Edwin R. Hancock
Ralph R. Martin
Malcolm A. Sabin (Eds.)

LNCS 5654

Mathematics of Surfaces XIII

13th IMA International Conference
York, UK, September 2009
Proceedings

Springer

Edwin R. Hancock Ralph R. Martin
Malcolm A. Sabin (Eds.)

Mathematics of Surfaces XIII

13th IMA International Conference
York, UK, September 7-9, 2009
Proceedings

Springer

Volume Editors

Edwin R. Hancock
University of York
Department of Computer Science
Heslington, York YO10 5DD, UK
E-mail: erh@cs.york.ac.uk

Ralph R. Martin
Cardiff University
School of Computer Science
Roath, Cardiff CF24 3AA, UK
E-mail: ralph.martin@cs.cardiff.ac.uk

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

Library of Congress Control Number: 2009931689

CR Subject Classification (1998): F.2.2, I.3.5, I.4.1, I.1, J.6, G.1, F.2, I.4

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743
ISBN-10	3-642-03595-7 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-03595-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12731565 06/3180 5 4 3 2 1 0

Preface

This volume collects the papers accepted for presentation at the 13th IMA Conference on the Mathematics of Surfaces, held at the University of York, UK September 7–9, 2009. Contributors to this volume include authors from many countries in America, Asia, and Europe. The papers presented here reflect the applicability of mathematics of surfaces to engineering and computer science, especially in domains such as computer-aided design, computer vision, and computer graphics.

The papers in the present volume include seven invited papers, as well as a larger number of submitted papers. They cover a range of ideas from underlying theory of surfaces to practical tools, and industrial applications of surfaces. Surface types considered encompass polygon meshes as well as parametric and implicit surfaces. Topics providing a theoretical basis include subdivision schemes and their continuity, polar patchworks, compressive algorithms for PDEs, surface invariant functions, swept volume parameterization, Willmore flow, computational conformal geometry, heat kernel embeddings, and self-organizing maps on manifolds. Toward the practical and applied end of the scale, papers cover such issues as mesh and manifold construction, editing, flattening, morphing and interrogation, dissection of planar shapes, symmetry processing, morphable models, computation of isophotes, point membership classification and vertex blends.

We would like to thank all those who attended the conference and helped to make it a success, especially the keynote speaker, the renowned Field's Medallist Shing-Tung Yau from Harvard University, as well as the other invited speakers whose contributions were a highlight of the meeting.

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. Many of the papers have been improved as a result of their comments. Our thanks go to the International Programme Committee, and to other people upon whom they called to help with refereeing.

We are also grateful to Amy Marsh at the Institute of Mathematics and its Applications for her hard work in organizing many aspects of the conference, and to Anna Kramer and Springer for their help in publishing this volume.

June 2009

Edwin Hancock
Ralph Martin
Malcolm Sabin

Organization

The Mathematics of Surfaces XIII Conference was organized by the Institute of Mathematics and its Applications (Southend-on-Sea, Essex, UK).

Organizing Committee

Edwin Hancock	University of York, UK
Ralph Martin	Cardiff University, UK
Malcolm Sabin	Numerical Geometry Ltd., UK

Invited Speakers

Shing-Tung Yau	Harvard University, USA
Freddy Bruckstein	Technion, Israel
Massimo Fornasier	Johann Radon Institute for Computational and Applied Mathematics, Austria
Tom Funkhauser	Princeton University, USA
Bert Jüttler	Johannes Kepler University, Austria
Konrad Polthier	Free University of Berlin, Germany
Thomas Vetter	University of Basel, Switzerland

Programme Committee

Pierre Alliez	INRIA Sophia Antipolis, France
Chandrajit Bajaj	University of Texas at Austin, USA
Alexander Belyaev	Heriot-Watt University, UK
Bob Cripps	University of Birmingham, UK
Gershon Elber	Technion, Israel
Gerald Farin	Arizona State University, USA
Rida Farouki	University of California, Davis, USA
Bob Fisher	University of Edinburgh, UK
Mike Floater	University of Oslo, Norway
Leila de Floriani	University of Genova, Italy
Xiao-Shan Gao	Chinese Academy of Sciences, China
Peter Giblin	University of Liverpool, UK
Ron Goldman	Rice University, USA
Laureano Gonzalez-Vega	University of Cantabria, Spain
Xianfang Gu	State University of New York, Stony Brook, USA
Hans Hagen	University of Kaiserslautern, Germany
Kai Hormann	Technical University of Clausthal, Germany

