the amount of user interaction, the efficiency of the computations and the quality of the surfaces. We will discuss how rapid surfacing methods evolve to obtain better surface qualities, and how functional decomposition techniques evolve to obtain high-quality surfaces in a more efficient manner.

2 Segmentation and Surfacing

2.1 Partitioning the Mesh

The key issue in digital shape reconstruction is *segmentation*, that is, how to partition the polygonal mesh into smaller regions. This corresponds to the yet unknown CAD model we want to create. The most widely accepted data structure is *boundary representation* (B-rep), which contains a collection of *faces*, each bounded by one or more *edge-loops*. Each region is the pre-image of a face in the final model; each face lies on a surface that was created by approximating the points of the given region. In order to achieve good surface quality the regions must be well-partitioned—only then can surface characteristics be recognized and appropriate connections be built.

2.2 Regions and B-Rep Models

Prior to investigating the types of region structures yielded by different DSR methods, we briefly characterize B-rep models from topological and geometrical points of view.

Faces. The simplest face structures cover the object with a *uniform* face-set, such as triangular meshes or quadrilateral tiles. A more general face structure permits *n-sided faces* with different numbers of sides. In addition to *simply connected* faces with a single perimeter loop, faces with *internal loops* often occur.

Face connections. At *Face-to-Face* connections, the corners of the neighboring faces are identical. At *Face-to-Multiple-Face* connections the edges of the adjacent faces are not identical, and a corner point may fall into the middle of the edge of the opposite face, thus creating so-called *T-nodes.*

Surfaces are given either in simple analytic form (natural quadrics and tori) or in parametric form using control points (Bezier and NURBS). The method of surface generation can be an important issue for high quality shape reconstruction.

Extrusions, rotational symmetries, or complex sweeps should be reconstructed and archived utilizing the related shape characteristics. In CAD, function and form are determined mainly by the relatively large *primary surfaces. Feature surfaces* are generally smaller and play a secondary role—they smoothly connect the adjacent primaries. The most frequent feature types are blends (fillets) and free-form steps.

Continuity. There is extensive literature on various types of continuity including parametric, geometric and numerical continuity [15]. Geometric continuity is interpreted by incident points, tangent planes and principle curvature vectors. For numerical continuity, we take the two adjacent faces and compute the difference of normal vectors and/or curvature vectors along the corresponding boundary points. Then we check whether they remain within the prescribed tolerances (NG^1 or NG^2). It is important to distinguish between *internal* and *connecting* continuity. For example, trimmed bi-cubic NURBS surfaces are internally C^2 continuous, but along their common boundary they are joined only with numerical continuity. (Otherwise, we have to represent the exact mathematical curves by very high degree equations, in non-standard format.) To match two trimmed surfaces with higher degree numerical continuity is a difficult computational issue.

Quadrilaterals or trimmed faces. A non-four-sided face with internal loops is either composed as a *collection* of four-sided surface elements, or we have to use *trimming curves* to cut out a valid part from an analytic or parametric surface. In the first case, the main difficulty is subdivision and assuring smooth connections between the sub-faces. In the second case, tolerance control of the 'curves on surfaces' and constructions to stitch the adjacent surface elements require special care.

2.3 Difficulties in Segmentation

As explained earlier the quality of segmentation strongly influences the quality of the final surface. Data sets are noisy and the edges of triangles on the mesh are generally not aligned with the edges of the faces. While a human engineer can immediately recognize the face structure of the object and can virtually partition it, DSR systems may need to perform a complex 'trial and error' process. Alternatively, they may ask for a certain amount of user assistance. A few related issues are discussed as follows.

Sharp edges. The detection of *sharp* edges is not obvious, since the measured data points rarely lie on the ideal edge, moreover, data capture is error prone along strong discontinuities. Extra effort is needed to extract sharp edges and repair the mesh accordingly, see for example [2].

Smooth edges. While sharp edges must be located at the exact position, the problem with *smooth* edges is that their ideal location is not known and it is hard to detect them. If we include points which should not be included, or miss points which should be included, surface quality will be poor. In Figure 1(a), if the exact circular boundary between the middle torus and right cylindrical regions is not

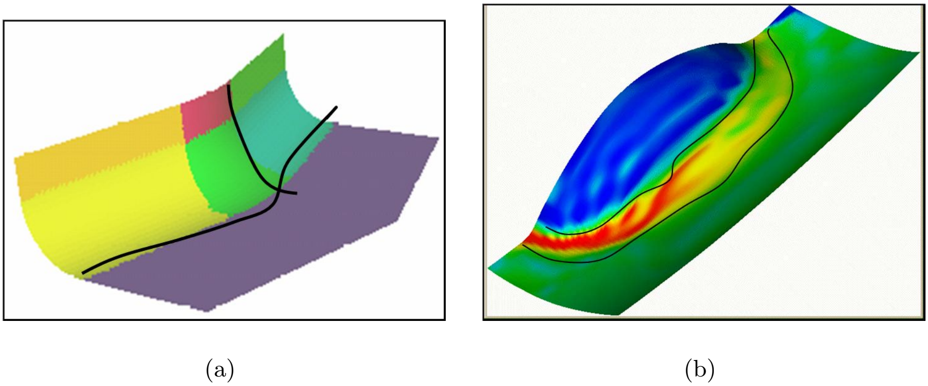


Fig. 1. Two examples of segmenting along smooth edges that will result in poor surface fitting

detected, it is impossible to reconstruct them in their best form, they will be roughly approximated. (We return to this topic later by discussing constrained surface fitting.) In Figure 1(b), if the smooth boundary of the variable radius rolling ball blend is not well-located the primary regions will accidentally contain data points from the blend, and the blend region will contain primary data points, killing surface quality.

Artificial edges. When dealing with complex regions containing hole loops and/or when only quadrilateral tiles are permitted, the regions need to be split into smaller ones by artificial edges. Thus surface portions which functionally would belong together are separated, and only lower degree, (numerical) continuity can be achieved. This results in loss of quality and preventing us to perform global surface fairing.

Aligned free-form patches. The most widely used free-form surfaces are represented by a regular grid of control points. They are given in $r = r(u, v)$ parametric form, and map a planar rectangle into a curved 3D quadrilateral. This representation is perfect for shapes where the control points and the 'intrinsic curvature variation of the shape' can be aligned to each other; however, quality gets worse when no such natural assignment exists. Figure 2 shows two examples: compare the evenness of the curvature distribution when the control points are aligned or non-aligned.

Hybrid patches. It is a crucial quality issue to find such a segmentation where the highly curved and relatively flat parts can be separated. In this case the total number of control points will be optimal, the surface quality improves and the models will be lighter. To approximate the flat parts, relatively few control points will be used and a dense grid of control points are needed only for the highly-curved parts. For hybrid patches the regions have not been separated by curvature, and they will be approximated inefficiently since to fit the

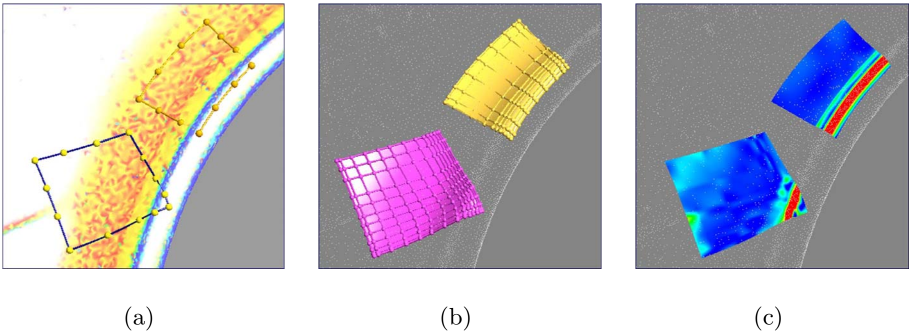


Fig. 2. (a) Two quadrilateral surfaces (b) Control points aligned (top) and non-aligned (bottom) (c) Related curvature distributions

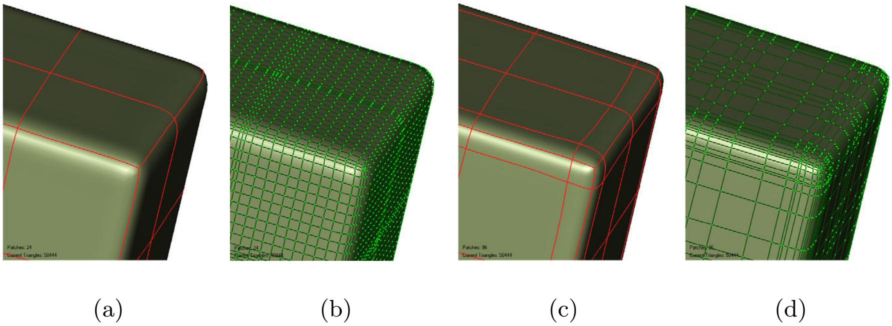


Fig. 3. (a,b) Hybrid curvature layout and control points (c,d) High curvature layout and control points

highly curved portions, superfluously full rows or columns need to be inserted into the control point grid. There is ongoing research to find alternative surface representations for non-regular structures reflecting the curvature changes, see subdivision surfaces [32] or T-splines [33], but these have not yet been integrated into commercial CAD/CAM systems.

2.4 Basic Approaches

There are five approaches that can be found in commercial systems and are worth analyzing from various points of view: (i) manual creation of quadrilaterals, (ii) manual segmentation with CAD functionality, (iii) automatic (rapid) creation of quadrilaterals, (iv) functional decomposition, and (v) template-based segmentation. This classification is not clear-cut, and contains overlaps, but it helps to gain a better understanding. A simple example, a portion of a CAD model, was chosen (Figure 4(a)) to show different surface structures. It consists of two trimmed surfaces—one planar and one free-form—being connected by a fillet surface.

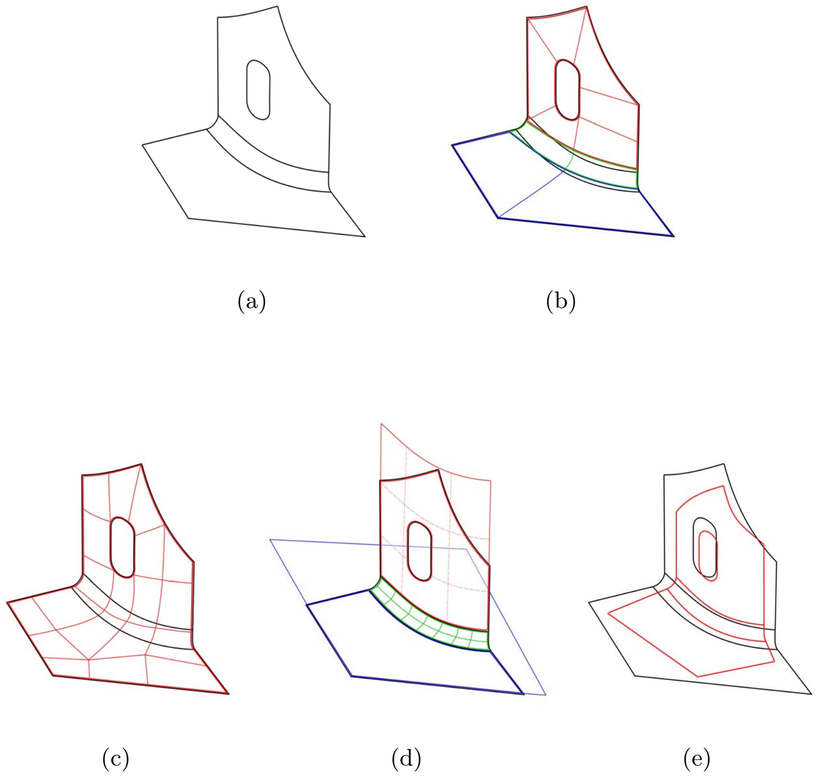


Fig. 4. (a) Simple CAD model, (b) Manually created quadrilaterals, (c) Automatic tiling, (d) Functional decomposition, (e) Templates

Manual creation of quadrilaterals. The main idea is to let the user drive and control the creation of a quadrilateral curve network. Typical representatives include [8, 26, 28, 29, 30]. Difficulties include the proper location of smooth boundaries, see middle blend in Figure 4(b). The lack of the fitting capability for trimmed regions leads to artificial subdivisions, which worsens internal continuity. The user has to fix—in advance—the boundaries of the four-sided surfaces; interpolating these and approximating the internal data points while also joining and tweaking the neighboring surface elements is difficult and may lead to undesirable surface artifacts. Finally, to create complex parts with this approach can be cumbersome.

Manual segmentation with CAD functionality. There are systems that offer a wide range of CAD operations to overcome some of the above mentioned deficiencies. The primary interest of these systems is conceptual design or styling. Only certain parts of the object are reconstructed using the point data; and the remaining parts are defined independently by standard CAD operations. Typically four-sided surfaces are fitted which are later trimmed by auxiliary surfaces or as a result of other operations. Problems may occur at connecting two adjacent surfaces: tweaking can improve connection quality, but may destroy internal smoothness.

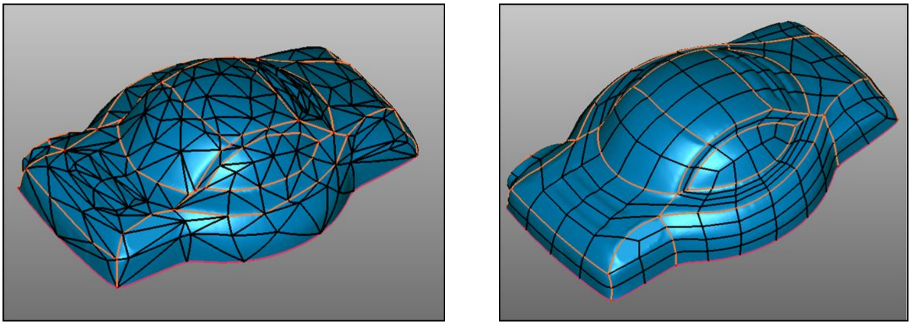
The most advanced manual systems with advanced CAD are [1, 16, 17, 28]. These systems also offer special tools to obtain Class A surfaces mainly used in the automotive industry. There is no exact mathematical definition of Class A; loosely speaking the surface curvature must be evenly distributed across the majority of the model. These systems tweak the control points or modify a set of section curves in a long tedious process until the desired, visually pleasing surface quality is achieved. The work is based on continuous visual feedback using isophotes and reflection lines.

Automatic (rapid) creation of quadrilaterals. This approach is capable of handling very detailed parts with minimal user assistance in a reasonably efficient manner, see [14, 28, 30]. Typically, a special quadrilateral patch layout is computed using 'Face-to-Face' NURBS tiles, with watertight G^1 (or G^2) continuity, see Figure 4(c). If necessary, the user can modify the tile boundaries or rearrange the tiles to obtain more pleasing patch layouts. Automatic surfacing is mostly a self-contained solution that provides reasonably good surface models in a relatively short time. The problems of artificial subdividing edges are also valid as in the previous approaches. The lack of good alignment can also become a problem.

Functional decomposition. The basic purpose of the fourth approach is to partition the mesh into faces as it might have been created by a CAD program. While in manual surfacing the user first defines the patch boundaries, here these are to be determined *automatically* during surface fitting by the DSR system. The process is based on a rough segmentation of the mesh: the created regions only reflect the *topological structure* and record their *dependencies*. It is important that the regions directly correspond to the trimmed faces of the final model; so there is no need for internal subdivisions. Moreover, the original surfaces can also be reproduced in untrimmed form as well [14].

For functional decomposition it is important to classify the regions as *primary* or *connecting features* and perform surface fitting in this order. Quality gains are possible only if special, constrained fitting algorithms are applied, see later. The result of the ideal process is illustrated in Figure 4(d) where the original surface elements—planar, free-form, fillet—are reconstructed having 'well-located' boundaries. This method provides the best possible internal surface quality, though there is only numerical continuity between the trimmed faces, as in standard CAD systems. Another disadvantage is that a certain amount of user assistance is needed for creating the regions and orienting the trimmed parametric surfaces. For very complex objects with hundreds of faces this method cannot compete with the automatic approach, though partial success has been reported for reconstructing conventional objects bounded exclusively by planes, natural quadrics and blends [3].

Templates. There are several applications when the structure of the object is known and can be retrieved. For example, somebody may want to reconstruct



(a)

(b)

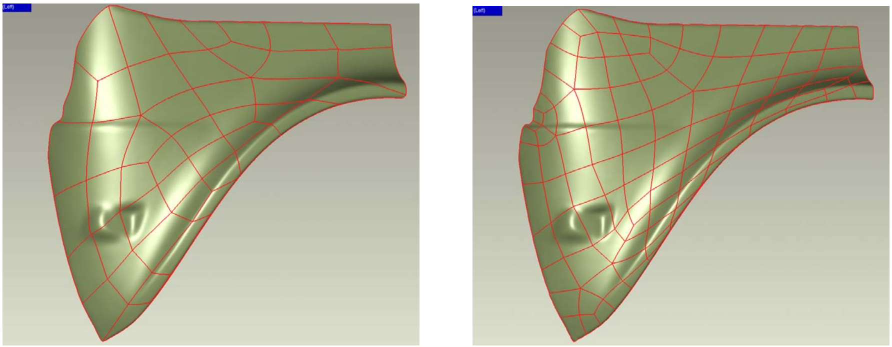
Fig. 5. Decimated triangulation and the final mesh of quadrilaterals

a family of objects with different dimensions; or create a similar, but altered car body panel; or generate shapes that look different, but are identical from a structural point of view, such as human faces. In these situations, the use of *templates* is recommended. A template is a curve network that carries the region structure and the best alignment with the existing polygonal mesh needs to be found. In Figure 4(e) the boundaries of the planar face and the vertical faces need to be snapped to the mesh, the hole loop needs to be displaced to some extent, and the fillet boundaries should be relocated according to the previous constraints. The template based approach naturally coexists with the manual, rapid surfacing, and functional decomposition approaches.

3 Automatic (Rapid) Surfacing

Automatic tiling. Automatic (rapid) surfacing is based on the automatic construction of a quadrilateral tiling. In order to produce appropriate partitioning for surface fitting, the tiling should adapt to the features of the shape by following its curvature, and minimize the number of non-degree-4 corners. Efforts in the past have produced various methods, including Voronoi-based methods [9] and decimation-based methods [23]. In the latter method, a decimated triangulation is used to construct the tiling [10], which is then remapped to the original model, see Figure 5(a). Various quad-mesh generation methods can also be used to produce a tiling, including advancing front methods [7] and triangle-to-quadrilateral mesh conversion [18].

In some situations, the tiling will not be sufficiently good, and further restructuring will be needed, as shown in Figure 5(b). Local or global operations can be defined to restructure a portion of the layout [24], and can be applied manually or automatically as a post-processing step. The resolution of the tiling can be controlled by the level of decimation or by prescribing the number of tiles. Figure 6 shows a car body panel approximated by two, G^1 continuous, quadrilateral meshes with target patch count 50 and 100.



(a)

(b)

Fig. 6. Two automatic layouts by different patch counts, (a) 50, (b) 100

Patch layout relaxation. Due to remapping distortions, the resulting patch layout may require improvement through relaxation [21]. The goals of the relaxation are to produce smooth curves that are well-distributed around the corners where they meet. Various types of relaxation may be required depending on the type of patch boundary. Boundaries that lie along or beside highly-curved features should be smoothed while maintaining their proximity to the feature; patch boundaries that lie in the middle of relatively flat regions can be relaxed without constraints.

Strengths and weaknesses. To sum it up: by the decimation based contour extraction and automatic tiling users can process large and complex models in an efficient way with reasonably tight tolerance and at least tangent plane continuity. If the automatic layout is not good enough further manual operations are needed which may take a longer time. The region boundaries are often not properly aligned, and the surface quality is poorer since contours may run in the middle of highly curve areas. Applying uniform control point density, though efficient, is not optimal. In the next paragraph we introduce a new concept in evolution, which can preserve the advantages of rapid surfacing, but makes significant steps towards overcoming these deficiencies.

Feature-based segmentation. Rapid surfacing systems make it possible to directly incorporate smooth or sharp edges into the decimated structure and build up the best layout accordingly. In the new approach the original triangular mesh is segmented directly. It is possible to extract locally estimated geometric properties using a set of points or triangles around a given vertex of the mesh, see for example [5, 27]. These are characterized by one or more numerical values called *indicators*. Indicators may come from differential geometry, such as the normal vector of the local tangent plane or the estimated mean curvature value. They may characterize more complex properties, such as the error of a fitted least-squares surface or the best fit direction of an extrusion or a local rotational

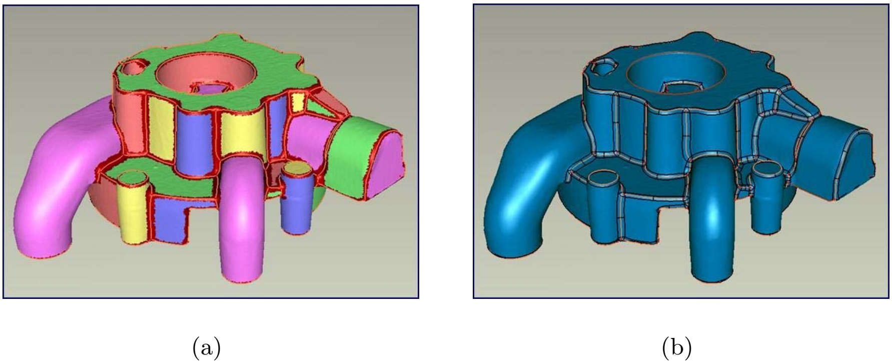
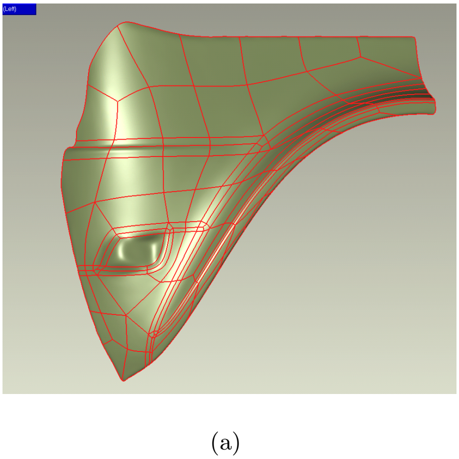

(a) (b)

Fig. 7. Automatic (a) feature-based segmentation, (b) feature boundary extraction

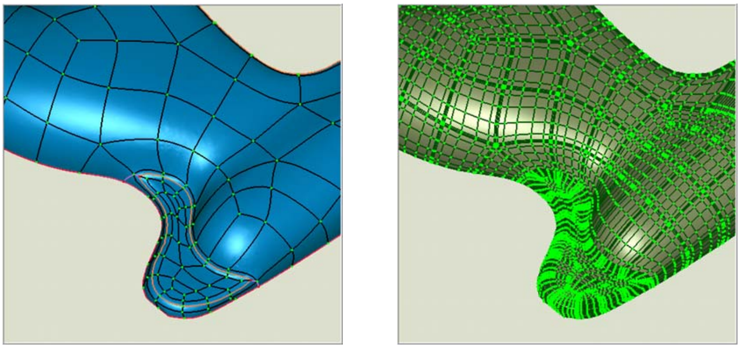


(a)

Fig. 8. Automatic feature-based segmentation

axis. Finally, they can highlight abstract properties, showing whether the point is considered 'similar' to its neighboring points or not. Indicators help driving automatic processes or giving visual hints to the user.

In order to deduce a consistent structure, good threshold values need to be found, which is often difficult and requires user expertise. Thresholds strongly depend on the quality of the scanned data and the shape of the object, and often—due to variations in the shape—no appropriate global value exists. There are new emerging techniques to separate highly curved and relatively flat regions without explicit threshold setting while guaranteeing the consistency of the obtained structures [11]. The result of the process is illustrated in Figure 7, where an object is segmented into larger regions being separated by small strips of feature regions (a). These strips (red) provide good initial estimates for the boundaries of the features which can later be incorporated into the final layout (b).



(a) (b)

Fig. 9. (a) Feature-based patch layout, (b) Feature-based adaptive fitting

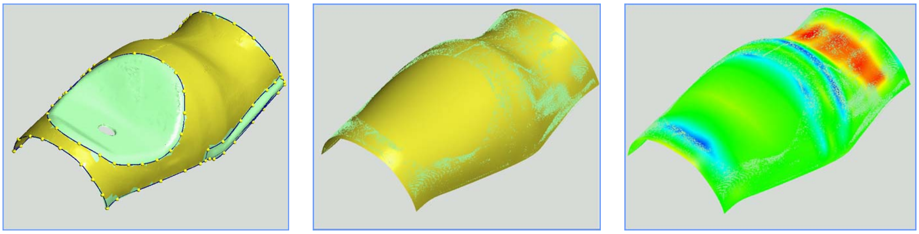
As it can be seen, this sort of segmentation creates a structure which is much closer to a standard CAD model representation than earlier. It provides significant improvements in alignment and reduces the number of patches with hybrid curvature. Another illustration is a feature-based layout for the previous car body panel in Figure 8. The number of patches is roughly the same as before (100), but their distribution is different reflecting the intrinsic characteristics of the shape. Surface quality and accuracy become better, and the need for manual rearrangements is drastically reduced. The number of control points can be significantly reduced using adaptive surface fitting methods to the highly curved and flat parts, as shown in Figure 9.

4 Functional Decomposition

The functional decomposition paradigm attempts to reproduce the design intent of CAD models from point clouds having the assumption that the object has been or might have been defined by a CAD system.

Natural extensions. We deal with the most general region structure with an arbitrary number of edges and an arbitrary number of internal loops. For further CAD/CAM processing, it is essential to reconstruct the original untrimmed surfaces with natural extensions within and outside the region boundaries. An example is given in Figure 10, where a NURBS surface is generated based on a partial point set. Looking at its curvature map the region boundaries can hardly be detected.

Curve networks. As it was explained earlier, in manual systems the *user curve network* not only defines the region structure, but at the same time pins down the surface boundaries over the mesh. In functional decomposition the network only roughly defines the regions and the final *system network* will be computed by the system. This is topologically *identical* to the user network, but its shape



(a) (b) (c)

Fig. 10. (a) Trimmed point region (b) Extended surface (c) Curvature map

is optimized and the curves are repositioned. The boundary curves are not 'a priori' fixed mesh curves, but *curves on surfaces*, (e.g. [31]) obtained through sequential surface fitting steps.

Dependencies. An important characteristic of functional decomposition is that adjacent surfaces depend on each other. This also corresponds to the logic of CAD/CAM systems: first the large primary surfaces (PR) are defined, followed by creating connecting features (CF) and finally N-patches (NP) or vertex blends. An example is shown in Figure 11(b), where the object consists of four primaries—`PR.Top`, `PR.Bottom`, `PR.Side`, `PR.Strip`, two connecting features—`CF1` and `CF2`, and two N-patches—`NP.Lamp` and `NP.Small`. `CF1` is a *free-form step*, `CF2` is a *rolling-ball blend*. `NP.Lamp` depends only on `PR.Bottom`, having its perimeter loop constrained to lie on it. `NP.Small` depends on four surfaces, namely `PR.Bottom`, `PR.Side`, `PR.Strip` and `CF2`.