Shimin Hu	Tsinghua University, China
Deok-Soo Kim	Hanyang University, Korea
Myung-Soo Kim	Seoul National University, Korea
Tom Lyche	University of Oslo, Norway
Bernard Mourrain	INRIA Sophia Antipolis, France
Alberto Paoluzzi	University of Rome 3, Italy
Nick Patrikalakis	MIT, USA
Xavier Pennec	INRIA Sophia Antipolis, France
Sonia Pérez-Díaz	University of Alcalá, Spain
Jorg Peters	University of Florida, USA
Helmut Pottmann	Technical University of Vienna, Austria
Hong Qin	State University of New York, Stony Brook, USA
Christophe Rabut	INSA Toulouse, France
Ulrich Reif	Technical University of Darmstadt, Germany
Martin Rumpf	University of Bonn, Germany
Paul Sablonnière	INSA Rennes, France
Hanan Samet	University of Maryland, USA
Will Smith	University of York, UK
Hiromasa Suzuki	University of Tokyo, Japan
Georg Umlauf	University of Kaiserslautern, Germany
Luiz Velho	IMPA, Brazil
Wenping Wang	University of Hong Kong, China
Joe Warren	Rice University, USA
Mike Wilson	University of Leeds, UK
Guoliang Xu	Chinese Academy of Sciences, China

Table of Contents

Computing Isophotes on Free-Form Surfaces Based on Support Function Approximation

M. Aigner[1], L. Gonzalez-Vega[2], B. Jüttler[1], and M.L. Sampoli[3]

[1] Johannes Kepler University, Linz, Austria
[2] University of Cantabria, Santander, Spain
[3] University of Siena, Italy

Abstract. The support function of a free-form-surface is closely related to the implicit equation of the dual surface, and the process of computing both the dual surface and the support function can be seen as dual implicitization. The support function can be used to parameterize a surface by its inverse Gauss map. This map makes it relatively simple to study isophotes (which are simply images of spherical circles) and offset surfaces (which are obtained by adding the offsetting distance to the support function).

We present several classes of surfaces which admit a particularly simple computation of the dual surfaces and of the support function. These include quadratic polynomial surfaces, ruled surfaces with direction vectors of low degree and polynomial translational surfaces of bidegree (3,2).

In addition, we use a quasi-interpolation scheme for bivariate quadratic splines over criss-cross triangulations in order to formulate a method for approximating the support function. The inverse Gauss maps of the bivariate quadratic spline surfaces are computed and used for approximate isophote computation. The approximation order of the isophote approximation is shown to be 2.

1 Introduction

This paper is devoted to the use of support functions and dual implicitization in order to deal with the problem of computing isophotes of free form surfaces. Like reflection lines and highlight lines, isophotes are very useful tools for shape interrogation, see [11,12]. They are defined as lines of equal light intensity and they are used to detect and visualize small surface irregularities and discontinuities that can not be seen by other means such as, for example, a shaded surface image.

The first use of isophotes in this context is due to [22]. Isophotes appear also in other contexts such as Image Processing for the so called image interpolation (e.g. [18,28]), Computer Vision for object detection (e.g., [17]) or in the study of feature sensitive mathematical morphology of surfaces where an isophotic metric has been introduced [25].

Dual implicitization (or support function computation) provides an alternative way to represent curves and surfaces that allows, in many cases, to analyze and to manipulate the considered curves and surfaces [9,26]. In this paper we show several concrete cases (quadratic polynomial surfaces, ruled surfaces with polynomial direction vectors of low degree, polynomial translational surfaces of low degree or some special cubics) where the dual implicit equation is very easy to compute explicitly in terms of rational functions and square roots. Using dual implicitization and the inverse Gauss map, it is then simple to characterize the isophotes for these cases.

In order to compute the isophotes on free form surfaces in the general case we propose to compute a support function approximation through the computation of a piecewise quadratic approximation for the considered free form surface. This support function then defines the inverse Gauss maps which are used to determine the isophotes for the considered free form surface.

The remainder of this paper is organized as follows. After recalling the notions of dual surfaces and support functions of free-form surfaces, the second section discusses the support functions of several particular classes of surfaces: quadratic polynomial surfaces, rational ruled surfaces with direction vectors of low degree, polynomial translational surfaces of low degree, and special cubic polynomial surfaces. Section 3 uses the support function to analyze offset surfaces, isophotes and contour generators. Based on a simple representation for the inverse Gauss map, it presents results on exact rational paramaterizations of offsets and on parametric represetations of isophotes, in both cases for special classes of surfaces. Section 4 discusses the case of general free-form surfaces. We propose to approximate the support function via quasi-interpolation by piecewise quadratic surfaces and use the result for approximate isophote computation. Finally we analyze the convergence of the method.

2 Dual Implicitization

We recall the notions of the dual surface and the support function of a rational surface. In the second part we discuss several classes of surfaces whose support function can be expressed by using solely square roots and rational functions.

2.1 Dual Surface and Support Functions

We consider a non-developable polynomial or rational surface $\mathbf{p} : \Omega \to \mathbb{R}^3$ with domain $\Omega \subseteq \mathbb{R}^2$. Its partial derivative vectors with respect to the surface parameters u, v will be denoted with $\mathbf{p}_u$ and $\mathbf{p}_v$, respectively.

Consider the three equations

$$h - \mathbf{p}(u,v)^\top \mathbf{n} = 0, \tag{1}$$

$$\mathbf{p}_u(u,v)^\top \mathbf{n} = 0, \tag{2}$$

$$\mathbf{p}_v(u,v)^\top \mathbf{n} = 0. \tag{3}$$

If these equations are satisfied by non-trivial values of h and $\mathbf{n}$, then the plane with normal $\mathbf{n}$ and distance $h/||\mathbf{n}||$ to the origin is the tangent plane of the surface at its point $\mathbf{p}(u,v)$.

This system is homogeneous with respect to h and $\mathbf{n}$. Eliminating u and v from (1)–(3) leads to a single equation of the form

$$F(\mathbf{n}, h) = \sum_{i=0}^{d} h^i f_{m-i}(\mathbf{n}) = 0 \tag{4}$$

for certain values of the total degree m and the degree d with respect to h. This equation is homogeneous of degree m with respect to h and $\mathbf{n}$, since the original equations (1)–(3) were homogeneous. Consequently, the functions f_{m-i} are homogeneous polynomials of degree $m - i$ in $\mathbf{n}$.

Eq. (4) is the dual implicit equation of the surface $\mathbf{p}$, as it is satisfied by the coefficients of the tangent planes of the surface. We refer to the process of computing this equation as *dual implicitization*.

In order to visualize it, one may consider the surface

$$F((x, y, z)^\top, 1) = 0 \tag{5}$$

which is obtained by substituting $h = 1$ and $\mathbf{n} = (x, y, z)^\top$. This is the *dual surface* (see [13]), which is generated by applying the polarity with respect to the unit sphere to the tangent planes of $\mathbf{p}$.

Hoschek [13] uses the polarity with respect to the imaginary unit sphere. We shall use the standard unit sphere instead, see e.g. [10]. In this case, a point $\mathbf{q}$ is mapped to the plane $\mathbf{q} \cdot \mathbf{x} = 1$ and vice versa, and the center of the sphere is mapped to the plane at infinity.

We are particularly interested in surfaces where it is possible to solve the two equations (2)–(3) for u, v by using rational operations and square roots. After substituting the result into (1), the variable h can be expressed by a function

$$h = H(\mathbf{n}) \tag{6}$$

of $\mathbf{n}$. The function H may possess different branches, if square roots are involved. For a given normal $\mathbf{n}$, the different roots of $F(\mathbf{n}, h) = 0$ are the different possible values of H.

The function H is a 1-homogeneous function, i.e.

$$H(\lambda \mathbf{n}) = \lambda\, H(\mathbf{n}), \tag{7}$$

since the elimination procedure preserves the homogeneity of the original system. The equation

$$H(\mathbf{n}) = 1 \tag{8}$$

is another implicit representation of the dual surface (5).

The restriction of H to the unit sphere is the *support function* of the surface. This function is always odd,

$$H(-\mathbf{n}) = -H(\mathbf{n}), \tag{9}$$

since H is homogeneous. The value(s) of $H(\mathbf{n})$ is (are) the support distance(s) of the tangent plane(s) with unit normal $\mathbf{n}$ (cf. see [9,10,26]).

We describe several classes of surfaces leading to certain special values of the degrees d and m which admit closed form solutions of (4) with respect to h.

2.2 Quadratic Polynomial Surfaces

We recall results from [10, Example 2] on the support function of non-developable quadratic polynomial surfaces (triangular Bézier surfaces of degree two); see also [16]. These surfaces possess an affine classification, which is described in [4,21]. We are mainly interested in non-developable surfaces. Developable quadratic polynomial surfaces are planes, parabolic cylinders or quadratic cones.