Detect surface types. It is an emerging need to represent surfaces with the best possible surface representation. Related hypotheses can be automatically formed having the segmented point regions and their indicator sets, see also [3]. If this type belongs to some analytic, translational or rotational type, a *single best fit* step is performed instead of using iterative free-form fitting. For example, if we have a cylindrical point set, we check the tolerances of the best cylinder computed by the system. If it is out of tolerance we have to reclassify the region to be conical, rotational or eventually free-form, and repeat the fitting steps. Free-form surfaces are different in this sense: it is always possible to insert further degrees of freedom and fit a surface *iteratively* until the required accuracy is reached.

Surface fitting with NURBS. Generally the squared distances between data points and the corresponding points of an unknown surface are minimized. For NURBS, there exist infinitely many 'possible' surfaces, even if we fix the degree of the patches. The control point configuration, i.e., the number of rows and columns are unknown. Their 'best' locations in space are also unknown and the parametrization of the data points (i.e., the best correspondence) is also unknown. Furthermore to find a good ratio between *tight fitting* and an *even curvature distribution* is difficult. The ratio is unknown and shape dependent.

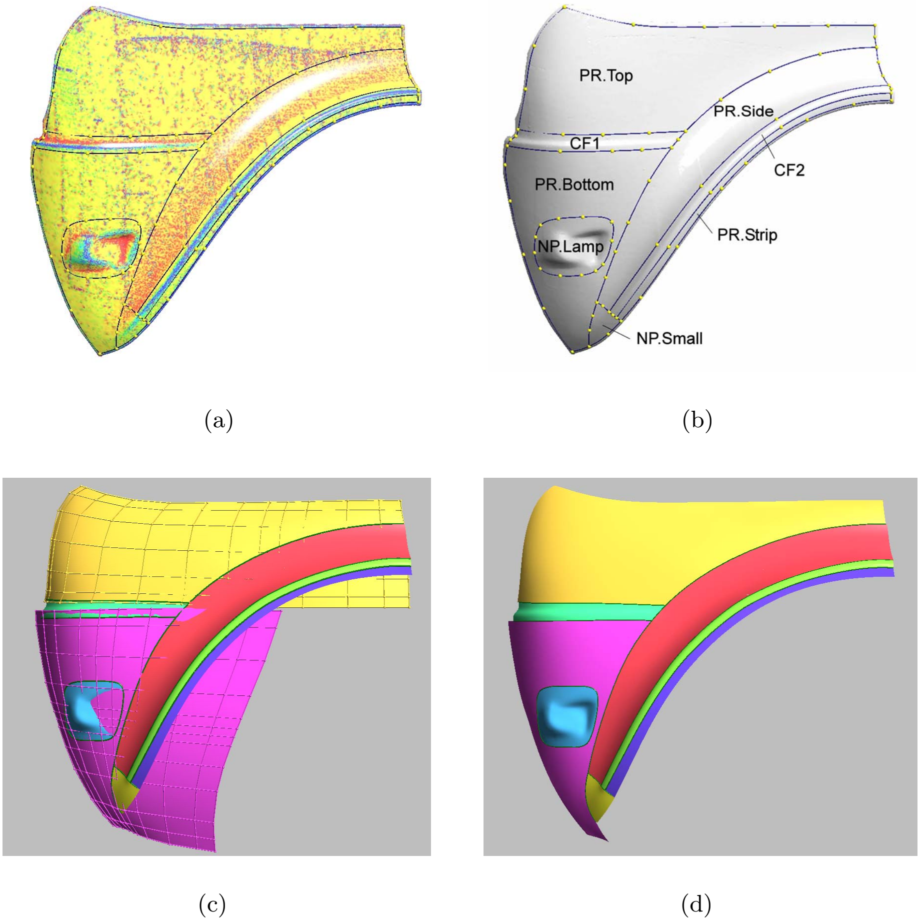
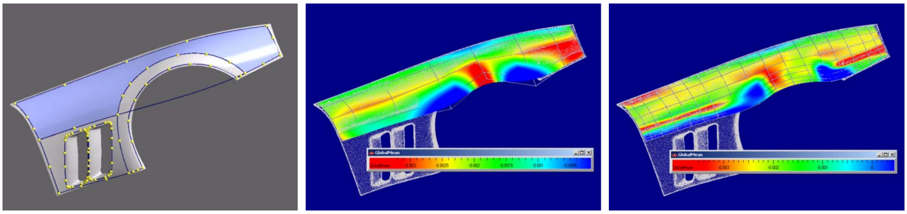


Fig. 11. Car body panel: (a) numerical curvature map, (b) region structure, (c) reconstruction with untrimmed surfaces (Top and Bottom), (d) final trimmed model

The only thing we know is that very tight fitting kills smoothness, and oversmoothed surfaces cannot closely approximate data points. So we face a very complex, non-linear optimization problem, and there are many different mathematical techniques to fit surfaces with high quality within an acceptable time.

Self-tuning algorithms. Many systems let the user to manually set the number of control points. This is a simple solution: if the approximation error is too big, the user sets a larger number and tries again. If the approximation error is within the requested range, the user may try to refit with a smaller number of control points and evaluate again whether the surface is still acceptable. Things get complicated, or soon very tedious, when both the number of rows and columns need to be set while having many surface elements with different local shape characteristics.



(a) (b) (c)

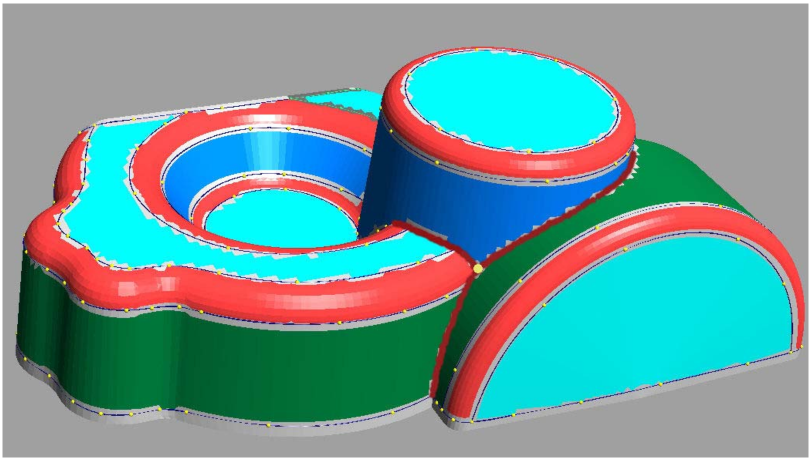
Fig. 12. (a) Region on a car body panel, (b) Fitting with loose tolerance; control net and curvature map, (c) Fitting with tight tolerance

The real demand is to specify a *distance tolerance* for the whole shape, and ask the system to iteratively optimize all the above parameters until the surface approximates all the data points within tolerance. To reach this goal *self-tuning* algorithms gain more importance [36]. In addition to the tolerance a so-called *outlier percentage* is also needed to ignore noisy data points. Then by breaking the process into linear subtasks, it is possible to iteratively compute the optimal number of control points, their optimal 3D positions, an optimal parametrization and a good smoothness weight, which will finally assure that the smoothest possible surface is obtained while the tolerance criterion is still satisfied.

Automatic parametrization for irregular regions. There is another difference between fitting analytic and free-form surfaces. For analytic surfaces only the data points within the trimmed region matter; for free-form surfaces the orientation and initial parametrization of the parametric rectangle is also important. There are four-sided regions where the assignment can be done in a natural way, but for the general case, further information is needed. This can be computed through the analysis of flattened 3D meshes in the domain plane, see for example [12, 20].

Special fitting algorithms improve quality. We have pointed out that specific surface fitting algorithms yield better surface qualities. This is particularly true for connecting features where not only the type-dependent fitting algorithms need to be applied—say for rolling ball or constant range blends—but also further constraints must be satisfied to smoothly join to already existing primary surfaces [19].

Constrained fitting for groups of surfaces. It is also an emerging request to detect constraints between the individual faces of the object. For example, Figure 13 shows the well-known benchmark, where the object is bounded by conventional surface elements and blends. To obtain the best surface quality, the three planar point regions (light blue) with z-normal should be fit together, having related constraints satisfied. Moreover, the axes of the conical surfaces (dark blue) should also be constrained to be parallel to this z-axis. Constrained fitting is a relatively new research area, with good results being reported for conventional geometry



(a)

Fig. 13. Darmstadt benchmark: classified regions

[4, 37], but several problems remain unsolved where constraints need to be enforced for groups of free-form surface elements.

Beautifying surface models. The last emerging request is to beautify the obtained models as much as possible. This is part of the previous constrained fitting problem since regularities must be detected. At the same time our starting point can be an already reconstructed geometric model. The main issue is how we can make it better by recognizing likely local and global properties, such as parallelism, concentricity, rotational or mirror symmetry. Results for a restricted object class have been reported in [22].

5 Conclusion

There is an enormous ongoing effort in digital shape reconstruction. An ideal system would automatically find the best-suited topological structure, would recognize the most likely underlying surface types, would detect the appropriate connecting elements and stitch them together into a single model with perfect sharp and smooth edges. The system should combine the strengths of tiling for artistic or highly detailed portions and functional decomposition for regions following high end CAD design philosophies. The user interaction would be minimal and, of course, the system would produce high quality surfaces everywhere. Such an ideal system does not exist yet, but there are alternative partial solutions where there is significant progress in at least one of the above areas.

CAD systems extend their digital reconstruction functionality, DSR systems tend to add more CAD functionality. Rapid surfacing systems can reproduce more and more complex parts without losing efficiency and tangent plane continuity. They move towards feature-based segmentation with more CAD-like structures and surfaces, which leads to the generation of better surface models with less user intervention. Functional decomposition approaches put emphasis on simplifying the creation of region structures, building dependencies and providing stable algorithms to fitting and stitching smooth, trimmed regions. Both the quality of the surfaces and the efficiency of the algorithms are improving. There are ongoing attempts to automate the detection of surface types and fit accordingly using special fitting algorithms. This is often coupled with various automatic and user-defined constraints for surface groups in order to assure smooth connections and other regular properties. The authors believe that in the not very distant future, it will be possible to merge the above partial solutions into a single system, when we will get very close to the functionality of the ultimate digital shape reconstruction system.

Acknowledgments. Figures in this paper were generated by the digital shape reconstruction products of Raindrop Geomagic Inc., Geomagic Studio and Cadmus Fashion. The support of two NSF Small Business Innovation, Phase II Grants are acknowledged: "Automatic Creation of NURBS Patches from Triangulated Surfaces" and "Creating Functionally Decomposed Surface Models from Measured Data".

References

1. Alias Studio Tools: http://www.alias.com
2. M. Attene, B. Falcidieno, J. Rossignac, M.Spagnuolo. Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces, In: *Eurographics Symposium on Geometry Processing*, Eds.: L. Kobbelt, P. Schreder, H. Hoppe, The Eurographics Association, (2003).
3. P. Benkő, R. R. Martin, T. Várady. Algorithms for Reverse Engineering Boundary Representation Models. *Computer Aided Design*, **33**, (2001), 839–851.
4. P. Benkő, L. Andor, G. Kós, R. R. Martin, T. Várady. Constrained Fitting in Reverse Engineering. *Computer Aided Geometric Design*, **19**, 1, (2002), 173–205.
5. P. Benkő and T. Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design* **36**, (2004), 511–523.
6. F. Bernardini and H. Rushmeier. The 3D Model Acquisition Pipeline. *Computer Graphics Forum*, **21** (3), (2002), 149–172.
7. T. D. Blacker and M. B. Stephenson. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *International Journal for Numerical Methods in Engineering*, **32**, 849–866.
8. CopyCad: http://www.delcam.com
9. M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. *Computer Graphics*, Proc. Siggraph (1996), 325–334.

10. H. Edelsbrunner, M.A.Facello, P. Fu, J. Qian, D.V. Nekhayev. Wrapping 3D scanning data. In: *Proc. SPIE Vol. 3313, Three-Dimensional Image Capture and Applications*, Eds.: R. N. Ellson, J. H. Nurrep, (1998) 148–158.
11. H. Edelsbrunner, J. Harer and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete Comput. Geom.* **30** (2003), 87–107.
12. M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, **14**, (1997), 231–250.
13. M. Garland. Multiresolution modeling: Survey and future opportunities. In: *Eurographics '99—State of the Art Reports*, (1999), 111–131.
14. Geomagic Studio: http://www.geomagic.com
15. J. Hoschek, D. Lasser: *Fundamentals of Computer Aided Geometric Design,* A. K. Peters, Wellesley, 1993.
16. ICEM Surf and RevEng: http://www.icem.com
17. ImageWare: http://www.ugs.com/products/nx/imageware
18. T. Itoh and K. Shimada. Automatic Conversion of Triangular Meshes Into Quadrilateral Meshes with Directionality. *International Journal of CAD/CAM*, **1** (2002), 11–21.
19. G. Kós, R. R. Martin, and T. Várady. Methods to recover constant radius rolling ball blends in reverse engineering. *Computer Aided Geometric Design* **17**, (2000), 127–160.
20. G. Kós and T. Várady. Parameterizing Complex Triangular Meshes. In: *"Curve and Surface Design*, Saint-Malo 2002, Eds: T. Lyche, M-L. Mazure, L. L. Schumaker, Nashboro Press, (2003), 265–274.
21. V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygonal meshes. *Computer Graphics*, Proc. Siggraph 1996, 313–324.
22. F.C. Langbein, A.D. Marshall, R.R. Martin. Choosing consistent constraints for beautification of reverse engineered geometric models. *Computer-Aided Design* **36**, (2004), 261–278.
23. A. Lee, W. Sweldens, P. Schroeder, L. Cowsar and D. Dobkin. MAPS: multiresolution adaptive parametrization of surfaces. *Comput. Graphics*, Proc. SIGGRAPH, (1998), 95–104.
24. A. Nakamoto. Diagonal transformations in quadrangulations of surfaces. *J. Graph Theory* **21** (1996), 289–299.
25. S. J. Owen. A Survey of Unstructured Mesh Generation Technology. *Proceedings 7th International Meshing Roundtable, Dearborn, MI*, October 1998. URL: http://www.andrew.cmu.edu/user/sowen/survey/index.html.
26. ParaForm: http://www.metris.com
27. S. Petitjean. A survey of methods for recovering quadrics in triangle meshes, *ACM Computing Surveys*, (2002).
28. PointMaster: http://www.knotenpunkt.com
29. PolyWorks: http://www.innovmetric.com
30. RapidForm: http://www.rapidform.com
31. G. Renner, V. Weiss. Exact and approximate computation of B-spline curves on surfaces, *Computer-Aided Design*, **36**,(2004), pp 351–362.
32. M. Sabin. Subdivision Surfaces, Chapter 12, *Handbook of Computer Aided Geometric Design*, Eds.: G. Farin, J. Hoschek, M.-S. Kim, NORTH-HOLLAND, 2002.
33. T. W. Sederberg, J. Zheng, A. Bakenov, A.H. Nasri. T-splines and T-NURCCs. *ACM Transactions on Graphics* **23**, (2003), 477–484.

34. TEBIS: http://www.tebis.com
35. T. Várady, R. Martin: Reverse Engineering, Chapter 26, *Handbook of Computer Aided Geometric Design*, Eds.: G. Farin, J. Hoschek, M.-S. Kim, NORTH-HOLLAND, 2002.
36. V. Weiss, L. Andor, G. Renner, T. V'arady. Advanced surface fitting techniques. *Computer Aided Geometric Design*, **19**, 1, (2002), 19–42.
37. N. Werghi,R. Fisher, C. Robertson, A. Ashbrook. Object reconstruction by incorporating geometric constraints in reverse engineering. *Computer-Aided Design*, **31**, (1999), 363–399.

Backward Errors and Condition Numbers of Regular and Singular Points on Algebraic Curves

Joab R. Winkler

Sheffield University, Sheffield, United Kingdom
j.winkler@dcs.shef.ac.uk
http://www.dcs.shef.ac.uk/∼joab

Abstract. Expressions for the componentwise and normwise backward errors at an arbitrary point on an algebraic curve are derived and the formulae that relate them to the condition numbers are established. The expressions for the condition numbers at a regular point that is near a singular point are examined and it is shown that restrictions on their validity arise in this circumstance. In particular, the lowest order approximation that is used when condition numbers are derived places tight bounds on the maximum allowable perturbation on this class of point. An example that illustrates this limitation is given.

1 Introduction

Geometric modelling requires that a large number of numerical operations be performed on the equations that define curves and surfaces, and this has been the motivation for extensive research into the numerical stability of curves and surfaces that are defined in the Bernstein basis. This paper continues this investigation by considering some aspects of the numerical condition and backward error of an arbitrary point on an algebraic curve.

Condition estimation and backward error analysis play an essential role in numerical computation because the condition number of a problem is a measure of the sensitivity of the solution to perturbations in the data, and the backward error is a measure of the minimum distance between the problem whose solution is sought, and the problem whose solution has been computed [2]. These concepts are well developed for linear algebraic and polynomial equations, and Farouki and Rajan [1] have developed expressions for the componentwise condition number at regular and singular points on an algebraic curve. In this paper, expressions for the componentwise and normwise backward errors at an arbitrary point on an algebraic curve are established, and the formulae that relate them to the condition numbers are established.

A planar algebraic curve is defined by the zero set of a bivariate function $f(x,y)$,

$$f(x,y)=\sum_{i=1}^{m}a_i\phi_i(x,y)=0, \tag{1}$$

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 413–433, 2005.

where $a = \{a_i\}_{i=1}^{m}$ is the vector of coefficients, which are assumed to be real, and $\phi(x,y) = \{\phi_i(x,y)\}_{i=1}^{m}$ is a vector of bivariate polynomial basis functions. A polynomial of degree n in d variables has $\binom{n+d}{d}$ coefficients, and since $d = 2$, it follows that the number of coefficients in (1) is $m = \frac{(n+2)(n+1)}{2}$.

The derivation of expressions for the condition numbers at a point on an algebraic curve requires that a distinction be made between regular and singular points, and these terms are now defined [3].

Definition 1. *A point P on a curve $f = 0$ has multiplicity r if all partial derivatives of f up to and including the $(r-1)$-th vanish at P, but at least one r-th partial derivative does not vanish at P.*

This definition leads to the following definitions:

Definition 2. *A point P with coordinates (x_α, y_α) on a curve $f = 0$ is regular or simple if its multiplicity is 1:*

$$\nabla f = \begin{bmatrix} f_x & f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \neq \begin{bmatrix} 0 & 0 \end{bmatrix} \quad \text{at } x = x_\alpha,\ y = y_\alpha.$$

Definition 3. *A point P with coordinates (x_α, y_α) on a curve $f = 0$ is singular if its multiplicity is greater than 1.*

For example, the point P on a planar curve has a singularity of multiplicity 2 if

$$\nabla f = \begin{bmatrix} f_x & f_y \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad \text{at } x = x_\alpha,\ y = y_\alpha,$$

but one or more of the derivatives

$$\frac{\partial^2 f}{\partial x^2}, \qquad \frac{\partial^2 f}{\partial xy} \qquad \text{and} \qquad \frac{\partial^2 f}{\partial y^2},$$

do not vanish at P.

Expressions for the componentwise and normwise backward errors at a point P on an algebraic curve are derived in Section 2, and expressions for the condition numbers are reviewed in Section 3. It is shown that these two quantities are related by a simple expression that is dependent on the multiplicity of P.

There exists an important difference between the condition numbers and backward errors because the former require that the lowest order term in a Taylor expansion be used, which necessarily imposes restrictions on the class of point for which these formulae are valid. This feature of condition numbers must be compared with the expressions for the backward errors, for which the derivations do not require approximations. These restrictions on the class of points for which the condition numbers are valid are considered in Section 4, and an example that illustrates these limitations is given. Section 5 contains a summary of the paper and a discussion of further work.

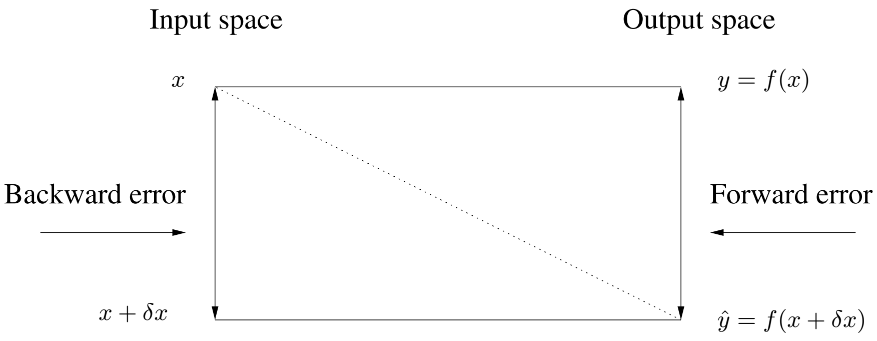


Fig. 1. The forward and backward errors for $y = f(x)$

2 Backward Error

The difference between the forward error and backward error of a problem is illustrated in Figure 1, which is reproduced from [2]. Specifically, the forward error is measured in the output or solution space, and the backward error is measured in the input or data space. The exact data x yields the exact value $y = f(x)$, and the perturbed data $\hat{x} = x + \delta x$ yields the perturbed value $\hat{y} = f(\hat{x}) = f(x + \delta x)$. The dotted line in Figure 1 represents the computation that is performed.

The forward error is defined as the relative error in the solution,

$$\frac{|\hat{y} - y|}{|y|},$$

and the backward error is based on the observation that the computed solution $\hat{y}$ is the exact solution of the problem with perturbed data $\hat{x}$. If it is assumed that roundoff errors can be interpreted as errors in the data, then the combined effects of these errors, and errors in the data, can be represented by $\hat{x}$. It follows, therefore, that although the exact or specified data is x, the solution of a problem whose data is $\hat{x}$ has been computed. The backward error is defined as

$$\frac{|\hat{x} - x|}{|x|} = \frac{|\delta x|}{|x|},$$

which is of the same form as the forward error, but measured in the input or data space, rather than the output or solution space.

The backward error is defined for componentwise and normwise perturbations, and these are considered in Sections 2.1 and 2.2 respectively.

2.1 Componentwise Backward Error

The componentwise backward error of a point $(\tilde{x}_0, \tilde{y}_0)$ on the curve (1) is defined, and an expression for it is then derived.

Definition 4. *The componentwise backward error of the approximation* $(\tilde{x}_0, \tilde{y}_0)$*, which may be complex, of the point* (x_0, y_0) *that lies on the curve* $f(x,y) = 0$ *is defined as*

$$\eta_c(\tilde{x}_0, \tilde{y}_0) = \min\left\{ \varepsilon_c : \sum_{i=1}^{m} \tilde{a}_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right) = 0 \ \text{ and } \ \left|\delta a_i\right| \leq \varepsilon_c \left|a_i\right| ; \tilde{a} = a + \delta a \right\},$$

where $\delta a = \left\{\delta a_i\right\}_{i=1}^{m}$.

Theorem 1. *The componentwise backward error of the approximation* $(\tilde{x}_0, \tilde{y}_0)$*, which may be complex, of the point* (x_0, y_0) *that lies on the curve* $f(x,y) = 0$ *is given by*

$$\eta_c(\tilde{x}_0, \tilde{y}_0) = \frac{\left|f\left(\tilde{x}_0, \tilde{y}_0\right)\right|}{\sum_{i=1}^{m} \left|a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right)\right|}. \qquad (2)$$

The perturbations in the coefficients that achieve this backward error are

$$\delta a_k = -\left(\frac{\left|a_k\right| f\left(\tilde{x}_0, \tilde{y}_0\right)}{\sum_{i=1}^{m} \left|a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right)\right|}\right)\left(\frac{\overline{\phi_k\left(\tilde{x}_0, \tilde{y}_0\right)}}{\left|\phi_k\left(\tilde{x}_0, \tilde{y}_0\right)\right|}\right), \qquad k = 1, \ldots, m, \qquad (3)$$

where $\overline{(\cdot)}$ *denotes the complex conjugate of* $(\cdot)$.

Proof. By definition, $\tilde{a}_i = a_i + \delta a_i, i = 1, \ldots, m$, and thus

$$\begin{aligned}\sum_{i=1}^{m} \tilde{a}_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right) &= \sum_{i=1}^{m} a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right) + \sum_{i=1}^{m} \delta a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right) \\ &= f\left(\tilde{x}_0, \tilde{y}_0\right) + \sum_{i=1}^{m} \delta a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right). \qquad (4)\end{aligned}$$

By assumption, the term on the left hand side is equal to zero, and thus

$$\begin{aligned}\left|f\left(\tilde{x}_0, \tilde{y}_0\right)\right| &= \left|\sum_{i=1}^{m} \delta a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right)\right| \\ &\leq \sum_{i=1}^{m} \left|\frac{\delta a_i}{a_i}\right| \left|a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right)\right| \\ &\leq \max_k \left|\frac{\delta a_k}{a_k}\right| \sum_{i=1}^{m} \left|a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right)\right|.\end{aligned}$$

It follows that

$$\varepsilon_c \geq \frac{\left|f\left(\tilde{x}_0, \tilde{y}_0\right)\right|}{\sum_{i=1}^{m} \left|a_i \phi_i\left(\tilde{x}_0, \tilde{y}_0\right)\right|},$$

and the result (2) is established.

Consider now the perturbations in the coefficients that achieve this backward error. By definition, these perturbations must satisfy

$$\eta_c(\tilde{x}_0,\tilde{y}_0) = \min\left\{\varepsilon_c : \varepsilon_c \geq \left|\frac{\delta a_k}{a_k}\right|, \, k=1,\ldots,m\right\}, \tag{5}$$

and (4) implies that they must also satisfy

$$f\left(\tilde{x}_0,\tilde{y}_0\right) = -\sum_{i=1}^{m} \delta a_i \phi_i\left(\tilde{x}_0,\tilde{y}_0\right). \tag{6}$$

Consider the perturbations, from (5), $|\delta a_k| = \eta_c(\tilde{x}_0,\tilde{y}_0)\,|a_k|$, or equivalently,

$$|\delta a_k| = \frac{|a_k|\left|f\left(\tilde{x}_0,\tilde{y}_0\right)\right|}{\sum_{i=1}^{m}\left|a_i\phi_i\left(\tilde{x}_0,\tilde{y}_0\right)\right|}, \qquad k=1,\ldots,m. \tag{7}$$

It must be verified that these perturbations also satisfy (6), and this is now established.

It follows from (7) that δa_k is given by

$$\delta a_k = \frac{a_k f\left(\tilde{x}_0,\tilde{y}_0\right)}{\sum_{i=1}^{m}\left|a_i\phi_i\left(\tilde{x}_0,\tilde{y}_0\right)\right|} h_k\left(\tilde{x}_0,\tilde{y}_0\right), \quad \left|h_k\left(\tilde{x}_0,\tilde{y}_0\right)\right| = 1, \quad k=1,\ldots,m, \tag{8}$$

where each of the functions $h_k\left(\tilde{x}_0,\tilde{y}_0\right)$ is of unit magnitude and to be determined. The substitution of this equation into the right hand side of (6) yields

$$f\left(\tilde{x}_0,\tilde{y}_0\right) = -\frac{\sum_{k=1}^{m} a_k f\left(\tilde{x}_0,\tilde{y}_0\right)\phi_k\left(\tilde{x}_0,\tilde{y}_0\right)h_k\left(\tilde{x}_0,\tilde{y}_0\right)}{\sum_{i=1}^{m}\left|a_i\phi_i\left(\tilde{x}_0,\tilde{y}_0\right)\right|},$$

and this equation determines the functions $h_k\left(\tilde{x}_0,\tilde{y}_0\right), k=1,\ldots,m$. It follows that these functions must satisfy

$$\frac{\sum_{k=1}^{m} a_k\phi_k\left(\tilde{x}_0,\tilde{y}_0\right)h_k\left(\tilde{x}_0,\tilde{y}_0\right)}{\sum_{i=1}^{m}\left|a_i\phi_i\left(\tilde{x}_0,\tilde{y}_0\right)\right|} = -1,$$

and thus

$$h_k\left(\tilde{x}_0,\tilde{y}_0\right) = -\frac{\overline{a_k\phi_k\left(\tilde{x}_0,\tilde{y}_0\right)}}{\left|a_k\phi_k\left(\tilde{x}_0,\tilde{y}_0\right)\right|}, \qquad k=1,\ldots,m.$$

Equation (8) establishes the result (3).

2.2 Normwise Backward Error

Section 2.1 is extended from a componentwise perturbation in the coefficients to a normwise perturbation in the coefficients.

Definition 5. *The normwise backward error of the approximation* $(\tilde{x}_0,\tilde{y}_0)$*, which may be complex, of the point* (x_0,y_0) *that lies on the curve* $f(x,y)=0$ *is defined as*

$$\eta_n(\tilde{x}_0,\tilde{y}_0) = \min\left\{\varepsilon_n : \sum_{i=1}^{m}\tilde{a}_i\phi_i\left(\tilde{x}_0,\tilde{y}_0\right) = 0 \;\; and \;\; \|\delta a\| \leq \varepsilon_n\,\|a\|\,; \tilde{a} = a + \delta a\right\}.$$

Theorem 2. *The normwise backward error, measured in the 2-norm, of the approximation $(\tilde{x}_0, \tilde{y}_0)$, which may be complex, of the point (x_0, y_0) that lies on the curve $f(x,y) = 0$ is given by*

$$\eta_n(\tilde{x}_0, \tilde{y}_0) = \frac{|f(\tilde{x}_0, \tilde{y}_0)|}{\|\phi(\tilde{x}_0, \tilde{y}_0)\| \, \|a\|}, \qquad \|\cdot\| = \|\cdot\|_2 \,. \tag{9}$$

The perturbations in the coefficients that achieve this backward error are

$$\delta a_i = \frac{f(\tilde{x}_0, \tilde{y}_0) \, g_i}{\|\phi(\tilde{x}_0, \tilde{y}_0)\|}, \qquad i = 1, \ldots, m, \tag{10}$$

where

$$g^T \phi(\tilde{x}_0, \tilde{y}_0) = -\|\phi(\tilde{x}_0, \tilde{y}_0)\| \qquad \textit{and} \qquad \|g\| = 1.$$

Proof. The proof of this theorem follows closely that of Theorem 1. In particular, since the term on the left hand side of (4) is equal to zero, it follows that

$$|f(\tilde{x}_0, \tilde{y}_0)| = \left| \sum_{i=1}^{m} \delta a_i \phi_i(\tilde{x}_0, \tilde{y}_0) \right| \le \left(\frac{\|\delta a\|}{\|a\|} \right) \|a\| \, \|\phi(\tilde{x}_0, \tilde{y}_0)\| \,.$$

Thus

$$\frac{|f(\tilde{x}_0, \tilde{y}_0)|}{\|a\| \, \|\phi(\tilde{x}_0, \tilde{y}_0)\|} \le \frac{\|\delta a\|}{\|a\|} \le \varepsilon_n, \tag{11}$$

and (9) follows.

The perturbations in the coefficients that achieve this backward error must satisfy (6) and (11). Specifically, consider the perturbation vector whose norm is given by

$$\|\delta a\| = \eta_n(\tilde{x}_0, \tilde{y}_0) \, \|a\| = \frac{|f(\tilde{x}_0, \tilde{y}_0)|}{\|\phi(\tilde{x}_0, \tilde{y}_0)\|},$$

from which it follows that

$$\delta a_k = \frac{f(\tilde{x}_0, \tilde{y}_0) \, g_k(\tilde{x}_0, \tilde{y}_0)}{\|\phi(\tilde{x}_0, \tilde{y}_0)\|},$$

where

$$\|g(\tilde{x}_0, \tilde{y}_0)\| = \left(\sum_{k=1}^{m} |g_k(\tilde{x}_0, \tilde{y}_0)|^2 \right)^{\frac{1}{2}} = 1,$$

and the functions $g_k(\tilde{x}_0, \tilde{y}_0)$ are to be determined. The substitution of these expressions for the perturbations δa_k into (6) yields

$$\sum_{k=1}^{m} g_k(\tilde{x}_0, \tilde{y}_0) \phi_k(\tilde{x}_0, \tilde{y}_0) = -\|\phi(\tilde{x}_0, \tilde{y}_0)\| \,,$$

and thus the functions $g_k(\tilde{x}_0, \tilde{y}_0)$ must satisfy

$$g^T(\tilde{x}_0, \tilde{y}_0) \, \phi(\tilde{x}_0, \tilde{y}_0) = -\|\phi(\tilde{x}_0, \tilde{y}_0)\| \qquad \text{and} \qquad \|g(\tilde{x}_0, \tilde{y}_0)\| = 1.$$

This establishes the result (10).

It is assumed, for generality, that the approximation $(\tilde{x}_0, \tilde{y}_0)$ in Theorems 1 and 2 is complex, but practical applications require that it be real. In this circumstance, (2), (9) and (10) are unaltered, but (3) is simplified slightly.

3 Condition Numbers

Expressions for the componentwise and normwise condition numbers of a regular point on the curve (1) are derived in Section 3.1. The expression for the componentwise condition number is proved in [1], but it is reproduced here because reference will be made to particular lines in the proof. It is shown that the extension of the proof to a normwise perturbation follows easily by using the Cauchy-Schwarz inequality. These expressions enable the equations that unite the condition numbers and backward errors to be established, and these equations are developed in Section 3.2.

3.1 Componentwise and Normwise Condition Numbers

Expressions for the componentwise and normwise condition numbers of a regular point on the curve (1) are derived in Theorems 3 and 4 respectively.

Theorem 3. *Let (x_0, y_0) be a regular point P on the curve (1). If each coefficient $a_i, i = 1, \ldots, m$, is perturbed to $a_i + \delta a_i, i = 1, \ldots, m$, where $|\delta a_i| \leq \varepsilon_c |a_i|$, then the componentwise condition number $\kappa_c(x_0, y_0)$ at P is*

$$\kappa_c(x_0, y_0) = \max_{|\delta a_i| \leq \varepsilon_c |a_i|} \frac{|\delta s|}{\varepsilon_c} = \frac{1}{\|\nabla f\|} \sum_{i=1}^{m} |a_i \phi_i(x_0, y_0)|, \tag{12}$$

where δs is the normal component of the perturbation of the point P.

Proof. Let $(x_0 + \delta x_0, y_0 + \delta y_0)$ be a point on the perturbed curve, that is, the curve whose coefficients are $a_i + \delta a_i, i = 1, \ldots, m$,

$$\sum_{i=1}^{m} a_i \phi_i(x_0, y_0) = 0 \qquad \text{and} \qquad \sum_{i=1}^{m} (a_i + \delta a_i) \phi_i(x_0 + \delta x_0, y_0 + \delta y_0) = 0.$$

It therefore follows from (1) that

$$\begin{aligned} f(x_0 + \delta x_0, y_0 + \delta y_0) &= \sum_{i=1}^{m} a_i \phi_i(x_0 + \delta x_0, y_0 + \delta y_0) \\ &= \sum_{i=1}^{m} (a_i + \delta a_i) \phi_i(x_0 + \delta x_0, y_0 + \delta y_0) \\ &\quad - \sum_{i=1}^{m} \delta a_i \phi_i(x_0 + \delta x_0, y_0 + \delta y_0) \\ &= - \sum_{i=1}^{m} \delta a_i \phi_i(x_0 + \delta x_0, y_0 + \delta y_0), \end{aligned} \tag{13}$$

and if the perturbation vector $(\delta x_0, \delta y_0)$ is sufficiently small, then

$$\begin{aligned} f(x_0 + \delta x_0, y_0 + \delta y_0) &= f(x_0, y_0) + (\delta x_0 f_x + \delta y_0 f_y) + \text{ higher order terms} \\ &= \delta x_0 f_x + \delta y_0 f_y + \text{ higher order terms}, \end{aligned} \tag{14}$$

where all derivatives are evaluated at (x_0, y_0). The same approximation implies that the right hand side of (13) can be simplified,

$$\sum_{i=1}^{m} \delta a_i \phi_i \left(x_0 + \delta x_0, y_0 + \delta y_0\right) = \sum_{i=1}^{m} \delta a_i \phi_i \left(x_0, y_0\right),$$

and thus if terms of second and higher degree are sufficiently small such that they can be ignored, (13) becomes

$$\left[\delta x_0 \ \ \delta y_0\right] \cdot \frac{\left[f_x \ \ f_y\right]}{\|\triangledown f\|} = -\frac{1}{\|\triangledown f\|} \sum_{i=1}^{m} \delta a_i \phi_i \left(x_0, y_0\right). \tag{15}$$

The unit normal vector n to the curve $f(x,y) = 0$ at (x,y) is

$$n = \left[n_x \ \ n_y\right] = \frac{\left[f_x \ \ f_y\right]}{\|\triangledown f\|},$$

and thus (15) becomes

$$\left[\delta x_0 \ \ \delta y_0\right] \cdot \left[n_{x_0} \ \ n_{y_0}\right] = -\frac{1}{\|\triangledown f\|} \sum_{i=1}^{m} \delta a_i \phi_i \left(x_0, y_0\right). \tag{16}$$

The term on the left hand side of this equation is the normal component δs of the perturbation,

$$\delta s = \left[\delta x_0 \ \ \delta y_0\right] \cdot \left[n_{x_0} \ \ n_{y_0}\right], \tag{17}$$

and thus

$$|\delta s| \leq \frac{1}{\|\triangledown f\|} \sum_{i=1}^{m} \left|\delta a_i \phi_i \left(x_0, y_0\right)\right| \leq \frac{\varepsilon_c}{\|\triangledown f\|} \sum_{i=1}^{m} \left|a_i \phi_i \left(x_0, y_0\right)\right|.$$

The result (12) follows.

The next theorem extends this expression to a normwise perturbation in the coefficients.

Theorem 4. *Let (x_0, y_0) be a regular point P on the curve (1). If each coefficient $a_i, i = 1, \ldots, m$, is perturbed to $a_i + \delta a_i, i = 1, \ldots, m$, where $\|\delta a\| \leq \varepsilon_n \|a\|$, then the normwise condition number $\kappa_n \left(x_0, y_0\right)$ at P is*

$$\kappa_n \left(x_0, y_0\right) = \max_{\|\delta a\| \leq \varepsilon_n \|a\|} \frac{|\delta s|}{\varepsilon_n} = \frac{1}{\|\triangledown f\|} \|a\| \left\|\phi \left(x_0, y_0\right)\right\|. \tag{18}$$

Proof. The proof is similar to that of Theorem 3, but it differs from (16) onwards. In particular, it follows from this equation that

$$\begin{aligned}
|\delta s| &= \left|\left[\delta x_0 \; \delta y_0\right] \cdot \left[n_{x_0} \; n_{y_0}\right]\right| \\
&= \frac{1}{\|\nabla f\|}\left|\sum_{i=1}^{m} \delta a_i \phi_i\left(x_0, y_0\right)\right| \\
&\leq \frac{1}{\|\nabla f\|}\|\delta a\|\left\|\phi\left(x_0, y_0\right)\right\| \\
&\leq \frac{\varepsilon_n}{\|\nabla f\|}\|a\|\left\|\phi\left(x_0, y_0\right)\right\|,
\end{aligned}$$

and the result (18) follows.

These proofs are easily extended to a singularity of multiplicity two, and the componentwise and normwise condition numbers at this point are [1], respectively,

$$\kappa_c\left(x_0, y_0\right) = \max_{\left|\delta a_i\right| \leq \varepsilon_c\left|a_i\right|} \frac{|\delta s|}{\varepsilon_c} = \frac{1}{\sqrt{\varepsilon_c}}\left(\frac{2\sum_{i=1}^{m}\left|a_i \phi_i\left(x_0, y_0\right)\right|}{\sqrt{f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2}}\right)^{\frac{1}{2}}, \tag{19}$$

and

$$\kappa_n\left(x_0, y_0\right) = \max_{\|\delta a\| \leq \varepsilon_n\|a\|} \frac{|\delta s|}{\varepsilon_n} = \frac{1}{\sqrt{\varepsilon_n}}\left(\frac{2\|a\|\left\|\phi\left(x_0, y_0\right)\right\|}{\sqrt{f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2}}\right)^{\frac{1}{2}}, \tag{20}$$

where

$$|\delta s|^2 = \left|\delta x_0^2 n_{x_0 x_0} + \sqrt{2} \delta x_0 \delta y_0 n_{x_0 y_0} + \delta y_0^2 n_{y_0 y_0}\right|, \tag{21}$$

and

$$\begin{aligned}
n_{xx} &= \frac{f_{xx}}{\sqrt{f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2}}, \\
n_{xy} &= \frac{\sqrt{2} f_{xy}}{\sqrt{f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2}}, \\
n_{yy} &= \frac{f_{yy}}{\sqrt{f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2}}.
\end{aligned}$$

3.2 Backward Errors and Condition Numbers

It is shown in this section that the backward errors and condition numbers at a point on the curve (1) are related by simple expressions, the exact form of which depends on the multiplicity r of the point. Equations (12) and (18) show that, respectively, only the normal component of the perturbation is defined in the expressions for the componentwise and normwise condition numbers at a regular point, and the tangential component of the perturbation is not considered in these expressions.

The product of the componentwise condition number (12) at a regular point and componentwise backward error (2) is

$$\frac{1}{\|\nabla f\|}\sum_{i=1}^{m}|a_i\phi_i(x_0,y_0)|\,\frac{|f(\tilde{x}_0,\tilde{y}_0)|}{\sum_{i=1}^{m}|a_i\phi_i(\tilde{x}_0,\tilde{y}_0)|}, \tag{22}$$

where, to first order,

$$f(\tilde{x}_0,\tilde{y}_0)=f(x_0,y_0)+(\delta x_0 f_x+\delta y_0 f_y)=\frac{(\delta x_0 f_x+\delta y_0 f_y)}{\|\nabla f\|}\|\nabla f\|=\delta s\,\|\nabla f\|,$$

and δs, the normal component of the forward error, is defined in (17). It therefore follows that the product (22) simplifies to

$$|\delta s|\,\frac{\sum_{i=1}^{m}|a_i\phi_i(x_0,y_0)|}{\sum_{i=1}^{m}|a_i\phi_i(\tilde{x}_0,\tilde{y}_0)|}=|\delta s|,$$

to first order, and thus the product of the componentwise condition number and componentwise backward error at a regular point is equal to the normal component of the forward error. It is easily verified that the same result applies to the normwise condition number and normwise backward error, and thus the equations

$$|\delta s|=\kappa_c(x_0,y_0)\eta_c(\tilde{x}_0,\tilde{y}_0), \tag{23}$$

$$|\delta s|=\kappa_n(x_0,y_0)\eta_n(\tilde{x}_0,\tilde{y}_0), \tag{24}$$

are satisfied at a regular point.

The equivalent results for a singularity of arbitrary multiplicity are obtained by extending (12) and (18), and (19) and (20), to a point of this class, and it is noted that the backward errors (2) and (9) are independent of the multiplicity of the point. This generalisation leads to the expressions

$$|\delta s|=\kappa_c(x_0,y_0)\left(\frac{\eta_c(\tilde{x}_0,\tilde{y}_0)}{\varepsilon_c}\right)^{\frac{1}{r}}\varepsilon_c,$$

$$|\delta s|=\kappa_n(x_0,y_0)\left(\frac{\eta_n(\tilde{x}_0,\tilde{y}_0)}{\varepsilon_n}\right)^{\frac{1}{r}}\varepsilon_n,$$

where δs is equal to the extension of (17) from a singularity of multiplicity one, and (21) from a singularity of multiplicity two, to a singularity of multiplicity r. Equations (23) and (24), which are appropriate at a regular point, are obtained by setting $r = 1$.

4 Condition Numbers of a Nearly Singular Point

Examination of the proofs of the componentwise and normwise backward errors, (2) and (9) respectively, shows that the expressions for these errors are exact because approximations are not used in their derivation. By contrast, the expressions for the componentwise and normwise condition numbers at a regular point, (12) and (18) respectively, require that only the lowest order terms be used, which implies that they are not valid for all perturbations. It follows, therefore, that although the expressions for the backward errors are valid at all points on an algebraic curve, there exist restrictions on the class of point for which the condition numbers (12) and (18) are valid. This phenomenon has been considered for univariate polynomials [4], where it is shown that this restriction to the lowest order terms in a Taylor series places bounds on the validity of the condition numbers of a root that is near another root. This section extends these results from univariate polynomials to bivariate polynomials.

It follows from (13) and (14) that only first order terms are considered, and thus the perturbations δx_0 and δy_0 must satisfy

$$\frac{1}{2}\left|\begin{bmatrix}\delta x_0 \; \delta y_0\end{bmatrix}\begin{bmatrix} f_{xx} \; f_{xy} \\ f_{yx} \; f_{yy}\end{bmatrix}\begin{bmatrix}\delta x_0 \\ \delta y_0\end{bmatrix}\right| \ll \left|\delta x_0 f_x + \delta y_0 f_y\right|, \tag{25}$$

and

$$\left|\delta x_0 \frac{\partial \phi_i(x,y)}{\partial x} + \delta y_0 \frac{\partial \phi_i(x,y)}{\partial y}\right| \ll \left|\phi_i(x_0,y_0)\right|, \qquad i=1,\ldots,m, \tag{26}$$

for all perturbations δa_i, where the derivatives are evaluated at (x_0, y_0). By contrast, a singular point of multiplicity 2 requires that $f_x = f_y = 0$, and thus

$$\delta x_0 f_x + \delta y_0 f_y = 0, \tag{27}$$

for all δx_0 and δy_0 at this point. Equations (25), (26) and (27) show that the values of x_0, y_0, δx_0 and δy_0 that satisfy

$$\left|\delta x_0 f_x + \delta y_0 f_y\right| \approx \frac{1}{2}\left|\begin{bmatrix}\delta x_0 \; \delta y_0\end{bmatrix}\begin{bmatrix} f_{xx} \; f_{xy} \\ f_{yx} \; f_{yy}\end{bmatrix}\begin{bmatrix}\delta x_0 \\ \delta y_0\end{bmatrix}\right|, \tag{28}$$

$$\left|\phi_i(x_0,y_0)\right| \approx \left|\delta x_0 \frac{\partial \phi_i(x,y)}{\partial x} + \delta y_0 \frac{\partial \phi_i(x,y)}{\partial y}\right|, \qquad i=1,\ldots,m, \tag{29}$$

$$f_x, f_y \neq 0, \tag{30}$$

do not satisfy the conditions that are required for the derivation of condition numbers at regular and singular points. It follows, therefore, that if x_0, y_0, δx_0 and δy_0 satisfy (28), (29) and (30), Theorems 3 and 4 are not valid, and refined perturbation methods are required. This is illustrated in the following example, which also shows that a disadvantage of the condition numbers (12) and (18) – their restriction to the normal component of the perturbation – can be removed by including the tangential component of the perturbation in the definition of the condition number.

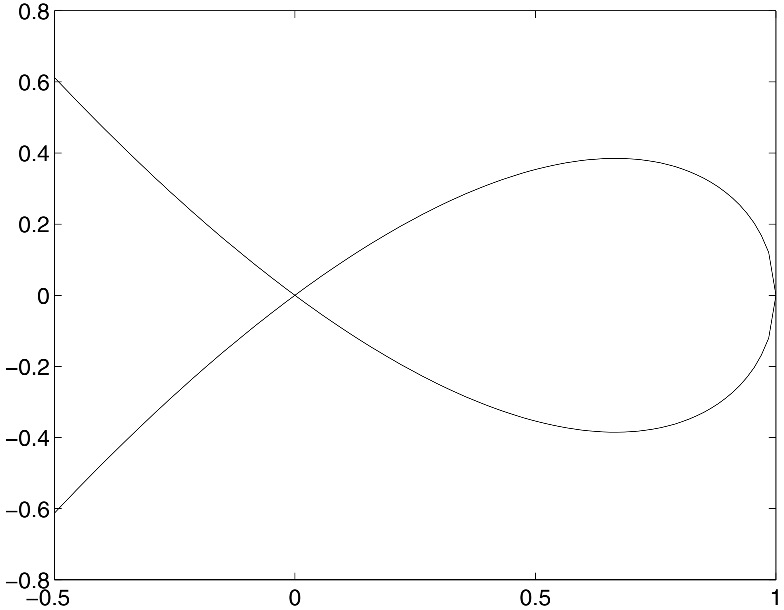


Fig. 2. The curve $x^3 - x^2 + y^2 = 0$

Example 1. Consider the curve shown in Figure 2,

$$f(x,y) = x^3 - x^2 + y^2 = 0. \tag{31}$$

The curve has a singularity of multiplicity 2 at the origin, and the non-zero partial derivatives of $f(x,y)$ are

$$f_x = 3x^2 - 2x, \quad f_y = 2y, \quad f_{xx} = 6x - 2, \quad f_{xy} = 0, \quad f_{yy} = 2, \quad f_{xxx} = 6.$$

All points on the curve have coordinates

$$(x_0, y_0) = \left(x_0, \pm x_0\sqrt{1 - x_0}\right), \qquad x_0 \leq 1,$$

and the derivation of expressions for the condition numbers at a point on the curve requires a Taylor expansion of $f(x,y)$, as seen in (14), and thus

$$\begin{aligned} f\left(x_0 + \delta x_0, x_0\sqrt{1-x_0} + \delta y_0\right) &= \left(x_0\left(3x_0 - 2\right)\delta x_0 + 2x_0\sqrt{1-x_0}\delta y_0\right) \\ &\quad + \left(\left(3x_0 - 1\right)\delta x_0^2 + \delta y_0^2\right) + \delta x_0^3, \\ f\left(x_0 + \delta x_0, -x_0\sqrt{1-x_0} + \delta y_0\right) &= \left(x_0\left(3x_0 - 2\right)\delta x_0 - 2x_0\sqrt{1-x_0}\delta y_0\right) \\ &\quad + \left(\left(3x_0 - 1\right)\delta x_0^2 + \delta y_0^2\right) + \delta x_0^3. \end{aligned} \tag{32}$$

Three points are considered, and they are distinguished by their distance from the singularity. These points and their coordinates are:

Point P_1**:** $(x_0, y_0) = \left(x_0, x_0\sqrt{1-x_0}\right) = (1, 0)$
Point P_2**:** $(x_0, y_0) = \left(x_0, x_0\sqrt{1-x_0}\right) \approx (10^{-6}, 10^{-6})$
Point P_3**:** $(x_0, y_0) = \left(x_0, x_0\sqrt{1-x_0}\right) = (0, 0)$

The point P_1 is regular and remote from the singularity, and this differs from P_2, which is regular but near the singularity. The point P_3 is, however, singular because $f_x = f_y = 0$ at this point.

It is assumed that $\varepsilon_c = 10^{-6}$, and only real perturbations are considered. Equations (25) and (26) specify the bounds on δx_0 and δy_0 in order that a first order Taylor series be appropriate, and (12) enables the area that is defined by these bounds to be reduced significantly.

Each point is now considered.

Point P_1**:** Equation (25) shows that a first order Taylor series is valid at P_1 if

$$\left|2\delta x_0^2 + \delta y_0^2\right| \ll |\delta x_0|,$$

and if it is assumed that a reduction of one order of magnitude is sufficient to replace the $\ll$ by $\leq$, this inequality becomes

$$2\delta x_0^2 + \delta y_0^2 \leq \frac{1}{10}|\delta x_0|. \tag{33}$$

Also, (26) shows that the restriction to first order analysis requires that the perturbations satisfy

$$|\delta x_0| \ll \frac{1}{3} \qquad \text{and} \qquad |\delta x_0| \ll \frac{1}{2},$$

or, using the assumption stated above,

$$|\delta x_0| \leq \frac{1}{30}. \tag{34}$$

It follows that if the perturbations in the coefficients are sufficiently small such that only first order terms are adequate, then the point P_1 is displaced to a point that lies in the intersection of the regions that are defined by the inequalities (33) and (34). It is now shown that the componentwise relative error allows this region to be reduced considerably.

The partial derivatives of $f(x, y)$ at this point are $f_x = 1$ and $f_y = 0$, and the unit normal vector at this point is

$$n_0 = \begin{bmatrix} n_{x_0} & n_{y_0} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

The componentwise condition number at this point is, from (12), $\kappa_c(x_0, y_0) = 2$, and thus

$$\max\frac{|\delta s|}{\varepsilon_c} = \frac{1}{\varepsilon_c}\max\left|\begin{bmatrix} \delta x_0 & \delta y_0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \end{bmatrix}\right| = 2,$$

or

$$|\delta x_0| \leq 2\varepsilon_c. \tag{35}$$

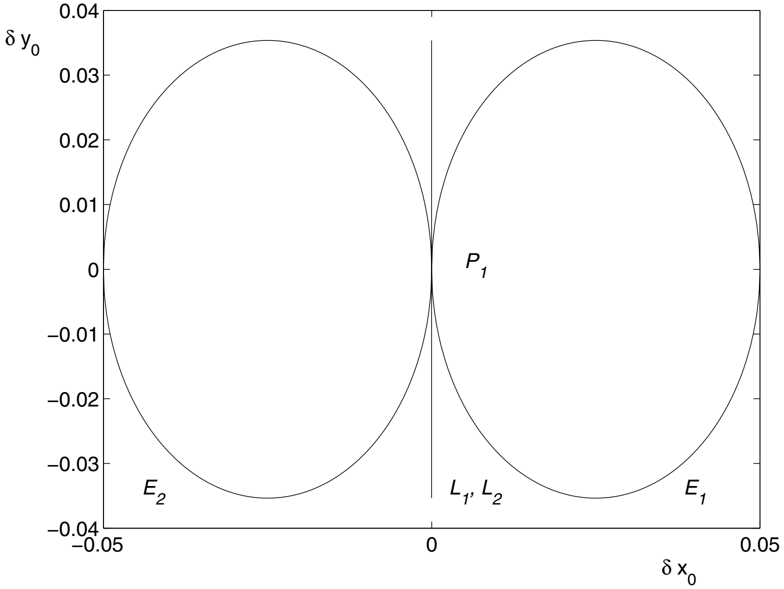


Fig. 3. The ellipses E_1 and E_2, which are defined in (36) and (37), respectively, and the lines L_1 and L_2, which are defined in (38)

This inequality defines an infinite strip, parallel to the δy_0-axis, within which the perturbed point must lie, assuming that the perturbations are of first order. The satisfaction of the inequality (35) necessarily implies that the inequality (34) is satisfied, from which it follows that (34) need not be considered.