If the components of $\mathbf{p}$ are quadratic polynomials in u and v, then—in the generic case—the dual implicit equation has the form (4) with $d = 1$ and $m = 3$. Consequently, the support function H is an odd rational function

$$H = -f_3(\mathbf{n})/f_2(\mathbf{n}) \tag{10}$$

which is the quotient of two homogeneous polynomials f_3 and f_2 of degree 3 and degree 2, respectively. The dual surface is a cubic monoid surface (see [15]) with a threefold point at the origin.

2.3 Ruled Surfaces with Polynomial Direction Vectors of Low Degree

Next we consider ruled surfaces,

$$\mathbf{p}(u, v) = \mathbf{q}(u) + v\mathbf{w}(u) \tag{11}$$

where the directrix $\mathbf{q}(u)$ is a rational curve and the direction vector of the generators are described by a polynomial function

$$\mathbf{w}(u) = \sum_{i=0}^{k} u^i\, \mathbf{w}_i \tag{12}$$

with certain real coefficient vectors $\mathbf{w}_i \in \mathbb{R}^3$. The two equations (2), (3) take the form

$$[\mathbf{q}'(u) + v\mathbf{w}'(u)]^\top \mathbf{n} = 0 \quad \text{and} \quad \mathbf{w}(u)^\top \mathbf{n} = 0. \tag{13}$$

We analyze the cases $k = 1$ and $k = 2$.

- $\underline{k = 1}$: The two equations (13) are linear with respect to v and u, respectively, and the second equation does not involve v. One may easily express both u and v by rational functions of $\mathbf{n}$.

  The dual implicit equation takes the form (4) with $d = 1$. The support function of the surface is an odd rational function.
- $\underline{k = 2}$: The first equation is linear in v and it can be used to express it as a rational function of $\mathbf{n}$ and u. The second equation is a quadratic equation for u, which can be solved by using one square root. Note that its argument is a homogeneous quadratic form of $\mathbf{n}$.

  The dual implicit equation takes the form (4) with $d = 2$. The support function of the surface can be expressed as

  $$H(\mathbf{n}) = R(\mathbf{n}, \pm\sqrt{\mathbf{n}^\top D\mathbf{n}}) \tag{14}$$

  where R is a homogeneous rational function and D is a symmetric 3×3 matrix.

Consequently, rational ruled surfaces with linear and quadratic direction vectors are special instances of the surfaces studied in [10] and [1], respectively.

2.4 Polynomial Translational Surfaces of Low Degree

A translational surface is generated by translating a moving curve along a fixed curve. The general parametric representation of a rational translational surface is

$$\mathbf{p}(u,v) = \mathbf{q}(u) + \mathbf{r}(v) \tag{15}$$

where $\mathbf{q}$ and $\mathbf{r}$ are rational curves. The last two equations (2) and (3) take the form

$$\mathbf{q}'(u)^\top \mathbf{n} = 0 \quad \text{and} \quad \mathbf{r}'(v)^\top \mathbf{n} = 0. \tag{16}$$

We analyze the special case where both $\mathbf{q}$ and $\mathbf{r}$ are polynomial curves of low degree (at most cubic). We assume that $\mathbf{q}$ is a cubic curve and study the cases of quadratic and cubic curves $\mathbf{r}$ separately.

- $\underline{\mathbf{r}\text{ is quadratic:}}$ The first equation in (16) is quadratic with respect to u and the coefficients are linear in the coordinates of $\mathbf{n}$. The second equation (16) is even just linear in v, hence u and v can immediately be computed from $\mathbf{n}$ using rational functions and one square root.
  The dual implicit equation takes the form (4) with $d = 2$ and $m = 6$. The support function of the surface can be expressed as

  $$H(\mathbf{n}) = R(\mathbf{n}, \pm\sqrt{\mathbf{n}^\top D\mathbf{n}}) \tag{17}$$

  where R is a homogeneous rational function with a numerator of degree 4 and a denominator of degree 3, and D is a symmetric 3×3 matrix. Consequently, polynomial translational surfaces of degree (3,2) are special instances of the surfaces studied in [1].
- $\underline{\mathbf{r}\text{ is cubic:}}$ Both equations in (16) are now quadratic. The parameters u and v can immediately be computed from $\mathbf{n}$ using rational functions and two square roots.
  The dual implicit equation takes the form (4) with $d = 4$ and $m = 12$. The support function of the surface can be expressed as

  $$H(\mathbf{n}) = R(\mathbf{n}, \pm\sqrt{\mathbf{n}^\top D_1\mathbf{n}}, \pm\sqrt{\mathbf{n}^\top D_2\mathbf{n}}). \tag{18}$$

  It is a homogeneous rational function with a numerator of degree 5 and a denominator of degree 4, and D_1 and D_2 are two symmetric 3×3 matrices.