Equations (33) and (35) show that the boundaries of the region within which the point P_1 is displaced are defined by the ellipses E_1 and E_2,

$$E_1 : \left(\delta x_0 - \frac{1}{40}\right)^2 + \frac{\delta y_0^2}{2} - \frac{1}{1600} = 0, \tag{36}$$

$$E_2 : \left(\delta x_0 + \frac{1}{40}\right)^2 + \frac{\delta y_0^2}{2} - \frac{1}{1600} = 0, \tag{37}$$

which have a common tangent at $\delta x_0 = \delta y_0 = 0$, and the lines L_1 and L_2,

$$L_1 : \ \delta x_0 - 2\varepsilon_c = 0; \qquad L_2 : \ \delta x_0 + 2\varepsilon_c = 0. \tag{38}$$

The ellipses E_1 and E_2, and the lines L_1 and L_2, are shown in Figure 3, and it is noted that the lines appear to be coincident because of the scale of the figure. The intersection region is the area shown in black in Figure 4, which is a schematic diagram of the region bounded by E_1, E_2, L_1 and L_2. The point P_1, which is not marked in this figure, is the origin of the local coordinate system $(\delta x_0, \delta y_0)$.

The lines and ellipses intersect at the points A, B, C and D, whose coordinates are, respectively,

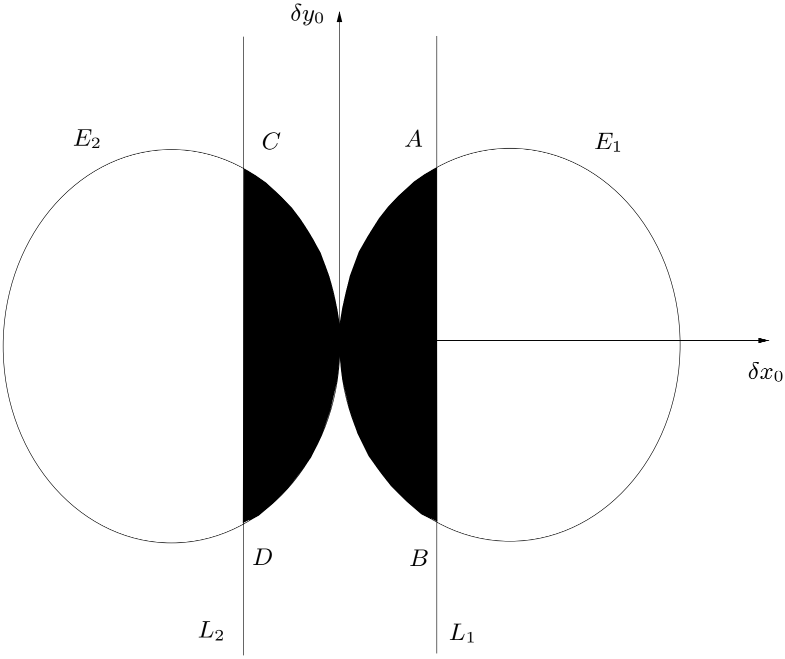


Fig. 4. A schematic diagram of the region within which the point P_1 is displaced. The ellipses E_1 and E_2 intersect tangentially at the point P_1, the origin of the coordinate system $(\delta x_0, \delta y_0)$. This point is not marked

$$(\delta x_A, \delta y_A) = \left(2\varepsilon_c, \sqrt{\tfrac{\varepsilon_c}{5}}\right), \quad (\delta x_B, \delta y_B) = \left(2\varepsilon_c, -\sqrt{\tfrac{\varepsilon_c}{5}}\right),$$

$$(\delta x_C, \delta y_C) = \left(-2\varepsilon_c, \sqrt{\tfrac{\varepsilon_c}{5}}\right), \ (\delta x_D, \delta y_D) = \left(-2\varepsilon_c, -\sqrt{\tfrac{\varepsilon_c}{5}}\right),$$

where terms of order ε_c^2 are neglected because $\varepsilon_c \ll 1$. It is clear that all points in the region shown in black in Figure 4 satisfy (34). The condition number $\gamma_c(x_0, y_0)$ at P_1 can be defined as the maximum value of the magnitude of the perturbation,

$$\gamma_c\left(x_0, y_0\right) = \frac{1}{\varepsilon_c} \max_{i=A,B,C,D} \left\{ \sqrt{\delta x_i^2 + \delta y_i^2} \right\} = \sqrt{4 + \frac{1}{5\varepsilon_c}} = 447.2,$$

and thus the ratio of this condition number to the condition number $\kappa_c\left(x_0, y_0\right)$ is

$$\frac{\gamma_c\left(x_0, y_0\right)}{\kappa_c\left(x_0, y_0\right)} = \frac{447.2}{2} = 223.6.$$

This large ratio implies that $\kappa_c\left(x_0, y_0\right)$ is a poor measure of the numerical condition of the curve at P_1. This arises because the unit vectors from P_1 to the vertices A, B, C and D, which are the points of maximum displacement, are

$$\frac{\left[\, \pm 2\varepsilon_c \ \pm\sqrt{\frac{\varepsilon_c}{5}}\, \right]}{\sqrt{4\varepsilon_c^2 + \frac{\varepsilon_c}{5}}},$$

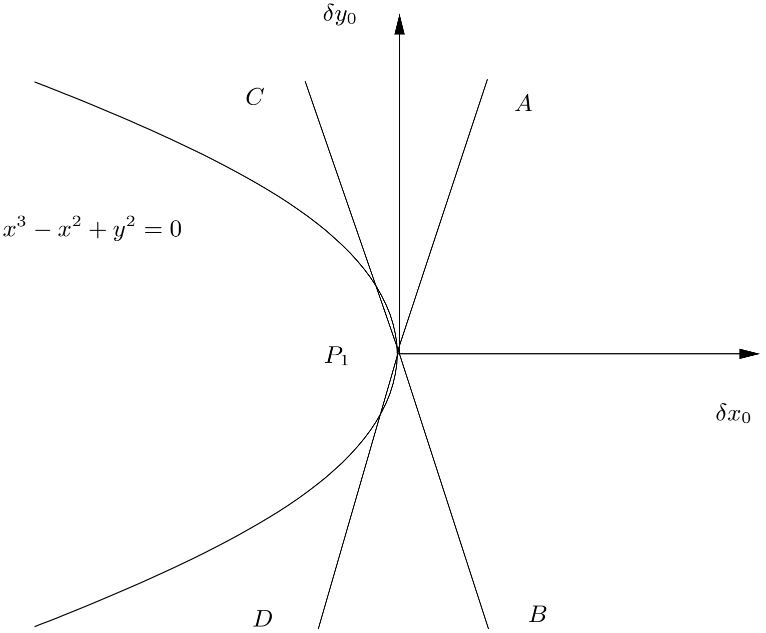


Fig. 5. The vectors P_1A, P_1B, P_1C and P_1D

and thus the angles between these vectors and the unit normal vector to the curve at this point are

$$\cos\theta = \frac{\left[\pm 2\varepsilon_c \ \ \pm\sqrt{\frac{\varepsilon_c}{5}}\right]}{\sqrt{4\varepsilon_c^2 + \frac{\varepsilon_c}{5}}} \cdot \left[1 \ \ 0\right] \approx \pm 2\sqrt{5\varepsilon_c}.$$

The vectors P_1A, P_1B, P_1C and P_1D are shown in Figure 5, and it is seen that they are almost orthogonal to the unit normal vector at P_1, which is parallel to the δx_0-axis. It follows that the maximum value of the normal component of the perturbation of P_1 is insignificant compared to the maximum value of its tangential component.

Finally, it is noted that the magnitude of the last term on the right hand side of (32) is $\left|\delta x_0^3\right|$. Its omission from the calculations is justified because its maximum value is $8\varepsilon_c^3 = 8 \times 10^{-18}$, which is insignificant compared to the term of first order.

Point P_2: The analysis for this point follows closely that for the point P_1. Thus (25) and (26) define the region in which the perturbed point must lie if first order perturbation analysis is valid, and the expression for $\kappa_c(x_0, y_0)$ enables the area of this region to be considerably reduced.

It follows from (25) that

$$\left|(3 \times 10^{-6} - 1)\delta x_0^2 + \delta y_0^2\right| \ll \left|(3 \times 10^{-12} - 2 \times 10^{-6})\delta x_0 + 2 \times 10^{-6}\delta y_0\right|, \quad (39)$$

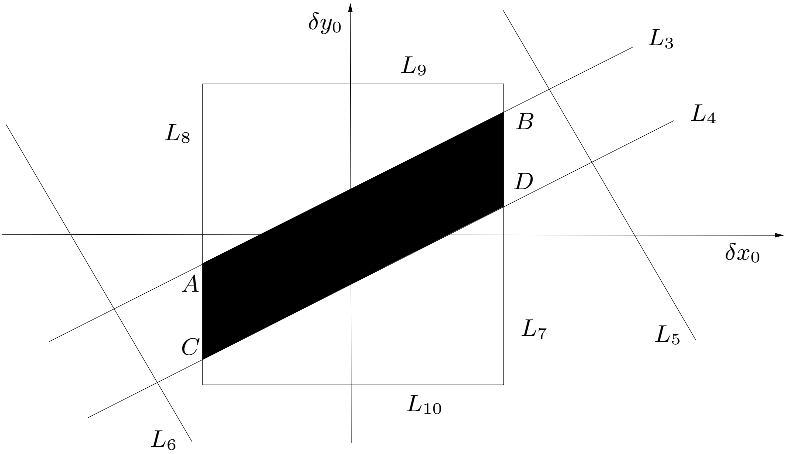


Fig. 6. A schematic diagram of the region of intersection that is defined by the inequalities (40), (41) and (42)

which reduces to

$$|\delta x_0 + \delta y_0| \leq 2 \times 10^{-7}, \tag{40}$$

if $\delta x_0 \neq \delta y_0$. Also, it follows from (26) that

$$|\delta x_0| \leq \frac{10^{-7}}{3} \qquad \text{and} \qquad |\delta y_0| \leq \frac{10^{-7}}{2}, \tag{41}$$

and the definition of $\kappa_c(x_0, y_0)$ shows that

$$|-\delta x_0 + \delta y_0| \leq 10^{-6}\varepsilon_c, \tag{42}$$

since the unit normal vector to the curve at P_2 is

$$n_0 = \begin{bmatrix} n_{x_0} & n_{y_0} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{43}$$

If $\delta x_0 \neq \delta y_0$, P_2 is displaced to a point that lies in the intersection of the regions that are defined by the inequalities (40), (41) and (42). These inequalities define eight lines,

$$\begin{aligned} -\delta x_0 + \delta y_0 &= \pm 10^{-6}\varepsilon_c && \text{(Lines } L_3,\ L_4\text{)} \\ \delta x_0 + \delta y_0 &= \pm 2 \times 10^{-7} && \text{(Lines } L_5,\ L_6\text{)} \\ \delta x_0 \qquad &= \pm\tfrac{10^{-7}}{3} && \text{(Lines } L_7,\ L_8\text{)} \\ \delta y_0 &= \pm\tfrac{10^{-7}}{2} && \text{(Lines } L_9,\ L_{10}\text{)}, \end{aligned}$$

which are shown in Figure 6. The coordinates of the points A, B, C and D of the region shown in black are

$$\begin{aligned} (\delta x_A, \delta y_A) &= \left(-\tfrac{10^{-7}}{3}, -\tfrac{10^{-7}}{3} + 10^{-6}\varepsilon_c\right), \ (\delta x_B, \delta y_B) = \left(\tfrac{10^{-7}}{3}, \tfrac{10^{-7}}{3} + 10^{-6}\varepsilon_c\right), \\ (\delta x_C, \delta y_C) &= \left(-\tfrac{10^{-7}}{3}, -\tfrac{10^{-7}}{3} - 10^{-6}\varepsilon_c\right), \ (\delta x_D, \delta y_D) = \left(\tfrac{10^{-7}}{3}, \tfrac{10^{-7}}{3} - 10^{-6}\varepsilon_c\right), \end{aligned}$$

and thus this region is aligned at 45 degrees to the δx_0-axis. The major axis of this region is

$$r = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix},$$

and this is orthogonal to the unit normal vector to the curve, which is given in (43).

Consider now the situation that occurs when $\delta x_0 = \delta y_0$. It is readily verified from (39) that this condition requires that

$$\left|\delta x_0\right|, \left|\delta y_0\right| \leq 10^{-7},$$

and all points that satisfy these inequalities lie in the region $ABDC$ in Figure 6. Since the maximum displacement occurs at the vertices A, B, C and D, the condition number $\gamma_c\left(x_0, y_0\right)$ is

$$\gamma_c\left(x_0, y_0\right) = \frac{1}{\varepsilon_c} \max_{i=A,B,C,D} \left\{ \sqrt{\delta x_i^2 + \delta y_i^2} \right\} = 4.714 \times 10^{-2}.$$

Furthermore, since $\kappa_c\left(x_0, y_0\right) = 7.071 \times 10^{-7}$, it follows that the ratio of the condition numbers $\gamma_c\left(x_0, y_0\right)$ and $\kappa_c\left(x_0, y_0\right)$ is

$$\frac{\gamma_c\left(x_0, y_0\right)}{\kappa_c\left(x_0, y_0\right)} = \frac{4.714 \times 10^{-2}}{7.071 \times 10^{-7}} = 6.67 \times 10^4,$$

which is large. The reason is the same as that for the point P_1 – the major axis of the region in which the perturbed point lies is almost orthogonal to the normal vector to the curve.

Point P_3: It is easily verified that the condition numbers (19) and (20) are equal to zero at this point. It is interesting to note, however, that (12) and (18), which are the condition numbers at a regular point and not a singular point, yield the indeterminate form 0/0. L'hôpital's rule is required for their evaluation, and this shows that both these condition numbers are also equal to zero, and not infinity, as would be expected. The value of zero arises because P_3 lies on the curve $f(x, y) = 0$ and its perturbed form

$$(1 + \delta a_1)x^3 - (1 + \delta a_2)x^2 + (1 + \delta a_3)y^2 = 0,$$

for all $\delta a_i, i = 1, 2, 3$.

The point P_2 is regular but near the singularity, and the limiting value of the condition numbers $\kappa_c(x_0, y_0)$ and $\gamma_c(x_0, y_0)$, and their ratio $\gamma_c(x_0, y_0)/\kappa_c(x_0, y_0)$, as the singularity is approached can be determined by noting that if $0 < \epsilon \ll 1$, then the point (ϵ, ϵ) is in the first quadrant and lies on the curve (31). As the singularity is approached, the condition number (12) decreases linearly to zero,

$$\lim_{\epsilon \longrightarrow 0} \kappa_c(\epsilon, \epsilon) = \lim_{\epsilon \longrightarrow 0} \frac{\epsilon^3 + 2\epsilon^2}{\sqrt{\left(3\epsilon^2 - 2\epsilon\right)^2 + (2\epsilon)^2}} = \frac{\epsilon}{\sqrt{2}}. \qquad (44)$$

It follows from (25) that since $\epsilon \ll 1$, the inequality

$$\left|-\delta x_0^2 + \delta y_0^2\right| \ll \epsilon \left|-2\delta x_0 + 2\delta y_0\right|,$$

must be satisfied. If $\delta x_0 \neq \delta y_0$, it reduces to

$$\left|\delta x_0 + \delta y_0\right| \leq \frac{\epsilon}{5}, \tag{45}$$

and if $\delta x_0 = \delta y_0$, (25) yields

$$\left|\delta x_0\right|, \left|\delta y_0\right| \leq \frac{\epsilon}{10}. \tag{46}$$

Equation (26) shows that the perturbations δx_0 and δy_0 must satisfy

$$\left|\delta x_0\right| \leq \frac{\epsilon}{30} \qquad \text{and} \qquad \left|\delta y_0\right| \leq \frac{\epsilon}{20}, \tag{47}$$

for first order perturbation analysis to be valid, and it is clear that the satisfaction of these inequalities necessarily implies that (46) is satisfied. Also, it follows from (44) and the definition of $\kappa_c\left(x_0, y_0\right)$ that since ϵ is small,

$$\left|-\delta x_0 + \delta y_0\right| \leq \epsilon \varepsilon_c, \tag{48}$$

and thus the region in which the perturbed point lies if first order perturbation analysis is used is equal to the intersection of the regions that are defined by the inequalities (45), (47) and (48). This region is identical to that shown in Figure 6, and the coordinates of the points A, B, C and D are

$$\begin{array}{ll}\left(\delta x_A, \delta y_A\right) = \left(-\frac{\epsilon}{30}, -\frac{\epsilon}{30} + \epsilon\varepsilon_c\right), & \left(\delta x_B, \delta y_B\right) = \left(\frac{\epsilon}{30}, \frac{\epsilon}{30} + \epsilon\varepsilon_c\right), \\ \left(\delta x_C, \delta y_C\right) = \left(-\frac{\epsilon}{30}, -\frac{\epsilon}{30} - \epsilon\varepsilon_c\right), & \left(\delta x_D, \delta y_D\right) = \left(\frac{\epsilon}{30}, \frac{\epsilon}{30} - \epsilon\varepsilon_c\right).\end{array}$$

It follows that the condition number of a point sufficiently near the singularity such that only first order terms are appropriate is

$$\gamma_c\left(\epsilon, \epsilon\right) = \frac{1}{\varepsilon_c} \max_{i=A,B,C,D} \left\{\sqrt{\delta x_i^2 + \delta y_i^2}\right\} = \left(\frac{\sqrt{2}}{30}\frac{\epsilon}{\varepsilon_c}\right),$$

and thus the ratio of the condition numbers $\gamma_c\left(\epsilon, \epsilon\right)$ and $\kappa_c\left(\epsilon, \epsilon\right)$ as $\epsilon \rightarrow 0$ is

$$\lim_{\epsilon \rightarrow 0} \frac{\gamma_c\left(\epsilon, \epsilon\right)}{\kappa_c\left(\epsilon, \epsilon\right)} = \lim_{\epsilon \rightarrow 0}\left(\frac{\sqrt{2}}{30}\frac{\epsilon}{\varepsilon_c}\right)\left(\frac{\sqrt{2}}{\epsilon}\right) = \frac{1}{15\varepsilon_c} = 6.67 \times 10^4.$$

Also, the maximum possible displacement of the point whose coordinates are (ϵ, ϵ) is (to lowest order) $(\epsilon/30, \epsilon/30)$, that is,

$$(\epsilon, \epsilon) \rightarrow \left(\epsilon \pm \epsilon/30, \epsilon \pm \epsilon/30\right).$$

It is clear that this result is in agreement with the condition number at P_2.

These calculations were repeated for two other points, and the results are summarised in Table 1. Several conclusions can be drawn from this table and the preceding analysis:

Table 1. A summary of the numerical results for five points on the curve (31)

Coordinates (x_0, y_0)	$\kappa_c\,(x_0, y_0)$	$\gamma_c\,(x_0, y_0)$	$\frac{\gamma_c(x_0,y_0)}{\kappa_c(x_0,y_0)}$	$\max_{i=A,B,C,D} \sqrt{\delta x_i^2 + \delta y_i^2}$
$(1, 0)$	2.00	447.2	224	4.47×10^{-4}
$(0.64, 0.384)$	1.064	298.4	280	2.98×10^{-4}
$(0.64 \times 10^{-3}, 0.64 \times 10^{-3})$	4.528×10^{-4}	30.12	6.65×10^4	3.01×10^{-5}
$(10^{-6}, 10^{-6})$	7.071×10^{-7}	4.714×10^{-2}	6.67×10^4	4.71×10^{-8}
$(0, 0)$	0	0	6.67×10^4	0

1. $\kappa_c(x_0, y_0)$ is smaller than $\gamma_c(x_0, y_0)$, possibly by several orders of magnitude, because $\kappa_c(x_0, y_0)$ considers only the maximum value of the normal component of the perturbation, but $\gamma_c(x_0, y_0)$ considers the maximum value of the perturbation in all directions.
2. The value of
$$\max_{i=A,B,C,D} \left\{ \sqrt{\delta x_i^2 + \delta y_i^2} \right\},$$
decreases as the distance to the singularity decreases, and thus the restriction to the lowest order terms in a Taylor series places a tight bound on the maximum allowable perturbation.
3. The ratio $\gamma_c\,(x_0, y_0)\,/\kappa_c\,(x_0, y_0)$ increases to a maximum value of 6.67×10^4, which is attained as the singularity is approached.
4. $\kappa_c(x_0, y_0)$ is independent ε_c, but $\gamma_c(x_0, y_0)$ is a function of ε_c.

5 Summary and Discussion

Expressions for the componentwise and normwise backward errors at a point on an algebraic curve, and the formulae that relate them to the condition numbers, have been established. It has been shown that the expressions for the backward errors are valid at all points on a curve, but that there are restrictions on the validity of the condition numbers at points near a singularity.

An example was used to motivate an improved expression $\gamma_c(x_0, y_0)$ for the numerical condition of a point on an algebraic curve, and it was shown that it may differ significantly from the established condition number $\kappa_c(x_0, y_0)$. This difference arises because $\kappa_c(x_0, y_0)$ quantifies the maximum value of the normal component of the perturbation, but $\gamma_c(x_0, y_0)$ is a measure of the maximum value of the perturbation in all directions. The revised condition number $\gamma_c(x_0, y_0)$ is, however, computationally more involved than $\kappa_c(x_0, y_0)$.

The analysis in this paper has been relatively simple, and some of the restrictions that are imposed by traditional methods have been discussed and illustrated by an example. There are, however, more issues that must be addressed for a complete discussion of the numerical condition of a regular point that is near a singular point:

– It was assumed in the paper that the inequality $a \ll b$ can be replaced by $a \leq b/k$ where $k = 10$. This choice of k was arbitrary, and other values of k, for example, $k = 20, 50, 100$, must be investigated in order to determine the dependence of $\gamma_c(x_0, y_0)$ on this constant.

- It was assumed that $\varepsilon_c = 10^{-6}$, but other values must be considered because $\gamma_c(x_0, y_0)$ is a function of this constant. It is expected that this dependence increases as the singularity is approached.
- A relatively simple curve was used to demonstrate some of the concepts that were discussed. A more detailed study requires that a curve of arbitrary degree be considered, and this should include the determination of the region in which a perturbed point lies.

References

1. R. T. Farouki and V. T. Rajan. On the numerical condition of algebraic curves and surfaces 1. Implicit equations. *Computer Aided Geometric Design*, 5:215–252, 1988.
2. N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, USA, 2002.
3. R. J. Walker. *Algebraic Curves.* Dover, New York, USA, 1978.
4. J. R. Winkler. Condition numbers of a nearly singular simple root of a polynomial. *Applied Numerical Mathematics*, 38:275–285, 2001.

Approximate Rational Parameterization of Implicitly Defined Surfaces

Elmar Wurm[1], Bert Jüttler[1], and Myung-Soo Kim[2]

[1] Institute of Applied Geometry, Johannes Kepler University, Linz, Austria
{elmar.wurm, bert.juettler}@jku.at
http://www.ag.jku.at/

[2] Department of Computer Science, Seoul National University, Korea
mskim@cse.snu.ac.kr
http://3map.snu.ac.kr

Abstract. We present a method for approximate rational parameterization of algebraic surfaces of arbitrary degree and genus (or more general implicitly defined surfaces), based on numerical optimization techniques. The method computes patches of maximal size on these surfaces subject to certain quality constraints. It can be used to generate local low degree approximations and rational approximations of non-parameterisable surfaces.

1 Introduction

In geometric modelling and computer aided design, various different representations for curves and surfaces exist, such as implicitly defined curves and surfaces, parametric representations by (piecewise) rational functions, procedurally defined surfaces, or triangular meshes. The duality of implicit and parametric representations makes each of them especially well suited for certain applications, cf. [3].

Parametric descriptions are suitable for fast generation of point meshes, fast visualization and interactive modeling. On the other hand, the use of implicitly defined surfaces provides simple criteria to decide whether points are located on, inside or outside a surface. These surfaces support simple techniques to define blend surfaces between objects, and they can easily be intersected with lines. Moreover the class of algebraic surfaces is closed under geometric operations such as intersection and offsetting (although this is a more theoretical advantage, since the resulting degrees are rather high).

Most computational applications yield optimal performance for one particular representation. Regardless, there exist some areas where it is crucial that both descriptions are available. An example is surface-surface intersection. Ideally, one of the surfaces should be given in implicit form, and the other in parametric form. In the case of the detection of self–intersections, both representations of the same surface should be available.

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 434–447, 2005.

This paper is devoted to the problem of converting an algebraic surface (or, more generally, an implicitly defined surface) to a (rational) parametric representation, which we shortly refer to as parameterization.

Several exact methods based on algebraic techniques are known. Most of them are constrained to special curves and surfaces (e.g., of low degree) [1, 2, 5, 12, 18]. Algorithms for solving the general parameterization problem are available [17].

Clearly, the algebraic techniques can be used only if an exact rational parameterization exists. In the surface case, both the arithmetic genus and the second plurigenus have to be equal to 0.

Alternatively, one may use approximate methods, which should be able to generate patches on any input surface. Also, we expect them to be computationally less expensive than exact methods.

In [4], a combination of algebraic and numerical techniques is used to construct G^1 spline approximations of algebraic surfaces. The algorithm starts with the computation of the singular points and curves. Later, Padé approximation and Taylor expansion are used to generate an approximation. The resulting surface maintains differential properties of the input surface and preserves the singularities.

The numerical parameterization method investigated in [8] uses the so called normal-form of a curve/surface. The output is a procedurally defined parameterization, i.e., an algorithm that maps a parameter (pair) to a point on the curve or surface $\mathcal{C}$. First a parametric patch relatively close to $\mathcal{C}$ is generated and then a parameter (pair) can be mapped to the according footpoint on $\mathcal{C}$. Note that $\mathcal{C}$ needs to be free of singularities in the area of interest.