Example 1. The computation of the support function of the translational surface

$$\mathbf{p}(u,v) = (u, v, u^2 + v^3)^\top \tag{19}$$

which is generated by translating the cubic parabola along the quadratic parabola gives

$$H(\mathbf{n}) = \frac{1}{36}\,\frac{-9\,x^2 \pm 8\,\sqrt{3}\sqrt{-zy}\,y}{z} \tag{20}$$

where $\mathbf{n} = (x,y,z)^\top$.

2.5 Special Cubic Polynomial Surfaces

Finally we mention two special classes of cubic surfaces whose support function can be computed explicitly simply by using square roots. A general polynomial cubic surface has a parametric representation of the form

$$\mathbf{p}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3-i} \mathbf{a}_{i,j} u^i v^j \tag{21}$$

with real coefficients $\mathbf{a}_{i,j} \in \mathbb{R}^3$. If the coefficients satisfy either

$$\mathbf{a}_{2,1} = \mathbf{a}_{2,0} = \mathbf{a}_{1,1} = \mathbf{a}_{0,2} = \mathbf{a}_{0,1} = 0 \tag{22}$$

or

$$\mathbf{a}_{2,1} = \mathbf{a}_{1,2} = \mathbf{a}_{0,3} = 0, \tag{23}$$

then the two equations (2), (3) can be used to express u and v as functions of $\mathbf{n}$ using rational functions and square roots. Consequently, the support functions of these surfaces involve only rational expressions and square roots, but the arguments of the roots are now polynomials with a higher degree than two.

3 Offsets, Isophotes and Contour Generators

We discuss the inverse Gauss map of surfaces with odd support functions and address the existence and computation of rational PN (Pythagorean normal) parameterizations. Finally we analyze the parameterization of isophotes with the help of the inverse Gauss map.

3.1 The Inverse Gauss Map

Any 1-homogeneous function H satisfies

$$\nabla H(\lambda \mathbf{n}) = \nabla H(\mathbf{n}), \tag{24}$$

i.e., the gradient field is *constant along the lines through the origin.* Indeed, applying the gradient operator to (7) gives (24), where both sides of the equation are multiplied by λ.

For any point $\mathbf{n}$, the scaled point $\mathbf{n}_0 = \mathbf{n}/H(\mathbf{n})$ lies on the dual surface, since it satisfies

$$H(\mathbf{n}_0) = H\left(\frac{\mathbf{n}}{H(\mathbf{n})}\right) = \frac{1}{H(\mathbf{n})} H(\mathbf{n}) = 1, \tag{25}$$

see (8). Moreover the normal vector of the dual surface at $\mathbf{n}_0$ is $(\nabla H)(\mathbf{n})$, due to (24). The equation of the tangent plane of the dual surface at $\mathbf{n}_0$ is

$$(\nabla H)(\mathbf{n}) \cdot \mathbf{x} = 1 \tag{26}$$

since

$$(\nabla H)(\mathbf{n}) \cdot \frac{\mathbf{n}}{H(\mathbf{n})} = \frac{1}{H(\mathbf{n})} \frac{\mathrm{d}}{\mathrm{d}t} H((1+t)\mathbf{n})\bigg|_{t=0} = \frac{1}{H(\mathbf{n})} \frac{\mathrm{d}}{\mathrm{d}t}(1+t)\bigg|_{t=0} H(\mathbf{n}) = 1.$$

Now we apply the polarity with respect to the unit sphere to the tangent plane ([26]). This leads to the corresponding point $\nabla H(\mathbf{n})$ of the original surface, such that the vector $\mathbf{n}$ is a normal vector of the surface. Consequently, the mapping

$$\Gamma : \quad \mathbb{S}^2 \to \mathbb{R}^3 : \quad \mathbf{n} \to (\nabla H)(\mathbf{n}) \tag{27}$$

is the *inverse Gauss map* of the surface with the 1-homogeneous support function H, see [9,10].

Example 2. The inverse Gauss maps of the translational surface from Example 1 are

$$\Gamma(\mathbf{n}) = \left(-\frac{x}{2z}, -\frac{\sqrt{3}y}{\pm 3\sqrt{-zy}}, \frac{\pm 9\sqrt{-zy}x^2 + 4\sqrt{3}zy^2}{\pm 36z^2\sqrt{-zy}} \right)^\top, \tag{28}$$

where $\mathbf{n} = (x, y, z)$. This expression is constant along the lines through the origin, hence it can also be evaluated at non-normalized normal vectors.

3.2 PN Parameterizations and Offset Surfaces

Consider a rational parameterization $\nu : \Omega \to \mathbb{S}^2$ of the sphere with parameters $(u, v) \subset \Omega \subseteq \mathbb{R}^2$, cf. [5]. The parameterization of the surface

$$\mathbf{q} = \Gamma \circ \nu \tag{29}$$

which is obtained by composing ν with the inverse Gauss map, is a PN (Pythagorean Normal vector) parameterization of the given surface, see [23]. It has an associated polynomial field of (non-normalized) normal vectors, such that their length is equal to the square of another polynomial, hence the components of the polynomial normals satisfy a Pythagorean condition.

A general construction and a design methodology for rational PN parameterizations have been described in [23], based on the dual representation and control structure of rational surfaces. These surfaces have also been studied in the frame of Laguerre geometry [20]. As an advantage of *rational* PN surfaces, the offset surfaces are again rational. They can be obtained either by adding the offsetting distance to the support function or by adding constant multiples of the rational unit normals to the PN parameterization.

Clearly, surfaces with rational support functions have rational PN parameterizations, since their inverse Gauss maps Γ are again rational. This is true both for odd rational support functions, where Γ takes the form (27), and for general rational support functions, see [10]. As observed in Section 2.2, this case includes quadratic polynomial surfaces and rational ruled surfaces with polynomial direction vectors.

In addition, surfaces with support functions of the form (14) or (17), where R is a rational function and the matrix D has rank 3, can be equipped with rational PN parameterizations. This is proved by describing an algorithm for generating the parameterization in [1], where support functions of this type have been derived by studying quadric surfaces. It is shown that the parameterization of these surfaces is closely related to the analysis of the intersection of two quadrics in four-dimensional space, which forms a del Pezzo surface.

Based on the results in Section 2 we note the following fact.

Proposition 1. *Non-developable ruled surfaces with polynomial direction vectors of degree less than three and non-developable polynomial translational tensor-product surfaces of degree* $(3, 2)$ *possess rational PN parameterizations.*

Proof. The PN parameterizations can be constructed as follows. First we compute the support function, which gives functions of the form (14). Second we apply the parameterization technique in [1]. This technique is not limited to regular matrices D. It is possible to use it for surfaces with singular quadratic forms, too. (See also the example below.) □

PN parameterizations of rational ruled surfaces were also discussed in [24,19]. The first paper [24] observes that these parameterizations always exist. It also presents an algorithm for parameterization for the case of direction vectors of low degree (less than 3). The second paper [19] shows how to find parameterizations over the field of complex numbers which are, however, not directly useful for geometric computing.

PN parameterizations of translational surfaces have not been discussed previously.

Example 3. In order to find a PN parameterization of the translational surface from Example 1 with the inverse Gauss maps (28), we consider the two quadrics

$$x^2 + y^2 + z^2 = 1 \quad \text{and} \quad yz = w^2 \tag{30}$$

in four-dimensional (x, y, z, w)-space. We generate a rational parameterization $(x(u, v), \ldots, w(u, v))$ of their intersection 2-surface. Then we may substitute it into (28) and replace the square roots by $|w(u, v)|$.

The substitution $x = Z$, $y = \frac{1}{2}\sqrt{2}(X-Y)$, $z = \frac{1}{2}\sqrt{2}(X+Y)$,$w = W$ transforms the two quadrics (30) into

$$X^2 + Y^2 + Z^2 = 1 \quad \text{and} \quad X^2 - Y^2 = W^2. \tag{31}$$

We apply a central projection with center $(1, 0, 0, 1)$ into the hyperplane $X = 0$ and obtain the cubic surface

$$W\,Z^2 - W + W\,Y^2 + Y^2 + W^2 = 0. \tag{32}$$

It contains the line $W = Y = 0$ which passes through two singular points (at $Z = \pm 1$) of the surface. One may now parameterize this cubic surface using the family of conics which are obtained as intersections with planes through the line. After a short computation one arrives at

$$\begin{aligned} x &= (-2 + 2\,u^2v^2 - 2\,uv^2 - 2\,u)/N, \\ y &= \tfrac{1}{2}\sqrt{2}(2 + uv^2 + u)u(v+1)^2/N, \\ z &= \tfrac{1}{2}\sqrt{2}(2 + uv^2 + u)u(v-1)^2/N, \\ N &= u^2v^4 + 4\,u^2v^2 + u^2 + 2\,uv^2 + 2\,u + 2 \end{aligned}$$

which defines a rational parameterization of the sphere of degree $(3, 4)$.