In the remainder of this paper we present a numerical method for generating an approximate rational parameterization of an algebraic surface. We combine nonlinear minimization techniques with a region growing approach, in order to obtain good initial solutions for the nonlinear minimization.

The paper is organized as follows. Section 2 describes the objective function. Its main ingredient is a distance functional, measuring the deviation of the rational surface patch from the given algebraic surface. Section 3 discusses the actual minimization procedure and the region growing process. Starting with a small initial patch we alternate minimization and extrapolation steps to obtain an approximation of maximal size subject to certain quality criteria. Various examples are described in section 4. Finally we conclude this paper.

2 Rational Parameterization as Nonlinear Optimization

A parameterization of a given surface is generated by computing a (possibly local) minimizer of an objective function of the form

$$S = I + \omega_J J + \omega_L L + \omega_R R + \omega_E E \tag{1}$$

among all rational surface patches of a given degree. The next section describes the space of rational patches, while the different contributions to the objective function will be explained in subsections 2.2–2.5.

2.1 Preliminaries

Consider an algebraic surface $\mathcal{F}$ of degree d. It consists of all points satisfying $F(x,y,z) = 0$, where F is a polynomial of total degree d in x, y and z with coefficients g_{ijk},

$$F(x,y,z) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} \sum_{k=0}^{d-i-j} x^i y^j z^k g_{ijk}. \tag{2}$$

For reasons of numerical stability, F should be represented in Bernstein–Bézier form. The techniques described below can be applied to any implicitly defined surface, provided that the function F is C^2.

We generate a rational surface patch $\mathcal{P}$ which approximates $\mathcal{F}$. It is represented as

$$\mathcal{P} : p(u,v) = \left(\frac{x(u,v)}{w(u,v)}, \frac{y(u,v)}{w(u,v)}, \frac{z(u,v)}{w(u,v)} \right)^T \qquad (u,v) \in [0,1] \times [0,1] \, .$$

The three numerators $x(u,v)$, $y(u,v)$, $z(u,v)$ and the common denominator $w(u,v)$ are tensor–product polynomials of degree (m,n) in the parameters (u,v). Using the Bernstein polynomials $B_k^l(.)$, and homogeneous coordinates, we may represent it as a tensor-product Bézier patch p^* in $\mathbb{R}^4$ (cf. [10]),

$$\begin{aligned} \mathcal{P}^* : p^*(u,v) &= (x(u,v), y(u,v), z(u,v), w(u,v))^T \\ &= \sum_{i,j=0}^{m,n} B_i^m(u) B_j^n(v) \mathbf{c}_{ij} \qquad (u,v) \in [0,1] \times [0,1] \, . \end{aligned}$$

The control points $\mathbf{c}_{ij}$ consist of four coordinates c_{ij}^x, c_{ij}^y, c_{ij}^z and c_{ij}^w. Note that a parameterization $p(u,v)$ in $\mathbb{R}^3$ corresponds to a one dimensional space of parameterizations $p^*(u,v)$ in $\mathbb{R}^4$, since multiplying all control points $\mathbf{c}_{ij}$ with a constant factor changes $p^*(u,v)$, but the related parameterization $p(u,v)$ remains invariant.

2.2 Distance Measure

The main objective is to approximate $\mathcal{F}$ by a patch $\mathcal{P}$. Hence, we need to measure the approximation quality, which is given by the distance of the two surfaces. As a measure for the approximation quality, we consider the integral

$$I = \int_0^1 \int_0^1 \frac{F^2(p(u,v))}{\|\nabla F(p(u,v))\|^2} \, \mathrm{d}u \, \mathrm{d}v, \tag{3}$$

whose integrand is the so-called squared "Sampson distance" [16]. I is a positive rational functional in the control points $\mathbf{c}_{ij}$. A local minimum represents a local best approximation of $\mathcal{F}$ by a patch $\mathcal{P}$.

Unfortunately, simple minimization of I is a task that is not well posed. The patch $\mathcal{P}$ is neither constrained in size nor position. Consequently, we obtain a local minimum for any patch $\mathcal{P}$ degenerating to a single point located on $\mathcal{F}$. This means (3) yields an infinite number of local minima. In order to obtain a unique solution, additional constraints have to be introduced.

2.3 Constraining the Weights

As described in the previous section, multiplying all control points $\mathbf{c}_{ij}$ with a constant factor leaves $p(u, v)$ invariant. Hence, we have to introduce a normalization in the linear space of the $m \times n$ homogeneous control points.

In addition, a point $p(\tilde{u}, \tilde{v})$ with vanishing denominator (weight) $w(\tilde{u}, \tilde{v}) = 0$ corresponds to a point at infinity, or to a base point (if the three numerators vanish, too). Since we are only interested in regular patches $p(u, v)$ without points at infinity, we have to satisfy the side–condition $w(u, v) \neq 0$.

Both requirements can be taken into account by introducing the auxiliary term

$$J = \int_0^1 \int_0^1 (w(u, v) - 1)^K \,\mathrm{d}u \,\mathrm{d}v,$$

where K is an even number K, in the objective function. (We chose $K = 8$.)

J is a non-negative functional that measures the deviation of the weight coordinates c_{ij}^w from 1. Let ω_J be a small positive weight factor. By adding $\omega_J J$ to the objective function, we obtain a patch $\mathcal{P}$ close to $\mathcal{F}$, with its weight coordinates c_{ij}^w being close to 1. This approach controls the weight coordinates of the control points. By choosing the weight ω 'sufficiently small', points at infinity (poles) can be avoided (see Section 3.3 for more information).

2.4 Controlling the Inner Geometry

Despite the additional term J, the minimization problem still does not have a unique solution. For instance, shrinking a patch $\mathcal{P}$ will usually decrease the values of I and J. Consequently, we have to constrain the size and shape of $\mathcal{P}$. For this purpose we use additional terms which are related to the inner geometry of the surface patch, in the sense of differential geometry [13].

Let g_{11}, g_{12} and g_{22} denote the first metric fundamental forms,

$$g_{11} = \langle p_u, p_u \rangle, \quad g_{12} = \langle p_u, p_v \rangle, \quad g_{22} = \langle p_v, p_v \rangle,$$

where $p_u = (\partial/\partial u)\, p(u, v)$ and $p_v = (\partial/\partial v)\, p(u, v)$ are the partial derivative vectors and $\langle \cdot, \cdot \rangle$ denotes the inner product. For any pair of positive constants (l_1, l_2), the integral

$$L = \int_0^1 \int_0^1 (g_{11} - l_1)^2 + (g_{22} - l_2)^2 \,\mathrm{d}u \,\mathrm{d}v$$

measures the deviation of the length of the first derivatives p_u and p_v from $\sqrt{l_1}$ and $\sqrt{l_2}$.

We choose another small positive weight ω_L and add the term $\omega_L L$ to the objective function. This leads to a more uniformly parameterized surface patch:

in the limit $\omega_L \to \infty$, the parameter lines are traced with the constant speed $\sqrt{l_1}$ and $\sqrt{l_2}$.

The term L does not take the angle between the parameter lines into account. This can be achieved by introducing the term

$$R = \int_0^1 \int_0^1 g_{12}^2 \, \mathrm{d}u \, \mathrm{d}v \, .$$

It penalizes the deviation of the angle between the parameter lines of $p(u,v)$ from a right angle. By adding $\omega_R R$ to the objective function (where ω_R is another non–negative constant), one obtains a patch that approximates the given implicit surface and has almost orthogonal parameter lines. More precisely, in the limit $\omega_L, \omega_R \to \infty$, the surface patch becomes an isometric embedding of a rectangle of size $\sqrt{l_1} \times \sqrt{l_2}$.

Remark 1. Another functional, which has a similar effect to L and R, can be obtained by considering the length of all tangent vectors at a point. If a linear parameterization q maps the parameter domain $[0,1]^2$ into a rectangle with lengths $\sqrt{l_1} \times \sqrt{l_2}$, then the directional derivative vectors

$$|| \, \frac{\mathrm{d}}{\mathrm{d}t} q(u_0 + t\sqrt{l_2}\cos(\phi), v_0 + t\sqrt{l_1}\cos(\phi)) \Big|_{t=0} \, || \tag{4}$$

at all points (u_0, v_0) are unit vectors. Hence, for a general surface p, one might consider the functional

$$\begin{aligned} &\int_0^{2\pi} (|| \frac{\mathrm{d}}{\mathrm{d}t} p(u_0 + t\sqrt{l_2}\cos(\phi), v_0 + t\sqrt{l_1}\cos(\phi))||^2 - 1)^2 \mathrm{d}\phi \\ &= (2 + + \tfrac{3}{4} g_{11}^2 l_2^2 - 2 g_{11} l_2 \tfrac{3}{4} g_{22}^2 l_1^2 - 2 g_{22} l_1 + \tfrac{1}{2} g_{11} g_{22} l_1 l_2 + g_{12}^2 l_1 l_2)\pi. \end{aligned} \tag{5}$$

As a potential advantage, this approach gives functionals which provide certain invariance properties with respect to transformations of the parameter domain. However, this may not be so important, since the space of functions which we are using (tensor–product polynomials) does not have such invariance properties anyway. In contrast with this, the space of all polynomials of certain degree would be invariant.

2.5 Controlling the Position

While the size and the inner geometry of the patch has now been constrained, its position on the given surface $\mathcal{F}$ is still variable, i.e., the patch can still "float" on the surface. We resolve this by pulling the points $p(u_i, v_i)$ of one or more parameter pairs (u_i, v_i) towards user– (or automatically) chosen positions $P_i(p_i^x, p_i^y, p_i^z)$. The sum of the squared Euclidean distances of the points $p(u_i, v_i)$ and points P_i is given by

$$E = \sum_i || \frac{1}{w(u_i, v_i)} \begin{pmatrix} x(u_i, v_i) \\ y(u_i, v_i) \\ z(u_i, v_i) \end{pmatrix} - \begin{pmatrix} p_i^x \\ p_i^y \\ p_i^z \end{pmatrix} ||_2^2 \tag{6}$$

By adding $\omega_E E$ to the objective function, where ω_E is another non–negative constant, the points $p(u_i, v_i)$ will be tied to the points P_i on the surface. As a consequence, the position of the resulting patch is approximately determined. In the examples in section 4 we prescribe the position of the four corner points of the parametric patch.

Note that specifying more than one triple (u_i, v_i, P_i) also affects the inner geometry of the resulting patch. In this case one has to pay attention concerning the term L, i.e., the values l_1 and l_2 need to be chosen suitable to prevent possible conflicts in the constraints.

3 Finding a Solution

The objective function (1) is obtained as the weighted sum of the terms described in the last section. It is a positive rational functional in the control points $\mathbf{c}_{ij}$ of $\mathcal{P}$. As a necessary criteria for a local minimum of S, the first partial derivatives have to vanish. This leads to a system of $M = 4(n+1)(m+1)$ nonlinear equations

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \text{where } \alpha \in \{c_{ij}^x, c_{ij}^y, c_{ij}^z, c_{ij}^w, \}_{\substack{i=0,\dots,m \\ j=0,\dots,n}}. \tag{7}$$

We solve it using Newton's algorithm ([7]), which guarantees fast convergence, provided that a good initial solution is available.

Alternatively, this can be seen as sequential quadratic programming, applied to the problem $S \to \min$. In each step, the objective function is replaced with a local quadratic approximation.

3.1 Computational Details

For each step of Newton's algorithm we need to solve a system of linear equations of size $M \times M$. The elements of the according matrices are the second partial derivatives of S,

$$\frac{\partial^2 S}{\partial \alpha \partial \beta} \qquad \text{where } \alpha, \beta \in \{c_{ij}^x, c_{ij}^y, c_{ij}^z, c_{ij}^w, \}_{\substack{i=0,\dots,m \\ j=0,\dots,n}} . \tag{8}$$

In order to generate this system, we need to compute $\frac{1}{2}M(M+1)$ integrals for each of the terms I, J, L, R, and E. For instance, related to I, we have to evaluate the integrals

$$\int_0^1 \int_0^1 \frac{\partial^2}{\partial \alpha \partial \beta} \frac{F^2(p(u,v))}{\|\nabla F(p(u,v))\|^2} \, \mathrm{d}u \, \mathrm{d}v ,$$

Though possible, the exact evaluation of the integrals is quite expensive. A simple alternative is to use Gaussian quadrature ([7]). As the integrands related to I and E are rational expressions, Gaussian quadrature will yield only approximations of these integrals. The integrals related to J, L, and R will be evaluated exactly, provided that the order of the numerical quadrature is sufficiently high.

Remark 2. In order to facilitate the evaluation of integrals, one may also use polynomial alternatives to the rational integrands in I and E. According to our numerical experience, however, the rational functionals give better results.

3.2 Choice of the Initial Solution, Extrapolation and Iteration

Convergence. The convergence of any Newton–type method depends strongly on the choice of a suitable initial solution. If the initial solution is sufficiently close to the minimum, then the algorithm converges quadratically.

In our situation, we may construct a good initial solution by a geometric approach. If we start with a sufficiently small planar patch which is part of the tangent plane to F at a point, then the iteration process can be expected to converge.

Patch Growing. Clearly, starting with a small planar patch we will obtain only a small resulting patch. Hence, we consider an iterative process to generate larger patches.

We start with a small patch, which has been obtained after several iterations of the Newton method. This patch is extrapolated in order to obtain a larger patch, which is then used as starting patch for a new cycle of Newton's algorithm. The extrapolation is restricted by the distance error, by the weights and by the inner geometry of the obtained bigger patch. This can be expressed by certain thresholds for the resulting value of the objective function.

The feasible values of the extrapolation parameters can be found by a simple bisection procedure.

Note that after each extrapolation step we need to reassign the values l_1, l_2 and the points P_i. The new locations of the P_i (typically representing the expected vertices) can be found by projecting the vertices of the extrapolated patch back onto the surface.

Termination Criteria. As termination criteria for both Newton and extrapolation steps we use the properties of the current patch, which are expressed by the values of the various contributions to the objective function. The overall process is controlled by user defined global limits and thresholds for single steps.

3.3 Adaptation of the Objective Function

Automatic Choice of Points and Lengths. The quantities l_1, l_2 and the points P_i specify the position of the patch and the expected parametric speed $\sqrt{l_1}$ and $\sqrt{l_2}$ of the parameter lines. These values have to comply with the current patch in the iteration process, in order to avoid chaotic behaviour. In our implementation we choose $\sqrt{l_1}$ and $\sqrt{l_2}$ to be equal to the lengths of the current patch. The points P_i are chosen as the footpoints of the points $p(u_i, v_i)$ on $\mathcal{F}$, where p is the current patch.

Automatic Adaptation of the Weights. The sum S and the resulting patch are affected crucially by the choice of the weights ω_J, ω_L, ω_R and ω_E. Of course,

Table 1. Examples: degrees, computing time, # steps

Surface	d	m,n	Time (sec.)	Extrapolation steps	Newton steps
Sphere	2	2,2	1.1	14	46
Minimal Surface	12	3,3	32	10	50
Self-intersecting	8	3,3	10	18	57
Whitney Umbrella	3	3,3	14.91	18	113

Table 2. Values of the weights

	ω_J	ω_L	ω_R	ω_E
start	100	1	1	10^{-2}
lower threshold	10^{-1}	10^{-5}	10^{-5}	10^{-4}

optimal values are not known a priori. Our implementation bypasses this problem by using an automatic adjustment of the weights according to the current contributions to the objective function. During the first steps of the algorithm, higher weights may be necessary in order to stabilize the algorithm, while they may later spoil the approximation quality.

Our main objective is to minimize the distance part I. The other terms are considered as secondary objectives. Let us assume that during the algorithm one of the values $\omega_J J$, $\omega_L L$, $\omega_R R$ or $\omega_E E$ is getting larger than I. This means that we spend most of the effort on minimizing that term instead of I. By lowering the corresponding weight the focus is shifted again to the Sampson distance.

Our implementation uses initial values for the weights ω_J, ω_L, ω_R and ω_E and additional lower thresholds. A weight ω_T is reduced if $\omega_T T$ gets larger than I, and ω_T is larger than the threshold. This approach guarantees a minimal influence of each term.

4 Examples

In order to demonstrate the capabilities and possible applications of the algorithm, we have chosen four examples ranging from very simple to quite challenging.

A summary of the computation time and the performed extrapolation and Newton steps is given in Table 1. Note that these examples all have been computed with the same parameters. The starting values and lower bounds for the weights are shown in Table 2. The upper bound for the total error S was 10^{-6}. In all figures, the size of the bounding cube is 1. In all cases, the algorithm was stable and we obtained satisfying results.

4.1 The Sphere

Our first example is the approximation of a sphere by a rational biquadratic patch. Figure 1 shows the planar starting patch (left), the approximation after the first round of Newton steps (center), and the final approximation (right). Our

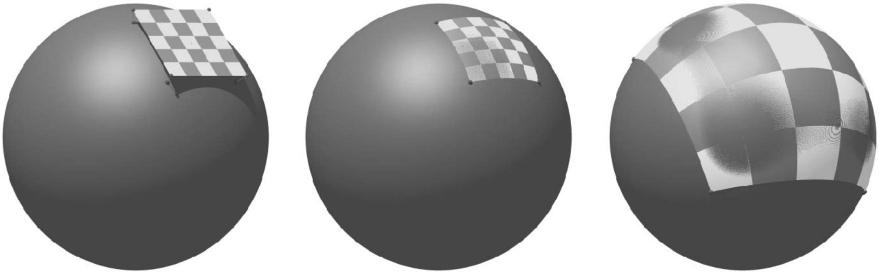


Fig. 1. Biquadratic patch approximating a sphere

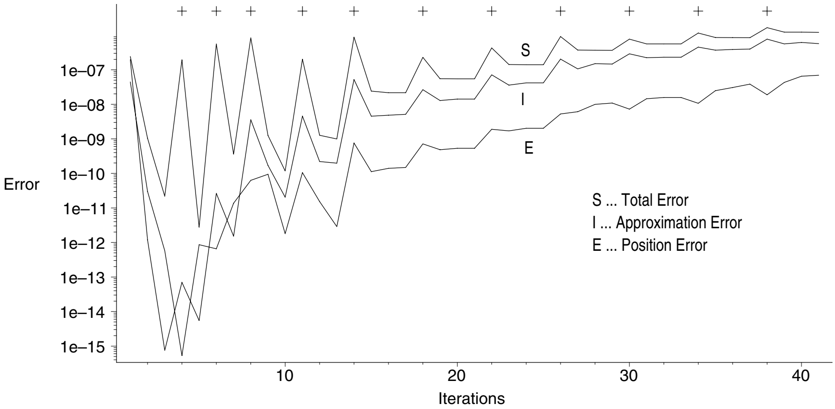


Fig. 2. Contributions to the objective function during the iteration steps

method generates an approximating patch whose parameter lines are nearly isoparametric and approximately orthogonal. The distribution of the parameter lines is visualized by the checkerboard pattern on the surface.

These properties can be enforced by increasing the (lower bounds of the) corresponding weights. On the other hand, the resulting patch will then stay smaller. Figure 2 displays the graphs of the total error S, the approximation error I, and the position error E during the 46 Newton steps. The 14 extrapolation steps correspond to the small peaks. They are also marked by small plus signs on top of the three graphs.

Finally, Figure 3 shows the squared Sampson distance of the final patch with respect to the implicitly given sphere as graph of the domain $[0,1]^2$. Due to the terms controlling the inner geometry, which tend to flatten the surface, the maximal error is present at the four vertices of the patch. Note that the approximation is highly accurate, since the squared Sampson distance (which is a good approximation of the squared distance) is in the order of 10^{-5}, while the radius of the sphere equals 1.

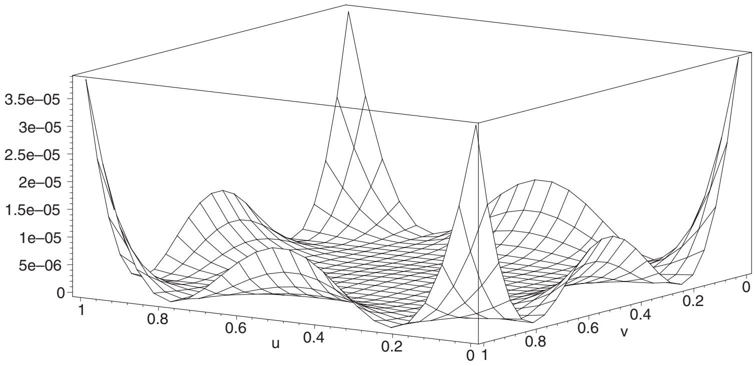


Fig. 3. The Sampson Distance of the final patch

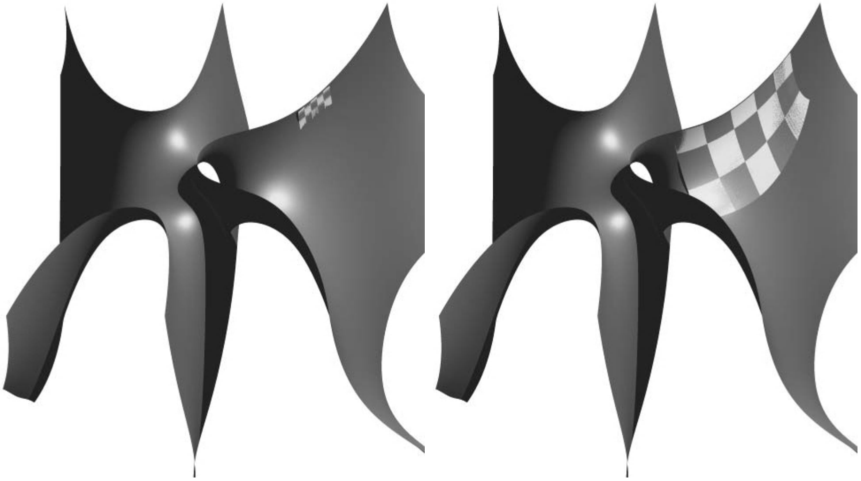


Fig. 4. Parameterization of an algebraic approximation of a minimal surface

4.2 An Approximate Minimal Surface

The second example, which is shown in Figure 4, is the approximation of a minimal surface taken from the Costa-Hoffman-Meeks surface family (see [6, 9]). Note that the upper part of the surface has been cut away, in order to get a better insight into the structure of the surface.

An exact rational parameterization cannot be found for this surface, since its topological genus is 1. Using our numerical method we can still generate finite patches approximating the surface with a high accuracy. Similar to the sphere case, the figure shows the initial solution and the final result.

4.3 Surfaces with Singularities

The remaining two examples (Figures 5 and 6) demonstrate that the method is able to handle self-intersections. We start with a small patch on one side of the self-intersection curve and finally get an approximation that 'dives through' the singularity and continues on the correct branch of the surface.

Once again, the figures shows the initial solution and the final result.

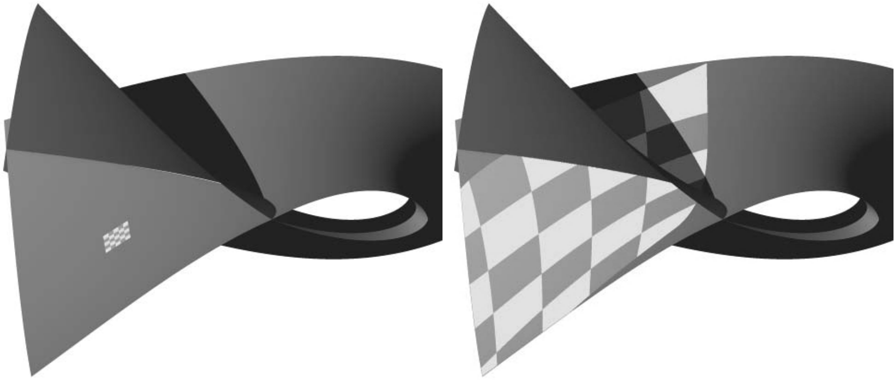


Fig. 5. Self-intersecting surface of degree 8

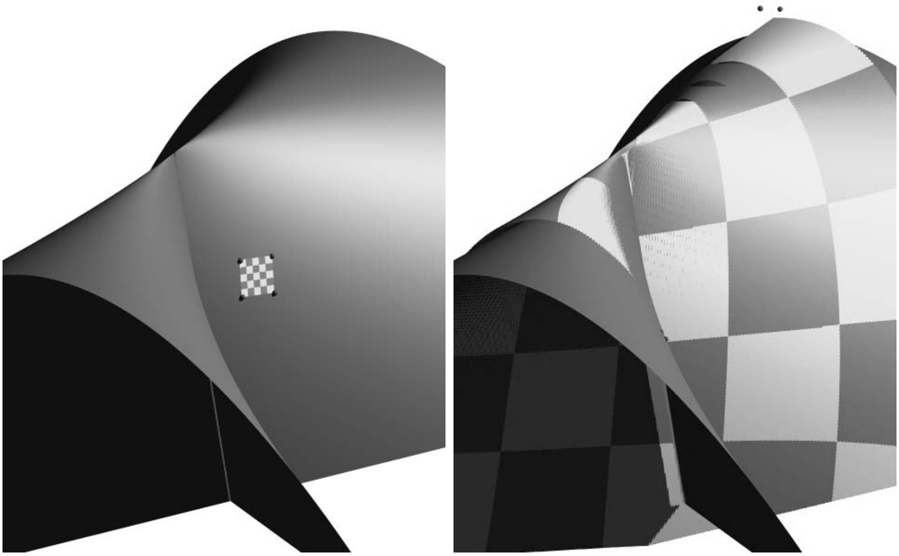


Fig. 6. Whitney Umbrella

5 Concluding Remarks

We presented a method for the approximation of an implicitly defined surface by a rational patch. The main ingredient is the minimization of the Sampson distance of the two surfaces, while additional side constraints are used to determine the inner geometry and the position of the parametric patch. The objective functional is minimized using of Newton's algorithm and Gaussian quadrature. In order to maintain a good initial solution, we alternate extrapolation steps and approximation steps, producing surface patches of optimal size, according to the specified criteria.

As a matter of future research, we plan to consider the problem of covering the whole implicitly defined surface. This can be achieved either by collecting several patches, which have been obtained starting from several seed points, or by parameterizing the surface not with a single patch, but with a spline surface. In order to use the latter approach, the extrapolation step should be modified so as to permit adding new segments to the spline surface.

The possible applications of the parameterization technique include the construction of rational surface patches from unorganized point data. As a first step, one may fit an algebraic spline surface to these data, e.g., using techniques as in [11]. In a second step, the implicitly defined surface can then be parameterized, using the technique described in this paper.

While other methods either have to address the parameterization problem [10] or depend on an initial solution [15], the combination of implicit fitting and approximate parameterization may help to circumvent both problems. Moreover, it allows for fully exploiting the weights of the rational surface representation. This can be highly useful for generating exact descriptions of many important classes of surfaces, such as natural quadrics. In addition, the use of the additional term controlling the inner geometry may – in combination with the error term – help to generate a segmentation of the surface.