3.3 Isophotes

Recall that an *isophote* (a line of constant brightness) on a C^1 smooth surface is the set of all points where

$$\angle(\mathbf{p}_u \times \mathbf{p}_v, \mathbf{d}) = \angle(\mathbf{n}, \mathbf{d}) = \phi_0 \tag{33}$$

where $\mathbf{d}$ is the direction of the incoming light and ϕ_0 is the (constant) angle between the surface normal and the light direction, see [14]. For given values of $\mathbf{d}$ and ϕ_0, the unit normals along the isophotes form a circle on the unit sphere.

In particular, if $\phi_0 = \pi$, then the isophote is the *contour generator* of the surface with respect to the parallel projection with direction r. The corresponding normals form the great circle on the sphere which lies in the plane with normal $\mathbf{d}$ through the origin. The contour generator contains the shadow boundary with respect to illumination with direction $\mathbf{d}$.

Since circles possess a rational parameterizations, one obtains that *isophotes on surfaces with rational support functions are rational curves.* This is equivalent to [23, Theorem 3.3], which analyzes isophotes on surfaces possessing PN parameterizations whose unit normals define birational parameterizations of the sphere. This class of surfaces is identical to the set of surfaces with rational support functions.

In particular we analyze the case of quadratic polynomial surfaces, which have rational support functions of degree (3/2). The inverse Gauss map is a rational function of degree (4/4). We obtain the following result.

Proposition 2. *The isophotes (resp. contour generators) on non-developable quadratic polynomial surfaces are rational curves of degree 8 (resp. 4), which correspond to rational curves of degree* 4 *(resp. 2) in the parameter domain of the surface.*

Proof. The isophotes are obtained by applying the inverse Gauss map to spherical circles, which possess rational quadratic parameterizations. The contour generators are found by applying the inverse Gauss map to lines, since these project into great circles on the sphere (by a central projection with respect to the origin) and the inverse Gauss map gives the same point for all points along a line through the origin.

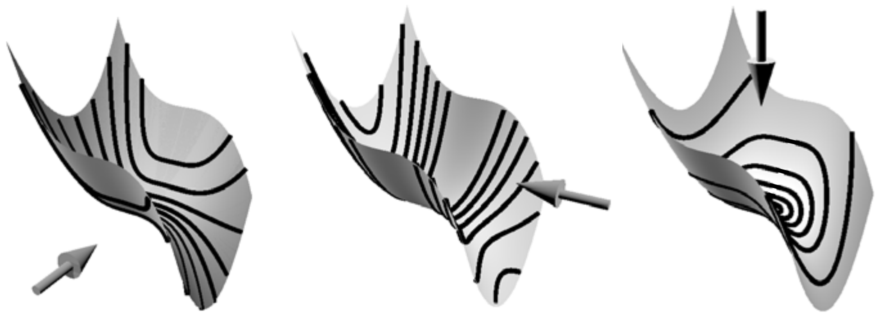


Fig. 1. Isophotes on a translational surface for different light directions **d**

The second part of this observation is proved by solving the linear system

$$\mathbf{p}_u(u,v)^\top \mathbf{n}(t) = 0, \quad \mathbf{p}_u(u,v)^\top \mathbf{n}(t) = 0 \tag{34}$$

for the surface parameters (u, v), where $\mathbf{n}(t)$ is a quadratic rational parameterization of the spherical circle defined by (33), or a line in the plane which is perpendicular to $\mathbf{d}$. $\square$

In the case of developable quadratic polynomial surfaces (34), which are either planes, quadratic cylinders or quadratic half-cones, the two linear equations in (34) do not possess solutions for all $\mathbf{n}$. If solutions exist, then they are non-unique, and they form lines in the parameter domain. It turns out that the isophotes are either straight lines (for planes and cylinders) or half-lines (for half-cones).

Isophotes on surfaces with more general support functions can be parameterized by composing the parameterization of a spherical circle with the inverse Gauss map. If the support functions involves only rational operations and square roots, then one obtains square-root parameterizations of the isophotes.

Example 4. The isophotes on the translational surface from Example 1 have square-root parameterizations. Fig. 1 shows the isophotes for three different light directions.