Preliminary results are shown in Figure 7. We start from a point cloud with $11,366$ points, which represents a cylinder with a cylindrical hole. The point cloud is the input data for the approximate implicitization algorithm described in [19]. The result, shown in figure 7 (middle), is a piecewise algebraic surface,

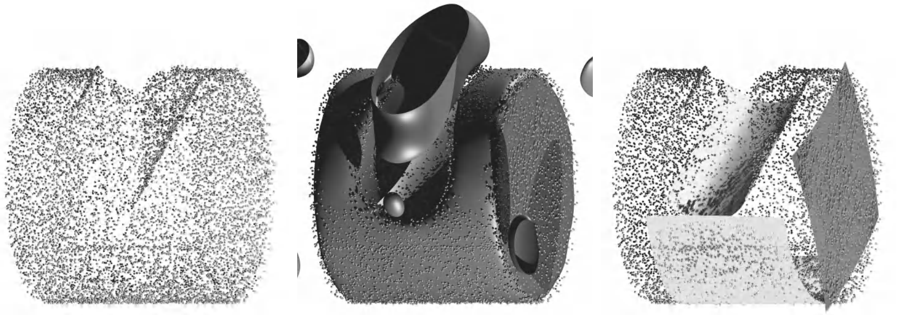


Fig. 7. Surface Reconstruction: Point cloud (left), piecewise implicit approximation (middle), parametric approximation (right)

which consists of 214 subpatches of tri-degree 3. One peculiar disadvantage of this implicit approximation is that it introduces additional branches.

We use the implicit approximation as input for the approximate rational parameterization algorithm described in this paper. Figure 7 (right) shows the results for three different starting patches on different sides of the object. As a byproduct of the procedure, we may identify the cylinder, the hole and the planar top. (Clearly, similar results can be obtained using existing techniques for automatic segmentation, which are often based on the analysis of the surface normals [14].) Note that the algorithm stops from growing the patches near regions of high curvature. This is due to the terms controlling the inner geometry.

Acknowledgement. This work was supported by the Austrian Science Fund (FWF) through subproject 15 of the Special Research Area SFB F013 "Numerical and Symbolic Scientific Computing" at Linz.

References

1. Abhyankar S., Bajaj C.: Automatic Parameterization of Rational Curves and Surfaces I: Conics and Conicoids. Computer Aided Design, **19** (1987), 11-14
2. Abhyankar S., Bajaj C.: Automatic Parameterization of Rational Curves and Surfaces II: Cubics and Cubicoids. Computer Aided Design, **19** (1987), 499-502
3. Bajaj C.: The Emergence of Algebraic Curves and Surfaces, in Geometric Design – Directions in Geometric Computing (R. Martin, ed.), Information Geometers Press, 1993, 1-29.
4. Bajaj C., Xu G.: Spline Approximations of Real Algebraic Surfaces. J. Symb. Comp., Special Issue on Parametric Algebraic Curves and Applications, **23** (1997), 315-333
5. Bajaj C., Holt R.L., Netravali A.N.: Rational Parametrizations of Nonsingular Real Cubic Surfaces. ACM Trans. Graph. **17**(1): 1-31 (1998)
6. Costa A.: Examples of a Complete Minimal Immersion in of Genus One and Three Embedded Ends. Bil. Soc. Bras. Mat. **15** (1984), 47-54.
7. Hämmerlin G., Hoffmann K.H.: Numerical Mathematics, Springer, 1991,
8. Hartmann E.: Numerical parameterization of curves and surfaces, Computer Aided Geometric Design **17** (2000), 251-266.
9. Hoffman D. and Meeks W. H. III: A Complete Embedded Minimal Surfaces in with Genus One and Three Ends. J. Diff. Geom. **21** (1985), 109-127.
10. Hoschek J., Lasser D.: Fundamentals of Computer Aided Geometric Design, 1993, A. K. Peters.
11. Jüttler, B., Felis, A., Least-squares fitting of algebraic spline surfaces, Advances in Computational Mathematics **17** (2002), 135–152.
12. Jüttler B. and Rittenschober K., Using line congruences for parameterizing special algebraic surfaces. The Mathematics of Surfaces X (R. Martin and M. Bloor, eds.), pages 223-243, Berlin, 2003. Springer.
13. Kreyszig, E., Differential Geometry, Dover, New York, 1990.
14. Varadý, T., Martin, R: Reverse Engineering, in Handbook of Computer Aided Geometric Design (G. Farin, J. Hoschek, M.-S. Kim, eds.) , North-Holland, 2002, 651–681.

15. Pottmann H. and Leopoldseder S.: A concept for parametric surface fitting which avoids the parametrization problem Computer Aided Geometric Design, **20** (2003), 343-362
16. Sampson, P.: Fitting conic sections to very scattered data: an iterative refinement of the Bookstein algorithm, Computer Graphics and Image Processing **18** (1982), 97–108.
17. Schicho J.: Rational parametrization of surfaces, J. Symb. Comp., **26** (1998), 1-30.
18. Sederberg T. W., Snively J.: Parameterizing Cubic Algebraic Surfaces. The Mathematics of Surfaces II (R.R. Martin, ed.), Oxford University Press, (1987), pp. 299–320.
19. Wurm E., Jü ttler B.: Approximate Implicitization via Curve Fitting, in Symposium on Geometry Processing (L. Kobbelt, P. Schröder, H. Hoppe, eds.), Eurographics / ACM Siggraph, New York 2003, 240-247.

Convergence Analysis of Discrete Differential Geometry Operators over Surfaces

Zhiqiang Xu[1], Guoliang Xu[2], and Jia-Guang Sun[3]

[1] Department of Computer Science, Tsinghua University, Beijing 100084, China
xuzq@tsinghua.edu.cn
[2] ICMSEC, LSEC, Academy of Mathematics and System Sciences,
Chinese Academy of Sciences, Beijing, China
xuguo@lsec.cc.ac.cn
[3] School of Software, Tsinghua University, Beijing 100084, China
sunjg@tsinghua.edu.cn

Abstract. In this paper, we study the convergence property of several discrete schemes of the surface normal. We show that the arithmetic mean, area-weighted averaging, and angle-weighted averaging schemes have quadratic convergence rate for a special triangulation scenario of the surfaces. By constructing a counterexample, we also show that it is impossible to find a discrete scheme of normals that has quadratic convergence rate over any triangulated surface and hence give a negative answer for the open question raised by D.S.Meek and D.J. Walton. Moreover, we point out that one cannot build a discrete scheme for Gaussian curvature, mean curvature and Laplace-Beltrami operator that converges over any triangulated surface.

1 Introduction

Estimation of normal vectors and curvatures on discrete surfaces are often required in Computer Aided Geometric Design and Computer Graphics. In the past decades, many discretized approaches for normal vectors, Gaussian curvature, mean curvature and Laplace-Beltrami operator have been proposed and used. The convergence of the discretized approaches has also been studied. In [5], the authors analyzed the convergence of the normal vector and Gaussian curvature. For normal vectors, they obtained the following result: *for non-uniform data, the unit vector parallel to the arithmetic mean of unit normals of the triangular faces around a point approximates the unit normal of the surface at that point to accuracy $O(h)$.* Furthermore, by the numerical test, they found that the accuracy of the arithmetic mean, area-weighted averaging, and angle-weighted averaging are not higher than $O(h)$. As pointed out in [5], normal estimation methods with accuracy $O(h^2)$ are very useful for the spherical image method of Gaussian curvature approximation. Hence, they raised an open question:*find a linear combination of the normals of the triangular faces, based on geometric considerations, that approximates the normal of the surface to $O(h^2)$.* In this paper, we prove that under certain conditions, the approximation accuracy

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 448–457, 2005.

of normal vectors can be $O(h^2)$, meaning the approximation converges with a quadratic rate. Moreover, we show that it is impossible to find a discretization scheme of normals that has quadratic convergence rate over any triangulated surface. Hence, the answer to the above mentioned open question is negative.

In [6], Meyer et al. proposed some discrete schemes to approximate several important geometric attributes, including normal vectors and curvatures on arbitrary triangular meshes. In [9], G. Xu proved that a well known discretized scheme of Gaussian curvature, derived from Gauss-Bonnet theorem, has quadratic convergence rate under certain conditions. In [10] and [11], he also studied the convergence of Laplace-Beltrami operators and mean curvature, include Taubin et al's discretization [7], Mayer et al's discretization [4], Desbrun et al's discretization[1], Meyer et al 's discretization[6], and proposed several simple discretization schemes of Laplace-Beltrami operator over triangulated surfaces. In [5], the author proposes an asymptotic analysis of Gaussian curvature for three methods: quadratic fit method, angular defect and spherical image method. A review of these schemes is given in [3]. However, none of these discretizations of Gaussian curvature and mean curvature has been proved to be convergent over any non-degenerate triangulated surface. Therefore, a natural questions is raised: *can one build a discrete scheme of Gaussian curvature and mean curvature which involves one-ring vertices and converges over any non-degenerate triangle surface?* In this paper, we shall give a negative answer to this question. Hence, we have to accept the fact that the discretization scheme for curvature only convergent over special triangular surface.

The rest of the paper is organized as follows. In Section 2, we introduce some definitions and formulations. In Section 3, we discuss the convergence property of discrete schemes of normals. In Section 4, by giving a counterexample, we show that one cannot construct a scheme of Gaussian curvature and mean curvature that converges over any non-degenerate triangle surface. Moreover, we also give a negative answer to the open question raised in [5].

2 Preliminaries

Let $\mathbf{S}(u,v) = (x(u,v), y(u,v), z(u,v)) \in \mathbf{R}^3$ be a regular parametric surface. We further assume that the point where the normal and curvature need to be approximated is $\mathbf{O} : (x(0,0), y(0,0), z(0,0))^T$. Then from differential geometry, the normal vector of $\mathbf{S}(u,u)$ at $\mathbf{O}$ is $\mathbf{S_u}(\mathbf{0},\mathbf{0}) \times \mathbf{S_v}(\mathbf{0},\mathbf{0})$.

Let $\mathbf{P}_i = \mathbf{S}(\mathbf{q}_i)$ be n distinct points on $\mathbf{S}(x,y)$ near the point $(x(0,0), y(0,0), z(0,0))^T$ and $\mathbf{q}_i = (r_i\cos(\theta_i)h, r_i\sin(\theta_i)h)$. The indices arithmetic modulo n so index $n+1$ is the same as index 1. Without loss of generality, we assume $0 \le \theta_1 \le \theta_2 \le \cdots \le \theta_n < 2\pi$. Denote the normal to the triangle $\mathbf{P}_i\mathbf{O}\mathbf{P}_{i+1}$ as $\mathbf{n}_{i,i+1}$, by using Taylor expansion,

$$\begin{aligned}\mathbf{n}_{i,i+1} &= (\mathbf{P}_i - \mathbf{O}) \times (\mathbf{P}_{i+1} - \mathbf{O}) \qquad (1)\\ &= \begin{pmatrix} (y_u z_v - y_v z_u)\sin(\theta_{i+1} - \theta_i) r_i r_{i+1} h^2 + A_i h^3 + O(h^4) \\ -((x_u z_v - x_v z_u)\sin(\theta_{i+1} - \theta_i) r_i r_{i+1} h^2 + B_i h^3 + O(h^4)) \\ (x_u y_v - x_v y_u)\sin(\theta_{i+1} - \theta_i) r_i r_{i+1} h^2 + C_i h^3 + O(h^4) \end{pmatrix},\end{aligned}$$

where

$$A_i = (y_{uu}\cos^2\theta_i + 2y_{uv}\cos\theta_i\sin\theta_i + y_{vv}\sin^2\theta_i)(z_u\cos\theta_{i+1} + z_v\sin\theta_{i+1})r_i^2r_{i+1}$$
$$- (z_{uu}\cos^2\theta_i + 2z_{uv}\cos\theta_i\sin\theta_i + z_{vv}\sin^2\theta_i)(y_u\cos\theta_{i+1} + y_v\sin\theta_{i+1})r_ir_{i+1}^2,$$

$$B_i = (x_{uu}\cos^2\theta_i + 2x_{uv}\cos\theta_i\sin\theta_i + x_{vv}\sin^2\theta_i)(z_u\cos\theta_{i+1} + z_v\sin\theta_{i+1})r_i^2r_{i+1}$$
$$- (z_{uu}\cos^2\theta_i + 2z_{uv}\cos\theta_i\sin\theta_i + z_{vv}\sin^2\theta_i)(x_u\cos\theta_{i+1} + x_v\sin\theta_{i+1})r_ir_{i+1}^2,$$

$$C_i = (x_{uu}\cos^2\theta_i + 2x_{uv}\cos\theta_i\sin\theta_i + x_{vv}\sin^2\theta_i)(y_u\cos\theta_{i+1} + y_v\sin\theta_{i+1})r_i^2r_{i+1}$$
$$- (x_{uu}\cos^2\theta_i + 2x_{uv}\cos\theta_i\sin\theta_i + x_{vv}\sin^2\theta_i)(y_u\cos\theta_{i+1} + y_v\sin\theta_{i+1})r_ir_{i+1}^2.$$

Denote the unit normal vector on triangle surfaces $\mathbf{P}_i\mathbf{O}\mathbf{P}_{i+1}$ as $\overline{\mathbf{n}}_{i,i+1} := \mathbf{n}_{i,i+1}/\|\mathbf{n}_{i,i+1}\|$. By using the formulation above, we have

$$\overline{\mathbf{n}}_{i,i+1} = \overline{\mathbf{n}}_0(1 - \frac{A_i(y_uz_v - y_vz_u)h - B_i(x_uz_v - x_vz_u)h + C_i(x_uy_v - x_vy_u)h}{\sin(\theta_{i+1} - \theta_i)r_ir_{i+1}} + O(h^2)). \tag{2}$$

where $\overline{\mathbf{n}}_0$ is the unit normal vector at $\mathbf{O}$, i.e.

$$\overline{\mathbf{n}}_0 = \frac{(y_uz_v - y_vz_u, -(x_uz_v - x_vz_u), x_uy_v - x_vy_u)^T}{\sqrt{(y_uz_v - y_vz_u)^2 + (x_uz_v - x_vz_u)^2 + (x_uy_v - x_vy_u)^2}}.$$

In general, the unit normal vector at $\mathbf{O}$ is approximated by

$$\sum_{i=1}^{n} \lambda_i\overline{\mathbf{n}}_{i,i+1}, \tag{3}$$

where λ_i is weight and $\sum_{i=1}^n \lambda_i = 1$. By (2), we find the covergence rate of the discrete scheme is $O(h)$, which agrees with the result in [5].

There are several ways to determine the weights. A simple way is to take arithmetic mean, i.e., $\lambda_i = \frac{1}{n}$. Other ways include take an area-weighted average and an angle-weighted average.

Using discretization normals, the spherical image method for Gaussian curvature approximation is built in [5], and moreover, the following lemma is proved.

Lemma 1. (see [5]) *When unit normals are known to accuracy $O(h^2)$, the spherical image method approximates the Gaussian curvature to accuracy $O(h)$.*

Hence, $O(h^2)$ accuracy normals are very useful for computing the Gaussian curvature.

3 Convergence of Normal Vectors

In [5], the authors showed that the accuracy of the arithmetic mean, area-weighted averaging, and angle-weighted averaging are not higher than $O(h)$.

However, we shall prove that under certain conditions, the approximation accuracy of the three ways can be $O(h^2)$.

We firstly exhibit the numerical behaviors of the discrete schemes of the surface normal. To show the numerical behavior of the discrete schemes , we take several two variable functions over xy-plane as three dimensional surfaces so that the exact normal can be computed. Both the exact and approximated normals are computed at some selected domain points $q_{ij} = (x_i, y_j) = (i/20, j/20), i = 1, \cdots, 19, j = 1, \cdots, 19$. The surfaces are triangulated around q_{ij} by triangulating the domain first, with mapping the planner triangulation onto the surfaces by the selected bivariate functions. As a simple case, the domain around q_{ij} is triangulated locally by choosing n regularly distributed points:

$$q_k = q_{ij} + h(\cos(\theta_k), \sin(\theta_k)), \theta_k = 2(k-1)\pi/n, k = 1, \cdots, n.$$

The convergence rate are checked by taking $h = 1/8, 1/16, 1/32, \cdots$ and $n = 3, 4, \cdots, 9$.

The functions we use are the following

$$\begin{aligned} F_1(x,y) &= \sqrt{4-(x-0.5)^2-(y-0.5)^2}, \\ F_2(x,y) &= \exp(-5((x-0.5)^2+(y-0.5)^2)), \\ F_3(x,y) &= \tan(5y-5x), \\ F_4(x,y) &= \frac{1+\cos(5y)}{6+6(3x-1)^2}. \end{aligned}$$

Denote $e_1(F_j, n), e_2(F_j, n)$ and $e_3(F_j, n)$ as the maximal error of the approximated surface normals computed by the arithmetic mean scheme,angle-weighted averaging and area-weighted averaging over the above mentioned local triangulations and the exact normal vector computed from the continuous surfaces defined by F_j. Tables 1–3 show the asymptotic maximal error $e_1(F_j, n), e_2(F_j, n)$ and $e_3(F_j, n)$.

From the above numerical resuls, we find the arithmetic mean scheme and area-weighted averaging can converge in the rate $O(h^2)$ for the $n > 3$ regularly distributed domain vertices. When the valence n is 3, in general, the approximate surface normal converges in the rate $O(h)$. Moreover, if n is even, the angle-weighted averaging can converge in the rate $O(h^2)$.

Table 1. The maximal errors of the arithmetic mean scheme

n	$e_1(F_1, n)$	$e_1(F_2, n)$	$e_1(F_3, n)$	$e_1(F_4, n)$
3	$1.7291e-02 \times h$	$8.2968e-01 \times h$	$1.7331e+00 \times h^2$	$7.16229e-01 \times h$
4	$7.2445e-02 \times h^2$	$5.4452e-01 \times h^2$	$1.1554e+00 \times h^2$	$6.3986e-01 \times h^2$
5	$6.0409e-02 \times h^2$	$5.9247e-01 \times h^2$	$1.73312e+00 \times h^2$	$8.6673e-01 \times h^2$
6	$5.6638e-02 \times h^2$	$6.6982e-01 \times h^2$	$1.73310e+00 \times h^2$	$9.0644e-01 \times h^2$
7	$5.4428e-02 \times h^2$	$7.2413e-01 \times h^2$	$1.73310e+00 \times h^2$	$9.8637e-01 \times h^2$
8	$5.3324e-02 \times h^2$	$7.6641e-01 \times h^2$	$1.73310e+00 \times h^2$	$1.0233e-00 \times h^2$
9	$5.2663e-02 \times h^2$	$7.9757e-01 \times h^2$	$1.73310e+00 \times h^2$	$1.0458e-00 \times h^2$

Table 2. The maximal errors of the angle-weighted averaging

n	$e_2(F_1,n)$	$e_2(F_2,n)$	$e_2(F_3,n)$	$e_2(F_4,n)$
3	$3.0024e-02 \times h$	$7.7029e-01 \times h$	$2.1349e+00 \times h^2$	$7.1111e-01 \times h$
4	$6.81814e-02 \times h^2$	$6.2178e-01 \times h^2$	$1.1560e+00 \times h^2$	$6.1403e-01 \times h^2$
5	$5.33700e-02 \times h^2$	$2.8789e-02 \times h$	$1.4518e+00 \times h^2$	$1.9115e-02 \times h$
6	$4.9291e-02 \times h^2$	$6.5325e-01 \times h^2$	$1.8996e+00 \times h^2$	$8.7364e-01 \times h^2$
7	$4.6771e-02 \times h^2$	$1.3400e-03 \times h$	$1.0810e+00 \times h^2$	$1.2636e-03 \times h$
8	$4.5462e-02 \times h^2$	$7.2842e-01 \times h^2$	$1.2577e+00 \times h^2$	$9.2338e-00 \times h^2$
9	$4.4681e-02 \times h^2$	$1.5197e-03 \times h$	$8.8635e-01 \times h^2$	$3.6756e-03 \times h$

Table 3. The maximal errors of area-weighted averaging

n	$e_3(F_1,n)$	$e_3(F_2,n)$	$e_3(F_3,n)$	$e_3(F_4,n)$
3	$1.7281e-02 \times h$	$8.2961e-01 \times h$	$1.7331e+00 \times h^2$	$7.1663e-01 \times h$
4	$8.4120e-02 \times h^2$	$4.3854e-01 \times h^2$	$1.1554e+00 \times h^2$	$6.1402e-01 \times h^2$
5	$7.3895e-02 \times h^2$	$8.0248e-01 \times h^2$	$1.7331e+00 \times h^2$	$7.5509e-01 \times h^2$
6	$6.8800e-02 \times h^2$	$8.5412e-01 \times h^2$	$1.7331e+00 \times h^2$	$8.1070e-01 \times h^2$
7	$6.7316e-02 \times h^2$	$8.9488e-01 \times h^2$	$1.7331e+00 \times h^2$	$8.4060e-01 \times h^2$
8	$6.6484e-02 \times h^2$	$9.2511e-01 \times h^2$	$1.7331e+00 \times h^2$	$8.6749e-01 \times h^2$
9	$6.4620e-02 \times h^2$	$9.4713e-01 \times h^2$	$1.7331e+00 \times h^2$	$8.8737e-01 \times h^2$

In the following, we shall give a sufficient condition for the convergence in rate $O(h^2)$.

Theorem 1. *Let p_0 be a vertex of M with valence n, and $p_i, i = 1, \cdots, n$ be its neighbor vertices. Suppose p_0 and $p_i, i = 1, \cdots, n$ are on a sufficiently smooth regular parametric surface $\mathbf{S}(x, y) \in \mathbf{R}^3$ and there exist $q_0, q_i \in \mathbf{R}^2$ such that $p_0 = \mathbf{S}(q_0), p_i = \mathbf{S}(q_i)$. Then in the following two cases*

(1). $n = 2m, m > 1, q_{i+m} = q_0 - (q_i - q_0), \sum_{i=1}^{m} \lambda_i = 1, \lambda_{i+m} = \lambda_i$,

(2). $n = 2m+1, m > 1, \angle q_i q_0 q_{i+1} = \frac{2\pi}{2m+1}, \|q_i - q_0\| = \|q_{i+1} - q_0\|, \sum_{i=1}^{m} \lambda_i = 1, \lambda_{i+1} = \lambda_i$,

$\sum_{i=1}^{n} \lambda_i \overline{\mathbf{n}}_{i,i+1}$ approximates the unit normal of the surface at the point p_0 to the accuracy $O(h^2)$.

Proof. Without loss of generality, we may assume $q_0 = (0, 0)$ and $q_i = (r_i \cos(\theta_i)h, r_i \sin(\theta_i)h)$. Since $\mathbf{S}(x, y)$ is a regular surface, we can use the notations and formulas proposed in Section 2.

It follows from (2) that,

$$\overline{\mathbf{n}}_{i,i+1} = \overline{\mathbf{n}}_0 (1 - \frac{A_i(y_u z_v - y_v z_u)h - B_i(x_u z_v - x_v z_u)h + C_i(x_u y_v - x_v y_u)h}{\sin(\theta_{i+1} - \theta_i) r_i r_{i+1}} + O(h^2)).$$

Consider $\sum_{i=1}^{n} \lambda_i \overline{\mathbf{n}}_{i,i+1}$. By the explicit formulation of $\overline{\mathbf{n}}_{i,i+1}$, to prove the theorem,we merely need to prove

$$\sum_{i=1}^{n} \lambda_i \frac{A_i}{(\sin \theta_{i+1} - \theta_i) r_i r_{i+1}} = 0, \quad \sum_{i=1}^{n} \lambda_i \frac{B_i}{(\sin \theta_{i+1} - \theta_i) r_i r_{i+1}} = 0,$$

$$\sum_{i=1}^{n} \lambda_i \frac{C_i}{(\sin\theta_{i+1} - \theta_i) r_i r_{i+1}} = 0.$$

Firstly, we consider the case where $n = 2m$. Since $q_{i+m} = q_0 - (q_i - q_0)$, we have $\theta_{i+m} = \pi + \theta_i, r_{i+m} = r_i$. Hence,

$$\begin{aligned}
&\sum_{i=1}^{n} \lambda_i \frac{r_{i+1}\cos^2\theta_{i+1}\sin\theta_i}{\sin(\theta_{i+1} - \theta_i)}\\
&= \sum_{i=1}^{m} \lambda_i \frac{r_{i+1}\cos^2\theta_{i+1}\sin\theta_i}{\sin(\theta_{i+1} - \theta_i)} + \sum_{i=m+1}^{n} \lambda_{i-m} \frac{r_{i+1-m}\cos^2(\pi + \theta_{i+1-m})\sin(\pi + \theta_{i-m})}{\sin(\theta_{i+1-m} - \theta_{i-m})}\\
&= \sum_{i=1}^{m} \lambda_i \frac{r_{i+1}\cos^2\theta_{i+1}\sin\theta_i}{\sin(\theta_{i+1} - \theta_i)} + \sum_{i=1}^{m} \lambda_i \frac{-r_{i+1}\cos^2\theta_{i+1}\sin\theta_i}{\sin(\theta_{i+1} - \theta_i)}\\
&\equiv 0.
\end{aligned}$$

Using similar method, $\sum_{i=1}^{n} \lambda_i \frac{A_i}{(\sin\theta_{i+1} - \theta_i) r_i r_{i+1}} \equiv 0$.

Secondly, we consider the case where $n = 2m + 1$. In this case, λ_i, r_i and $\theta_{i+1} - \theta_i$ are all constant. Hence, to prove $\sum_{i=1}^{n} \lambda_i \frac{A_i}{(\sin\theta_{i+1} - \theta_i) r_i r_{i+1}} \equiv 0$, we only need prove $\sum_{i=1}^{n} \cos^2\theta_i \sin\theta_{i+1} = 0, \sum_{i=1}^{n} \cos^2\theta_i \cos\theta_{i+1} = 0$, $\sum_{i=1}^{n} \cos\theta_i \sin\theta_i \sin\theta_{i+1} = 0, \sum_{i=1}^{n} \cos\theta_i \sin\theta_i \cos\theta_{i+1} = 0$, $\sum_{i=1}^{n} \cos\theta_{i+1} \sin^2\theta_i = 0$, and $\sum_{i=1}^{n} \sin\theta_{i+1} \sin^2\theta_i = 0$. We only prove one equation, with the proof of other equations being similar. Consider

$$\begin{aligned}
\sum_{i=1}^{n} \cos^2\theta_i \sin\theta_{i+1} &= \sum_{i=0}^{2m} \cos^2 \frac{i-1}{2m+1} 2\pi \sin \frac{i}{2m+1} 2\pi\\
&= 2\sum_{i=1}^{2m} \sin \frac{i}{2m+1} 2\pi \cos \frac{2(i-1)}{2m+1} 2\pi - \sum_{i=1}^{2m} \sin \frac{i}{2m+1} 2\pi.
\end{aligned}$$

Using the equality $\sum_{k=1}^{2m} \sin(a_0 + kd) = \frac{\cos(d/2 + a_0) - \cos(a_0 + 2md + d/2)}{2\sin d/2}$, we have

$$\begin{aligned}
&2\sum_{i=1}^{2m} \sin \frac{i}{2m+1} 2\pi \cos \frac{2(i-11)}{2m+1} 2\pi - \sum_{i=1}^{2m} \sin \frac{i}{2m+1} 2\pi\\
&= \sum_{i=1}^{2m} (\sin \frac{3i-2}{2m+1} 2\pi + \sin \frac{-i+2}{2m+1} 2\pi) - \sum_{i=1}^{2m} \sin \frac{i}{2m+1} 2\pi\\
&= 0.
\end{aligned}$$

Using the similar derivation above, we can prove $\sum_{i=1}^{n} \lambda_i \frac{B_i}{(\sin\theta_{i+1} - \theta_i) r_i r_{i+1}} = 0$ and $\sum_{i=1}^{n} \lambda_i \frac{C_i}{(\sin\theta_{i+1} - \theta_i) r_i r_{i+1}} = 0$.

Hence, under the condition (1) or (2),$\sum_{i=1}^{n} \lambda_i \overline{\mathbf{n}}_{i,i+1}$ has quadratic convergence rate. The theorem is proved.

Corollary 1. *Under the conditions of Theorem 1, if the weight λ_i is defined as the arithmetic mean or the area-weighted averaging $\sum_{i=1}^{n} \lambda_i \overline{\mathbf{n}}_{i,i+1}$ approximates the unit normal of the surface at the point p_0 to accuracy $O(h^2)$.*

Proof. When λ_i is selected as arithmetic mean, $\lambda_i = \frac{1}{n}$. Obviously, in this case $\lambda_i = \lambda_j, \forall i, j$. By Theorem 1, the Corollary holds, when λ_i is defined as the arithmetic mean. Denote the area of $\triangle p_i p_0 p_{i+1}$ as $A(p_i p_0 p_{i+1})$. Then, we have $A(p_i p_0 p_{i+1}) = \frac{1}{2}\|\mathbf{n}_{i,i+1}\|$. Under the condition of Theorem 1, it is easy to see that the coefficient of h in $\sum_{i=1}^{n} A(p_i p_0 p_{i+1})$ and $\sum_{i=1}^{n} \mathbf{n}_{i,i+1}$ is cancelled. The Corollary holds.

Corollary 2. *Under the condition (1) of Theorem 1, if the weight λ_i is defined as the angle-weighted averaging $\sum_{i=1}^{n} \lambda_i \overline{\mathbf{n}}_{i,i+1}$ approximates the unit normal of the surface at the point p_0 to accuracy $O(h^2)$.*

Proof. Let $\theta_{i,i+1}$ be the planar angle $p_i p_0 p_{i+1}$ and let it be positive by convention. Then we can derive, $\theta_{i,i+1} = \theta_{i+1} - \theta_i + a^{(i)}h + O(h^2)$. Under the condition (1) of Theorem 1, $a^{(i+m)} = -a^{(i)}$. Hence, it is easy to see that the coefficient of h in $\lambda_i \overline{\mathbf{n}}_{i,i+1}$ is cancelled. The Corollary is proved.

Remark 1. *The convergence results are established under particular conditions. As pointed out in [11], these special cases are very useful and important. A number of numerical simulations of geometric partial differential equations are conducted over a triangulated domain formed by a uniform three-directional or four-directional partition. Both partitions satisfy the condition in Theorem 1.*

Remark 2. *An interesting observation is that the condition of the discretization scheme of normals having quadratic convergence is exactly the same as the condition of a discretization of gradient having quadratic convergence proposed in [11]. (Theorem 4.1 in [11] presents only the condition (1). In fact, by using similar method with [11], under the condition (2), the discretization of gradient has also quadratic convergence.)*

4 Counterexamples to Convergence of Curvature and Normals

In the previous section, we have studied the convergence of the discrete unit normal. The convergence property of the discrete Gaussian curvature and mean curvature has been considered in [9],[10],[11] and [5]. But none of discretization schemes has been proved to be convergent over any non-degenerate triangle surfaces. A natural questions is raised: *can one build a discretization scheme of Gaussian curvature and mean curvature converging over any non-degenerate triangle surfaces?* In this section, by a counterexample we shall give a negative answer for the question.

Let p_0 be a vertex of M where the Gaussian curvature is are to be approximated and $p_i, i = 1, \cdots, n$ be its neighbor vertices. We make a hypothesis that the discretization scheme of Gaussian curvature involving one-ring neighbor vertices of p_0, denoted as $H(M, p_0; p_1, \cdots, p_n)$, is convergent for any triangle mesh surface M. Suppose M is a given triangle surface approximating the surface $\mathbf{S}(x,y) = (x, y, f(x,y))^T$, $f(x,y) = B_{02}x^2 + B_{11}xy + B_{02}y^2$ and the origin $p_0 = (0,0,0)$ is a vertex of M. Assume the valence of the origin point is 4 and the neighbor points are $p_i = \mathbf{S}(\mathbf{q}_i), i = 1, \cdots, 4$, where $\mathbf{q}_1 = h(1,2), \mathbf{q}_2 = h(-1,-2), \mathbf{q}_3 = h(-1,2)$ and $\mathbf{q}_4 = h(1,-2)$(see fig.1.a). Since Gaussian curvature of $\mathbf{S}(x,y,z)$ at p_0 equals to $4B_{02}B_{20} - B_{11}^2$, by the convergence property of $H(M, p_0; p_1, \cdots, p_n)$, we have $\lim_{h\to 0} H(M, p_0; p_1, p_2, p_3, p_4) = 4B_{02}B_{20} - B_{11}^2$.

Suppose $\widehat{M}$ is another given mesh surface approximating the surface $\widehat{\mathbf{S}}(x,y) = (x, y, \widehat{f}(x,y))^T$, $\widehat{f}(x,y) = (4B_{02} + B_{20})x^2 + B_{11}xy$, and the origin $\mathbf{O} : (0,0,0)$ is a vertex of $\widehat{M}$ where the curvature needs to be approximated. The neighbor points of the origin are $\widehat{p_i} = \widehat{\mathbf{S}}(\mathbf{q}_i), i = 1, \cdots, 4$, where $\mathbf{q}_1 = h(1,2), \mathbf{q}_2 = h(-1,-2), \mathbf{q}_3 = h(-1,2)$ and $\mathbf{q}_4 = h(1,-2)$. The Gaussian curvature of $\widehat{\mathbf{S}}$ at $\mathbf{O}$ is $-B_{11}^2$. By the convergence property of H, we have $\lim_{h\to 0} H(\widehat{M}, p_0; \widehat{p}_1, \widehat{p}_2, \widehat{p}_3, \widehat{p}_4) = -B_{11}^2$. Obviously, by the formulation of $f(x,y)$ and $\widehat{f}(x,y)$, for any h, $p_i = \widehat{p}_i, i = 1, \cdots, 4$. Since the discretization scheme H merely involves one-ring neighbor vertices of p_0, by $p_i = \widehat{p}_i$, $H(\widehat{M}, p_0; \widehat{p}_1, \widehat{p}_2, \widehat{p}_3, \widehat{p}_4) = H(M, p_0; p_1, p_2, p_3, p_4)$ for any h. Hence, $\lim_{h\to 0} H(\widehat{M}, p_0; \widehat{p}_1, \widehat{p}_2, \widehat{p}_3, \widehat{p}_4) = \lim_{h\to 0} H(M, p_0; p_1, p_2, p_3, p_4)$. But $-B_{11}^2$ is not always equal to $4B_{02}B_{20} - B_{11}^2$. Therefore, a contradiction appears. So, the hypothesis with $H(M, p_0; p_1, \cdots, p_n)$ being convergent for any mesh surface does not hold.

Since the mean curvature of $\mathbf{S}(x,y)$ and $\widehat{\mathbf{S}}(x,y)$ at the origin equals to $B_{02} + B_{20}$ and $4B_{02} + B_{20}$ respectively, we can show that one can not construct a discretization scheme of mean curvature converging over any mesh surface by using the similar method with the above,

From above, if merely using one-ring vertices, we can not build discretization schemes of Gaussian curvature and mean curvature converging over any mesh surface. For fixed integer k, using $k-$ring vertices, can we construct a

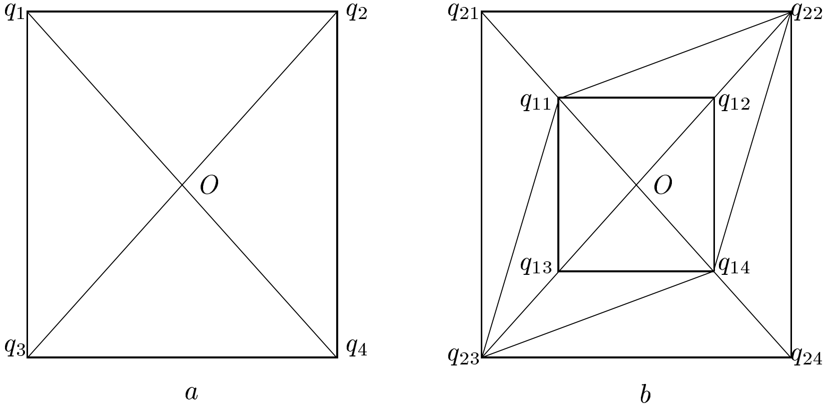


Fig. 1. A counterexample to convergence of curvature and normal

discretization scheme of Gaussian and mean curvature converging over any mesh surface?

Suppose the j-th ring vertices around p_0 is $p_{j,i} = \mathbf{S}(\mathbf{q}_{j,i})$, where $j \leq k, \mathbf{q}_{j,i} = j\mathbf{q}_i, i = 1, \cdots, 4$ (Fig.1.b shows the case where $k = 2$). Obviously, for any h, $S(\mathbf{q}_{j,i}) = \widehat{S}(\mathbf{q}_{j,i}), j \leq k, i = 1, \cdots, 4$. Hence, by using similar method with the above, we can show that for fixed integer k, if only use k-ring vertices, we can not build discretization schemes of mean curvature converging over any mesh surface. It is well known that Laplace-Beltrami operators relates closely to the mean curvature normal. Let p be a surface point on two-dimensional manifold $\mathcal{M}$. Then $\|\Delta_{\mathcal{M}} p\| = 2H(p)$, where $\Delta_{\mathcal{M}} p$ is the Laplace-Beltrami operator and $H(p)$ is the mean curvature at p. Hence, by above results, for fixed integer k, if only k-ring vertices are used, we can not build discretization schemes of Laplace-Beltrami operators converging over any mesh surface.

In [5], a open question is raised: *Find a linear combination of the normals of the triangular faces, based on geometric considerations, that approximates the normal of the surface to $O(h^2)$.* We shall give a negative answer for the open question. Suppose that the vertexes $\mathbf{S}(q_{1i}), i = 1, \cdots, 4$ (see Fig.1.b) are the vertexes of M where the unit normals are to be approximated. We make a hypothesis that there exits a linear combination of the normal of the triangular faces that approximates the normal at $\mathbf{S}(q_{1i})$ to $O(h^2)$. By Lemma 1, under the hypothesis, we can build a discretization schemes involving 2-ring vertexes which approximates the Gaussian curvature at $\mathbf{O}$ to accuracy $O(h)$. The conclusion contradicts to the above results. Hence, the hypothesis does not hold i.e. for any mesh surface, one can not find a linear combination of the normals of the triangular faces that approximates the normal of the surface to $O(h^2)$.

Remark 3. *In this counterexample, the valence of p_0 and p_{11} is 4 and 6 respectively. In fact, by using similar method, we can show one can not build a discretization schemes of Gauss and mean curvature which convergent at p_{11}. Hence, even if the valence of vertexes in mesh surface is bigger than 4, we also can not build convergent discretization schemes of Gauss and mean curvature for any mesh surface.*

Remark 4. *The points p_i or p_{ji} in this counterexample are under-determined for quadratic fit. As pointed out in [5] and [11], if the quadratic fit method has a unique solution, one can approximate the Gaussian curvature and mean curvature to accuracy $O(h)$ and the unit normal to accuracy $O(h^2)$. Then, when the quadratic fit method has a unique solution, can one find a convergence discretization scheme of Gaussian and mean curvature depending only on edge length, angles, and areas of triangle faces? We conjecture that the answer to the question is negative. If the conjecture holds, quadratic fit methods can be the most general method for finding convergent discretization schemes for geometry operators.*

Acknowledgements. The work of the first author is supported by NSFC under grant 10401021 and China Postdoctoral Science Foundation. The work of the second author is supported in part by NSFC under grant 10371130, National 973 Project under grant 2004CB318006.

References

1. M. Desbrun, M.Meyer, P.Schröder and A.H.Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, SIGGRAPH99, 317–324,1999.
2. G H. Liu, Y.S. Wong, Y.F.Zhang, H. T Loh, Adaptive fairing of digitized point data with discrete curvature, Computer Aided Design, 34 (4), 309–320, 2002.
3. Jean-Louis Maltret, Marc Daniel, Discrete curvatures and applications : a survey, preprint, 2003.
4. U. F. Mayer, Numerical solutions for the surface diffusion flow in three space dimensions. Computational and Applied Mathematics (to appear),2001.
5. D.S. Meek, D.J. Walton, On surface normal and Gaussian curvature approximations given data sampled from a smooth surface, Computer Aided Geometric Design, 17 (2000) 521–543.
6. M. Meyer, M. Desbrun, P. Schroder, A. Barr, Discrete differential-geometry operator for triangulated 2-manifolds, in:Proc. VisMath'02, Berlin, Germany.
7. G. Taubin, A signal processing approach to fair surface design, in SIGGRAPH'95 Proceedings 351–385.
8. C. Wollmann, Estimation of principal curvatures of approximated surfaces, Computer Aided Geometric Design,vol 17, 621–630, 2000.
9. G. Xu, Convergence analysis of a discretization scheme for Gaussian curvature over triangular surfaces, submitted for publication.
10. G. Xu, Convergence of discrete Laplace-Beltrami operator over surfaces, Computers and Mathematics with Applications, 48 (2004), 347–360.
11. G. Xu, Discrete Laplace-Beltrami operators and their convergence, Computer Aided Geometric Design 21 (2004)767-784.

A Marching Method for Computing Intersection Curves of Two Subdivision Solids

Xu-Ping Zhu[1], Shi-Min Hu[1], Chiew-Lan Tai[2], and Ralph R. Martin[3]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2] Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, China
[3] School of Computer Science, Cardiff University, Cardiff, Wales
ralph@cs.cf.ac.uk

Abstract. This paper presents a marching method for computing intersection curves between two solids represented by subdivision surfaces of Catmull-Clark or Loop type. It can be used in trimming and boolean operations for subdivision surfaces. The main idea is to apply a marching method with geometric interpretation to trace the intersection curves. We first determine all intersecting regions, then find pairs of initial intersection points, and trace the intersection curves from the initial intersection points. Various examples are given to demonstrate the robustness and efficiency of our algorithm.

1 Introduction

Subdivision surfaces are defined as the limit of repeated refinement of 3D control meshes using specific subdivision rules [4, 5, 9, 12]. Due to their advantages, such as being able to handle arbitrary topology and ease of coding, they are widely used in computer animation and game engines [16], for example. However, applications of subdivision surfaces to industrial design are still infrequent, one reason being that is difficult to construct complex subdivision models using the usual solid operations. This problem is mainly due to the lack of suitable geometric algorithms for computing intersection curves, offsets, blending, trimming, and Boolean operations.

Some such algorithms do already exist. Litke et al. [11] introduced a new method for trimming subdivision surfaces, which is based on the combined subdivision schemes to guarantee exact interpolation of trim curves. Biermann et al. [3] presented a method for computing approximate results of Boolean operations (union, intersection, difference) for free-form solids bounded by multiresolution subdivision surfaces. Both works cite the problem of computing intersection curves calculation as an open problem. Nasri [13] presented a general framework for intersecting two recursive subdivision surfaces based on divide and conquer methods to process complete surfaces. Instead, our goal is to apply a marching method with geometric interpretation to trace the intersection curves based on

R. Martin, H. Bez, M. Sabin (Eds.): Mathematics of Surfaces 2005, LNCS 3604, pp. 458–471, 2005.

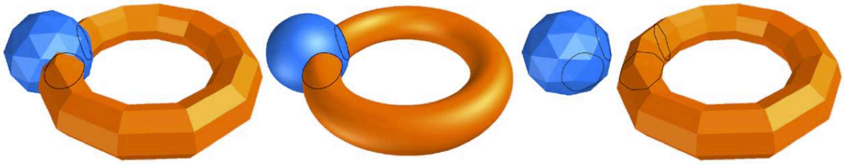


Fig. 1. Intersection curves

an inital intersection point. Grinspun et al. [6] developed an algorithm for detecting interference of subdivision surfaces. He used normal bounds to determine whether a surface interferes with itself or other surfaces. In this paper, we focus on computing all intersection curves between two subdivision surfaces.

Computing intersection curves for parametric and implicit surfaces has been extensively investigated [1, 2, 7, 10]. However, for subdivision surfaces, there are two main difficulties: analytical representation and parameterization. A breakthrough was made by Stam [15, 16], who described an approach for evaluating subdivision surfaces at arbitrary parameter values in the cases of both Catmull-Clark [4] and Loop [12] schemes. Building on the results of his work, we show how traditional algorithms for parametric surfaces can now be applied to subdivision surfaces. The main contribution of this paper is to extend the *moving affine frame* (MAF) marching method [8] to subdivision surfaces. In addition, we present a complete method to trace all intersection curves. Our approach has many applications, to trimming and Boolean operations, for example.

An example using our approach is shown in Figure 1 of the intersection between a Loop sphere and a Catmull-Clark torus. On the left, the two intersection curves are shown together with the given initial control meshes. The result plus both limit surfaces is shown in the centre middle, and the right hand figure gives pre-images of both curves.

The rest of the paper is organized as follows. Section 2 gives a brief review of parameterization of subdivision surfaces. In Section 3, we review the MAF marching method for tracing intersection curves for parametric surfaces, and extend it to subdivision surfaces. Section 4 presents an algorithm for calculating all intersection curves of subdivision surfaces. Section 5 shows various results and conclusions are given in Section 6.

2 Local Parameterization of Subdivision Surfaces

A subdivision surface is defined by an initial control mesh and a set of subdivision rules. As the control mesh is successively refined according to the rules, a sequence of meshes with an increasing numbers of faces is obtained. In the limit, a smooth surface is obtained (Figure 2). Intuitively, the initial control mesh can be considered the domain of the limit surface; each initial face of the control mesh is mapped to a patch on the limit surface. We call these faces the *domain faces* and denote the map as: $(i, u, v) \rightarrow (x, y, z)$ where i indexes the domain

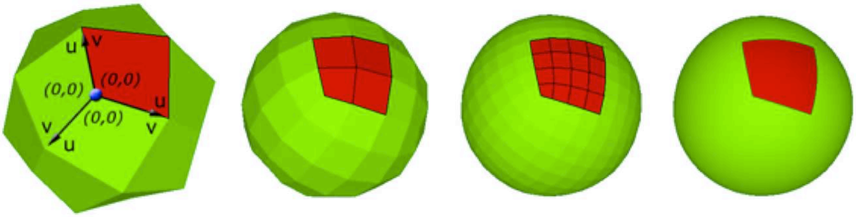


Fig. 2. Mapping a domain face to its corresponding patch. The extraordinary vertex (blue) has parameter (0,0) in all three surrounding extraordinary domain faces

faces, $u, v \in (0, 1)$ are parameter values on the i-th domain face, and (x, y, z) are 3D coordinates of the corresponding limit point on the surface. As in Stams work, we make two assumptions

- The initial control mesh has already been subdivided once so that each domain face of the control mesh contains at most one extraordinary vertex. For a quadrilateral (or triangular) mesh, a vertex having a valence not equal to four (or six, respectively) is called an *extraordinary vertex*. A domain face with an extraordinary vertex is called an *extraordinary domain face*. For example, the blue extraordinary vertex in Figure 2 belongs to three extraordinary domain faces.
- The parameterization is organized so that each extraordinary vertex has parameters $(0, 0)$ in all the extraordinary domain faces containing it.

3 Tracing a Branch of an Intersection Curve

3.1 MAF Marching Method for Parametric Surfaces

We first introduce the *moving affine frame* marching method [8] for parametric surfaces. Assume we are given two parametric surfaces $\mathbf{r}_1$ and $\mathbf{r}_2$, and an initial intersection point $\mathbf{M}$. Let $(u_i, v_i), i = 1, 2$ be the parameter values of $\mathbf{M}$ on each of the two surfaces. The goal is to step along the intersection curve and find the next intersection point. The MAF method gives a stepping size for finding the next intersection point. Let the desired stepping distance be δ. The stepping direction is along the tangent vector to the intersection curve, which is given by $\mathbf{T} = \mathbf{N}_1 \times \mathbf{N}_2$, where $\mathbf{N}_i$ is the normal vector of $\mathbf{r}_i$ at $\mathbf{M}$, i.e. $\mathbf{N}_i = \partial \mathbf{r}_i / \partial u_i \times \partial \mathbf{r}_i / \partial v_i, i = 1, 2$. Thus, the target point is $\mathbf{H} = \mathbf{M} + \delta \mathbf{T}$. Since $\mathbf{H}$ lies on the tangent planes of both surfaces, which are defined by the affine frames $\{\mathbf{M}; \partial \mathbf{r}_i / \partial u_i \partial \mathbf{r}_i / \partial v_i\}, i = 1, 2$, the vector $\mathbf{H} - \mathbf{M}$ can be written as:

$$\mathbf{H} - \mathbf{M} = \delta \mathbf{T} = (\partial \mathbf{r}_i / \partial u_i) \Delta u_i + (\partial \mathbf{r}_i / \partial v_i) \Delta v_i, \quad i = 1, 2 \tag{1}$$

Thus, the parameter increments, $\Delta u_i, \Delta v_i$, can be calculated as using the following triple products:

$$\begin{aligned} \Delta u_i &= [\mathbf{H} - \mathbf{M}, \partial \mathbf{r}_i / \partial v_i, \mathbf{N}_i] / [\partial \mathbf{r}_i / \partial u_i, \partial \mathbf{r}_i / \partial v_i, \mathbf{N}_i] \\ \Delta v_i &= [\mathbf{H} - \mathbf{M}, \partial \mathbf{r}_i / \partial u_i, \mathbf{N}_i] / [\partial \mathbf{r}_i / \partial v_i, \partial \mathbf{r}_i / \partial u_i, \mathbf{N}_i] \end{aligned} \tag{2}$$

With these increments, two new points can be computed as $\mathbf{P}_i = \mathbf{r}_i(u_i + \Delta u_i, v_i + \Delta v_i), i = 1, 2$. This procedure of computing $\mathbf{P}_i$ from a target point $\mathbf{H}$ is called a *sphere transformation*.

If $\mathbf{P}_1$ and $\mathbf{P}_2$ are sufficiently close, i.e. $||\mathbf{P}_1\mathbf{P}_2|| < \epsilon$, the MAF algorithm outputs $(\mathbf{P}_1 + \mathbf{P}_2)/2$ as the next intersection point. Otherwise, it performs a *mid-point transformation* on $\mathbf{P}_i$ as follows. Let π_i be the tangent plane of $\mathbf{r}_i$ at $\mathbf{P}_i$. If $\pi_i, i = 1, 2$ are not parallel, they must intersect in a line l. We project each $\mathbf{P}_i$ to the line l to get new points $\mathbf{R}_i$, and compute the mid-point $\mathbf{S} = (\mathbf{R}_1 + \mathbf{R}_2)/2$.

By applying a sphere transformation to $\mathbf{S}$, we obtain two new points $\mathbf{P}_i$ and repeat the above process of testing $||\mathbf{P}_1\mathbf{P}_2|| < \epsilon$ and, if again failing the test, performing the mid-point transformation, until $||\mathbf{P}_1\mathbf{P}_2|| < \epsilon$.

By repeatedly finding the next intersection point from a given intersection point, we can trace an entire intersection curve. But when should we stop tracing? If the initial surfaces are closed, their intersection curves are also closed. Hence, we use the distance between the current intersection point and the starting point to decide when to stop tracing. If the distance is decreasing and is less than a threshold, such as 2δ, we replace the stepping size δ by a smaller one. If the distance continues to decrease and fall below a smaller threshold, we shorten the stepping size once again and terminate tracing if the distance is less than a prescribed tolerance ϵ. If the initial curves are open, tracing terminates either on forming an intersection loop as above, or when reaching the boundary of either surface. In the latter case we again are using a decreasing step size near the boundary.

3.2 Extending the MAF Method to Subdivision Surfaces

The MAF method is an efficient iterative method with the benefit of a clear geometric interpretation; furthermore it is easy to implement, and it only requires evaluation of points and first-order derivatives. In order to extend the MAF method to subdivision surfaces, we need to resolve two issues:

- How to evaluate the surface and its first-order derivatives efficiently.
- How to find parameter values when updates move outside a domain face, given that we have a local piecewise parameterisation.

The first problem has already been solved by Stam, who has presented an efficient method to evaluate Catmull-Clark and Loop surfaces and all their derivatives at arbitrary parameter values [15, 16]. Hence, we just consider the second problem.

We deal with the quadrilateral mesh case first; we will then look at the triangular mesh case. Assume that the current parameter is (i, u, v). The sphere transformation step computes parameter increments, $\Delta u, \Delta v$, according to a target point $\mathbf{H}$. The new parameter values are then computed as $u = u + \Delta u$, $v = v + \Delta v$. If $u \notin [0, 1]$ or $v \notin [0, 1]$, the parameter will move out of the current domain face into an adjacent domain face. Hence, we must replace the computed parameter (i, u, v) by some (j, u', v'), where j is the index of the target domain face and (u', v') are the new parameter values in that domain face.

If the current domain face is not extraordinary, the target domain face must be one of the eight domain faces around it as depicted in Figure 3a. It is trivial

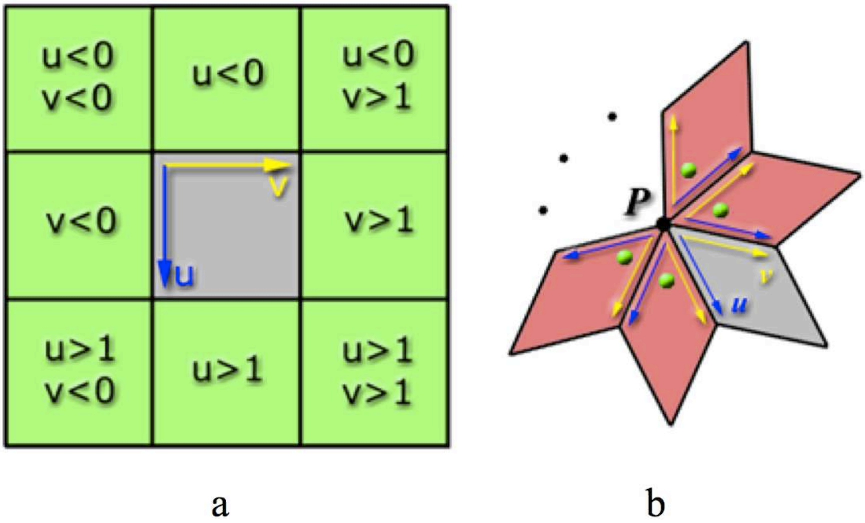


Fig. 3. Identifying parameter values when moving out of the current domain face (quadrilateral case): (a) regular domain face; (b) extraordinary domain face

to find the target domain face from u and v, and to calculate the new parameter values (u', v') according to the u and v directions in that domain face.

If the current domain face is extraordinary, we know that it has only one extraordinary vertex, denoted $\mathbf{P}$. By assumption, $\mathbf{P}$ has parameter $(0, 0)$ in all extraordinary domain faces containing it. Since the other three vertices of the current domain face are all regular, if $u > 0$ or $v > 0$, we can always find a target domain face according to the u and v directions as we do above for the regular case. If $u < 0$ and $v < 0$, the target domain face must be one of the other extraordinary domain faces containing P (red faces in Figure 3b). We need to determine to which in which candidate domain face the parameter values lie, and compute the new parameter values (u', v') in that face. Since $u, v < 0$ and $\mathbf{P}$ has parameter $(0, 0)$ in all domain faces containing it, we can calculate the distance between (u, v) and P in parametric space, $d = \sqrt{u^2 + v^2}$. We estimate the new parameter values by $u' = v' = \sqrt{d^2/2}$ and evaluate the corresponding points at (u', v') for all the candidate domain faces. Finally, we select that point nearest to the target point $\mathbf{H}$ and let its parameter values and domain face determine the new parameter values (j, u', v').

Next, we consider the case of triangular mesh. Assume that the current parameter is (i, v, w), and that for symmetry we add an auxiliary parameter $u = 1 - v - w$. In the sphere transformation step, we obtain the parameter increments as before, $\Delta v, \Delta w$, and compute the new parameter values as $v = v + \Delta v, w = w + \Delta w$. If $v \notin [0, 1]$ or $w \notin [0, 1]$ or $u \notin [0, 1]$, the parametric point lies in an adjacent domain face, and we must find new parameters (j, v', w') as before. The target domain face is one of the domain faces in the 1-ring surrounding the current domain face, i.e. those which share at least one vertex with

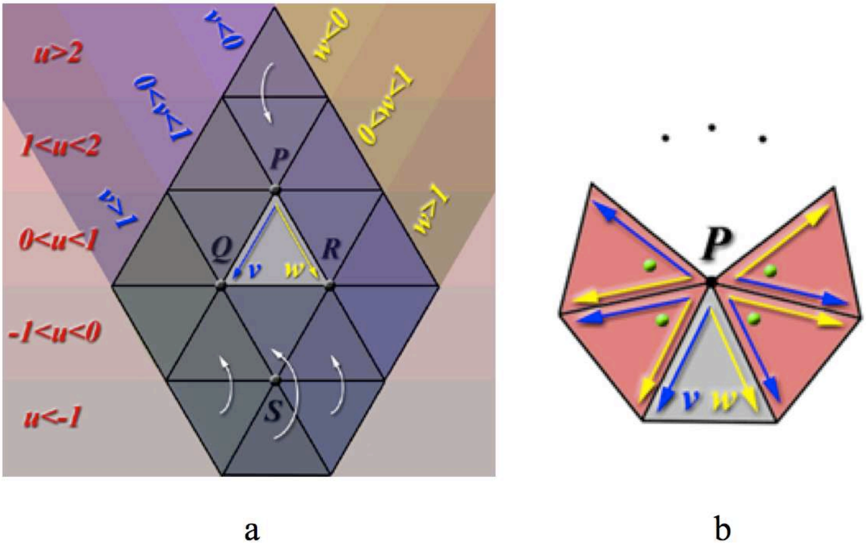


Fig. 4. Identifying parameter values when moving out of the current domain face (triangular case): (a) regular domain face; (b) extraordinary domain face

the current domain face (see Figure 4a). Thus, if $u < -1$ (i.e. $v + w > 2$), we recalculate the values (v, w) using $v = 2v/(v+w), w = 2w/(v+w)$, and if $u > 2$ (i.e. $v + w > -1$), we calculate $v = v/|v + w|, w = w/|v + w|$, so that in each case $-1 \leq u \leq 2$. Let **P** be the parametric point $(0, 0)$ in the current domain face. Regardless of whether **P** is regular or extraordinary, the other two vertices of the current domain face, **Q** and **R**, are both regular. Hence, if $-1 \leq u \leq 1$, we can easily find the target domain face from u, v and w (see Figure 4a) and calculate the new parameter (j, v', w') using the v and w directions in that domain face. For example, if $-1 < u < 0$ and $0 < v, w < 1$, the target domain face is the triangle **QSR**. If $1 < u < 2$, we must consider whether **P** is a regular or extraordinary vertex. If **P** is a regular vertex, we can still find the target domain face easily and calculate the new parameter values. If **P** is extraordinary (see Figure 4b), since $v, w < 0$ and **P** has parameter $(0, 0)$ in all domain faces containing it, we can calculate the distance between (v, w) and **P** in parametric space, $d = \sqrt{\min(1, v^2 + w^2)}$. Next, we compute $v' = w' = \sqrt{d^2/2}$ and evaluate the surface points corresponding to (v', w') for all candidate domain faces. Finally, we select the point nearest to the target point **H** and use it to give the new parameter values (j, v', w').

4 Surface-Surface Intersection Algorithm

As mentioned in Section 2, each patch in the limit surface corresponds to a domain face in the initial mesh. If X denotes the set of all patches for a given subdivision surface, its union gives the limit surface. When a domain face is

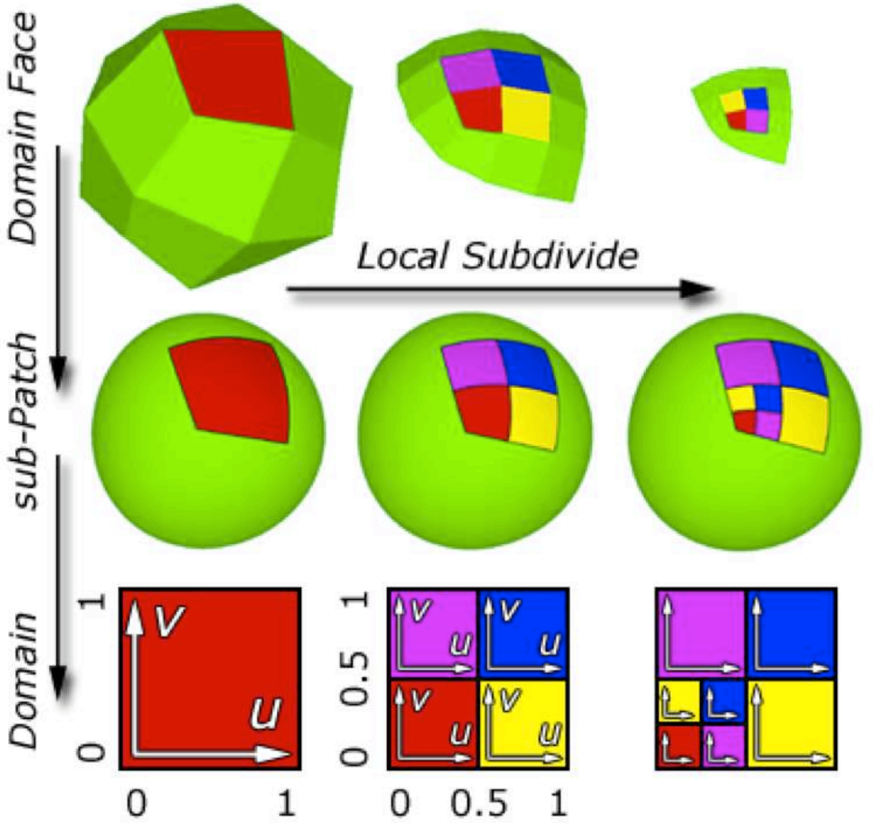


Fig. 5. Subdividing a domain face locally (top row), and splitting its sub-patches accordingly (middle row). The bottom row shows the domains of the sub-patches

subdivided into four sub-faces (using Catmull-Clark or Loop subdivision), the corresponding patch is also split into four sub-patches (see Figure 5). Henceforth, we refer to the elements of X as *sub-patches* (Figure 5, middle row), each having a corresponding sub-face as its domain face (Figure 5, top row), and having a parametric domain that is a subset of its parents parametric domain (Figure 5, bottom row). Additionally, when a face is subdivided locally, in order to be able to continue subdividing its four sub-faces, all the neighbouring faces that share at least a common vertex with those sub-faces (the green faces in the top middle and right of Figure 5) should be computed too.

4.1 Algorithm Overview

We now consider the overall algorithm. It finds starting points for tracing, and then march along the intersection from those starting points.

There are three main steps to this process; in the below, by *interfering*, we mean potentially but not necessarily intersecting:

- Split interfering sub-patches until they are approximately flat, and record all interfering sub-patch pairs.
- Find all pairs of intersecting sub-patches from the interfering pairs, and find an intersection point for each pair.
- Trace all intersection curves from these initial intersection points.

4.2 Convex Hull and Flatness Condition

From the subdivision rules, it can be shown that both Catmull-Clark and Loop surfaces possess the convex hull property, which means that each sub-patch is within the convex hull of its control vertices. A sub-patch has $2N + 8$ control vertices for a quadrilateral mesh and $N + 6$ control vertices in the case of a triangular mesh, where N is the valence of a regular vertex or, if present, of the only extraordinary vertex in the domain face. For simplicity, we use an axis-aligned bounding box (AABB) of the convex hull of each sub-patch as its bounding volume. This is used to detect interference between two sub-patches. If two bounding volumes intersect, we split both sub-patches by subdividing their domain faces, until all sub-patches are considered flat. To measure the flatness of a sub-patch S, we use the variable $f = 1 - \min(\mathbf{N}_0 \cdot \mathbf{N}_i)$, where $\mathbf{N}_0$ is the unit normal of Ss domain face and $\mathbf{N}_i$ is the unit normal of each of the neighbouring 1-ring faces of Ss domain face. (We estimate the normal of a quadrilateral face as the cross product of the two vectors connecting opposite vertices.) When the flatness f is less than a threshold T_f, the sub-patch is considered to be flat.

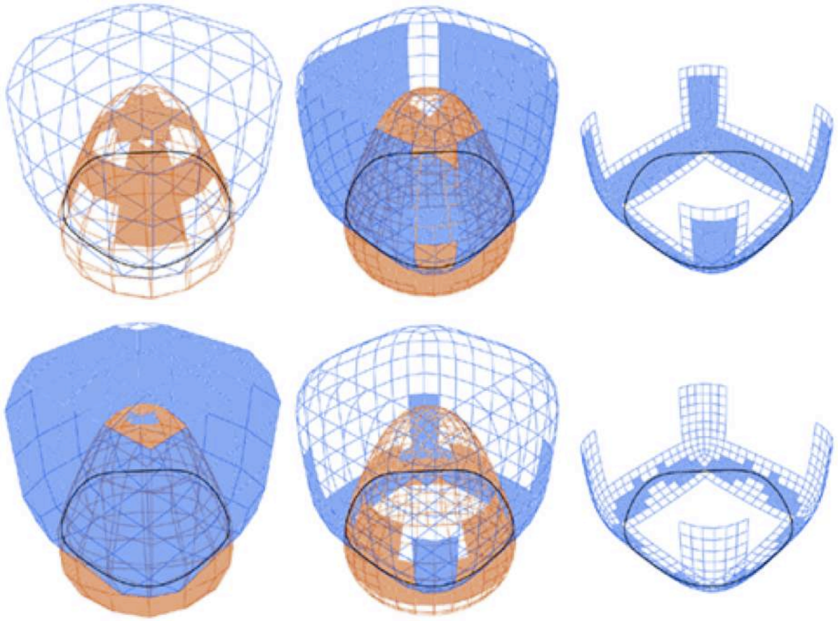


Fig. 6. Flat sub-patches during intersection between a cube and a cone

Additionally, if Ss domain face is quadrilateral, we must first ensure that the domain face is approximately planar by checking if $N_a \cdot N_b < T_f$, where N_a and N_b are the normals of the two triangles comprisong domain face. If the domain face is not planar, we consider the sub-patch not to be flat.

When a sub-patch is sufficiently flat, we take the average parameter values for its parametric domain and evaluate the limit point P_a at the average parameter. Instead of the original bounding volume, we then use the AABB of the convex hull formed by P_a and all the corner points of the sub-patch. We also evaluate the limit normal vector N_a at P_a, to be used as the normal of that sub-patch while finding an actual intersection point, as described in Section 4.3.

By detecting interference and splitting sub-patches, all interfering flat sub-patches are found, as illustrated in Figure 6. This example shows intersection between a cube and a cone. In this case, all interfering sub-patches are flat within tolerance after three levels of subdivision (see Figure 6 from left to right). The top row shows the domain faces of all flat sub-patches, and the bottom row shows the domain faces of all interfering sub-patches.

During this step, for each sub-patch on one surface, we record all the interfering sub-patches from the other surface. Thus, we obtain all pairs of interfering sub-patches. Most of these pairs of interfering sub-patches, however, do not truly intersect. In the next section, we identify those pairs that truly intersect and find an initial intersection point for each pair.

4.3 Intersection Between Two Sub-patches

If the flatness threshold T_f is sufficiently small, we can approximate all sub-patches by polygons and find all intersecting polygon pairs. However, using a small T_f is undesirable, as it leads to more levels of subdivision, incurring considerable cost. Hence, we prefer a patch-patch intersection method.

Inspired by the MAF method, given two sub-patches, we use the average parameters of their respective domains as their initial parameters and iterate the mid-point transformation and sphere transformation to search for an intersection point in their respective domains. If we find an intersection point within a prescribed number of iterations, n, we say that the sub-patch pair intersect. Their intersection point and the sub-patch pair are then added to a set of candidate intersecting pairs. If no intersection point is found after n iterations, it is still possible that the two sub-patches intersect. Such tricky cases are illustrated in Figure 7 for the simpler problem of curve-curve intersection in the plane: if we perform mid-point transformation on the two initial points (blue), both cases will search for the intersection point in the wrong direction and miss the intersection point in the domain of each curve.

To avoid such cases, it would suffice to ensure that for any point of the first curve, it is impossible to find a point in the second curve with parallel normal vector. But for surfaces, the above condition cannot eliminate such cases. Since our purpose is to approximately find initial intersection points for tracing, we simply try to avoid these cases by taking a hint from the following observation: in practice, such cases usually appear when the normal vectors of two sub-patches

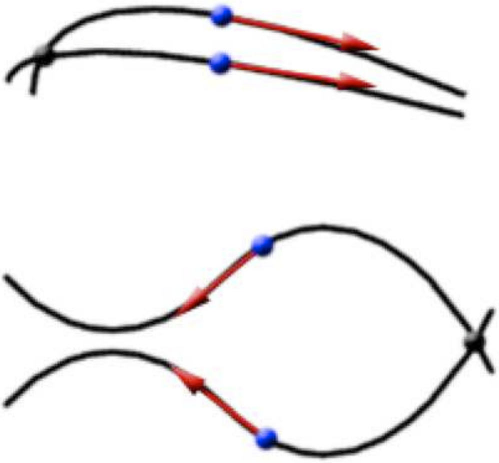


Fig. 7. Two cases which miss the intersection point when doing midpoint transformation

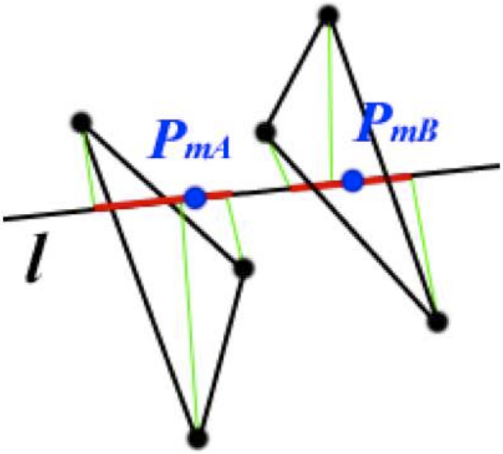


Fig. 8. Interference detection between two sub-patches

are almost parallel. Thus, before checking whether two sub-patches intersect, we estimate if their normals, $\mathbf{N}_{m_1}, \mathbf{N}_{m_2}$ (computed as in Section 4.2), are almost parallel as follows. Let $m = 1 - |\mathbf{N}_{m_1} \cdot \mathbf{N}_{m_2}|$. If $m < f_1 + f_2$, where f_1, f_2 are the flatnesses of the two sub-patches, we say that $\mathbf{N}_{m_1}$ and $\mathbf{N}_{m_2}$ are approximately parallel. If so, we use a tighter bounding volume to more strictly decide interference as follows. Consider two sub-patches A and B. If they are flat, their corner points and two further points P_{m_A}, P_{m_B} would have been calculated as described in Section 4.2. We project all the corner points to the line l connecting P_{m_A} and P_{m_B}, and obtain two maximal intervals on l (shown in red in Figure 8).

If the intervals do not overlap, then A and B do not intersect. Otherwise we split the sub-patch whose domain face has a lower subdivision level and recursively check if the resulting new pairs of sub-patches intersect. If both domain faces has reached the maximum subdivision level, since they are very close to each other and their normal vectors are almost parallel, we conclude that the two subdivision surfaces sharing a region. Perturbation schemes [14] may be used to resolve such degeneracies. For reasons of space, we do not consider this problem further here, and instead focus on presenting the efficient marching method.

We now have all the intersecting sub-patch pairs and an initial intersection point for each. Next, we trace all intersection curves.

4.4 Tracing

We now have all intersecting sub-patch pairs in the candidate set. For each pair, both sub-patches are flat and their normals are not parallel. Hence, we can reasonably assume that each intersecting sub-patch pair has only one intersection curve. Based on this assumption, we present the following tracing algorithm.

First, we randomly select an intersecting sub-patch pair from the candidate set and invoke the MAF procedure with the intersection point of the pair. When the algorithm marches to a new intersection point, it obtains a parameter for each of the two surfaces. For each parameter, we then find the sub-patch whose domain contains that parameter. This new sub-patch pair should also appear in the candidate set. Since there is only one intersection curve for each sub-patch pair, we mark that new sub-patch pair with the index of the current intersection curve and remove it from the candidate set. After tracing a curve, all the sub-patch pairs it has encountered by will have been marked, and thus will not be selected for initiating further curve tracing. If the step size is too large, the tracing curve may skip some pairs. However, tracing from these skipped pairs would result in a situation where most sub-patch pairs on the current curve already belong to another curve; the two curves obviously are identical and thus tracing is terminated. Starting points are selected from the candidate set and each time a corresponding intersection curve is traced. This is repeated until the candidate set is empty.

While we calculate all intersection curves between the two subdivision surfaces, we also record the parameter values of all the intersection points to obtain the pre-images of the intersection curves.

4.5 Parameter Settings

First, we consider the step size δ. We would like $|\delta u|$ and $|\delta v|$ both to be lass than 1 in the sphere transformation step. To achieve this, we estimate the step size according to the scale of the two meshes and adaptively adjust it according to the domain face containing the current tracing point.

In the first step of our algorithm, a sub-patch is considered to be sufficiently flat when its flatness is less than a threshold: $f < T_f$. We find a choice of $T_f = 0.1$ works well in practice. Usually, all sub-patches are found to be flat after two or three levels of subdivision, and for sub-patches with large curvature, six levels are sufficient. Hence, we conservatively set the maximum subdivision level to be 10. In the second step, if two sub-patches really intersect, in practice we always find an intersection point within three iterations; again to be safe, we set the number of iterations $n = 5$. Since most intersecting pairs do not actually intersect, the average number of iterations used is close to 5.

4.6 Remarks

Since our method is an extension of a known marching method for parametric surfaces to subdivision surfaces, we can handle all special cases that it can also cope with for parametric surfaces. For the same reason, our method faces the same kinds of degeneracy problems as do traditional surface-surface intersection methods.

5 Results

We show some results of calculating intersection curves between subdivision solids using our algorithm. All the examples in Figure 9 were calculated within 10 seconds on a PC with 900 MHZ CPU. More complex examples are shown in Figure 10.

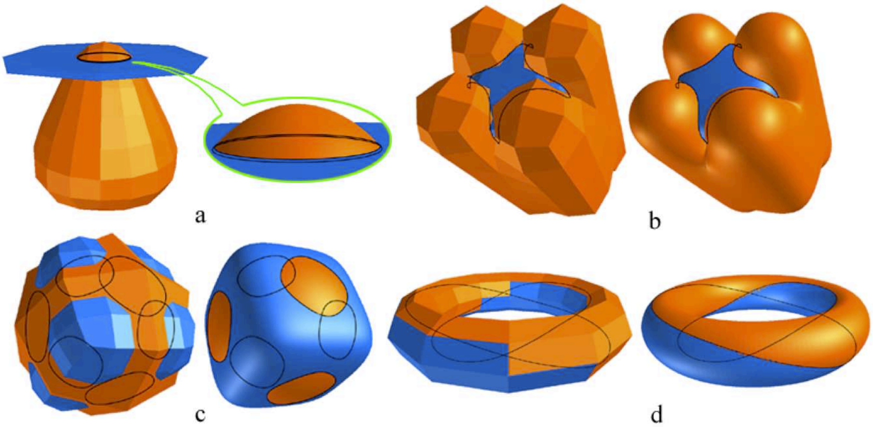


Fig. 9. Intersection curves between two subdivision solids

6 Conclusions

In this paper we have presented an efficient method to calculate intersection curves between two subdivision solids. We continued earlier work and have made a contribution that will extend subdivision surfaces to new applications. Our ultimate intention is to construct complex models from simple primitives using solid modelling operations, and so future works will include:

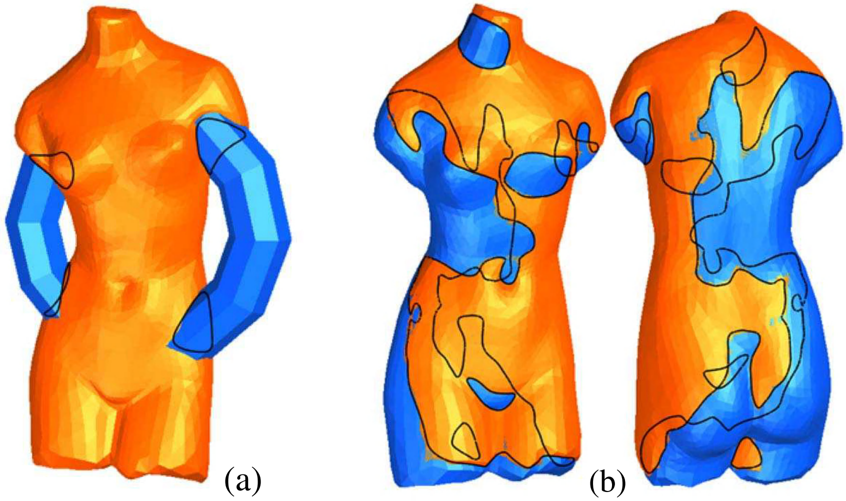


Fig. 10. (a) Intersecting a Venus (Loop surface) and a torus (Catmull-Clark) surface; (b) intersecting two Venuses (Venus model courtesy of the NYU Media Research Lab)

- Designing more robust algorithms that can better handle degeneracies and numerical instabilities.
- Extending our method to *piecewise* smooth subdivision solids using Zorin's work as a basis [18].
- Developing a CAD system based on subdivision and CSG representation by integrating a wide range of geometric computing algorithms, such as offseting, blending, trimming and Boolean operations.

Acknowledgements

This work was supported by the Natural Science Foundation of China (Project numbers 60225016, 60273012, 60333010).

References

1. R. E. Barnhill, G. Farin, M. Jordan, B. R. Piper (1987) Surface/surface intersection. *Computer Aided Geometric Design*, 4, 3–6
2. R. E. Barnhill, S. N. Kersey (1990) A marching method for parametric surface/surface intersection. *Computer Aided Geometric Design*, 17, 257–280
3. H. Biermann, D. Kristjansson, D. Zorin (2001) Approximate Boolean operations for subdivision surfaces. *Proc. SIGGRAPH 2001*, 185–194
4. E. Catmull, J. Clark (1978) Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10 (6), 350-?55
5. D. Doo, M. Sabin (1978) Analysis of the behavior of recursive division surfaces near extraordinary points, *Computer Aided Design*, 10 (6), 257–268

6. E. Grinspun, P. Schröder (2001) Normal bounds for subdivision-surface interference detection. *Proc. IEEE Scientific Visualization*
7. M. Hohmeyer (1991) A surface intersection algorithm based on loop detection. *International Journal of Computational Geometry and Applications*, 1 (4), 473–490
8. S. M. Hu, J. G. Sun, T. G. Jin, G. Z. Wang (2000) Computing the parameters of points on NURBS curves and surfaces via moving affine frame method. *Journal of Software*, 11 (1), 49–53
9. L. Kobbelt (2000) $\sqrt{3}$ subdivision. *SIGGRAPH 2000 proceedings*, 103–112
10. S. Krishnan, D. Manocha (1997) An efficient surface intersection algorithm based on lower dimensional formulation. *ACM transactions on Graphics* 16 (1)
11. N. Litke, A. Levin, P. Schröder (2000) *Trimming for Subdivision Surfaces.* Technical report, Caltech
12. C. T. Loop (1987) *Smooth Subdivision Surfaces Based on Triangles.* M.S. Thesis, Department of Mathematics, University of Utah
13. A. H. Nasri (1987) Polyhedral Subdivision Methods for Free-form Surfaces *ACM Trans. Graphics* 6 (1), 29–73
14. R. Seidel (1998) The nature and meaning of perturbations in geometric computing. *Discrete and Computational Geometry*, 19 (1), 1–17
15. J. Stam (1998) Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Proc. SIGGRAPH 1998*, 395–404
16. J. Stam. (1999) Evaluation of Loop subdivision surfaces. *SIGGRAPH'99 Course Notes*
17. D. Zorin, P. Schröder (2000) Subdivision for modeling and animation, *SIGGRAPH 2000 Course Notes*
18. D. Zorin, D. Kristjansson (2001) Evaluation of piecewise smooth subdivision surfaces. *Visual Computer*

Author Index