3.4 Other Applications of the Dual Surface

We conclude this section by briefly mentioning two additional applications of support functions and dual surfaces of free-form surfaces.

First, one may use them to compute the contours with respect to central projection. The tangent planes of the cone which is formed by the projection lines touching the surface correspond to a curve on the dual surface, which is simply the intersection with a plane. The intersection of this cone with the image plane gives the contour of the surface with respect to the central projection. In certain cases it is possible to find simple parameterizations of the planar intersections of the dual surfaces. E.g., in the case of quadratic polynomial surface, one obtains planar cubics on the dual surface which admit square-root parameterizations.

Second, it is an interesting idea to exploit duality for convex hull computation of free-form surfaces. This is described in more detail in [7].

4 Support Function Approximation

We propose a new method for computing isophotes on general free-form surfaces. The method consists of three steps. First, a piecewise quadratic approximation of the surface is generated. Second, this approximation is used to define a piecewise defined approximation of the inverse Gauss map of the surface. Finally, the Gauss map is applied to spherical circles.

4.1 Piecewise Quadratic Approximation

We consider a given surface $\mathbf{p} : \Omega \rightarrow \mathbb{R}^3$ with domain $\Omega = [0, 1]^2$, which is assumed to be C^3 smooth, i.e., $\mathbf{p} \in C^3(\Omega, \mathbb{R}^3)$. Several techniques for generating a C^1 smooth piecewise approximation $\mathbf{p}^*$ exist.

For instance, one may use any triangulation of the domain and then use Powell-Sabin elements with respect to this triangulation, see [14]. The Powell-Sabin spline is uniquely determined by given first order Hermite data at the vertices of the original triangulation (which can be sampled from the given function), and it is C^1 smooth. However, Powell-Sabin elements tend to create relatively long and thin triangles, and therefore the visual quality of the approximation is often not satisfying.

Instead we use a piecewise quadratic function with respect to a criss-cross partition of the domain. First, the domain Ω is split into n^2 boxes with vertices $(\frac{i}{n}, \frac{j}{n})_{i,j=0,\ldots,n}$. Second, each box is split into four triangles by adding the diagonals of the box. The space of piecewise quadratic functions, which are C^1 smooth, will be denoted with S_n. It is spanned by translates of the Zwart-Powell element, which is a special box spline, see [3].

Recently, a quasi-interpolation operator Q for this space has been presented in [8]. For a given value of n, the operator Q maps any function $f \in C^3(\Omega, \mathbb{R})$ into a piecewise quadratic approximation $Qf \in S_n$. When applied to the three components of the surface $\mathbf{p}$, one obtains a piecewise quadratic approximation $\mathbf{p}^* = Q\mathbf{p} \in S_n^3$. The computation of $Q\mathbf{p}$ requires solely the evaluations of $\mathbf{p}$ at certain points.

As shown in [8], this operator preserves quadratic functions and produces approximations Qf which possess the optimal approximation order 3 with respect to the maximum norm in $C(\Omega, \mathbb{R})$. In addition, all first and second derivatives are approximated with order 2 and 1.

More precisely, the maximum difference between $\mathbf{p}$ and $\mathbf{p}^*$ and between its various first and second derivatives, can be bounded by $\hat{C}_k \Delta^k$, where $k = 3, 2, 1$, respectively. The constants $\hat{C}_k$ depend on $\mathbf{p}$ and $\Delta = \frac{1}{n}$ is the size of the boxes.

4.2 Approximate Dual Implicitization

Once a piecewise quadratic approximation $\mathbf{p}^*$ has been found, we use it to define an approximate inverse Gauss map.

1. First we split $\mathbf{p}^*$ into its polynomial segments. Each segment is represented as a quadratic triangular Bézier patch.
2. If one of these patches contains parabolic points, but is not entirely developable, then the corresponding points in the triangular domain form one or more (at most three) straight lines. These lines will be called the parabolic lines, see [2].
   In this case we split the domain triangle along the parabolic lines and triangulate the resulting subdomains. After this step, all patches either do not contain any parabolic points in the interior of their domains, or they are developable surfaces. Let
   $$\{\mathbf{p}_i^*, i = 1, \ldots, N\} \tag{35}$$
   be the set of patches so obtained.
3. We compute the Gauss images $G_i \subset \mathbb{S}^2$ of all patches $\mathbf{p}_i^*$.
   – For non-developable patches we obtain *curved spherical triangles*, which are bounded by spherical conics, i.e., by intersections of quadratic cones (with their apices at the origin) with the unit sphere.
   – If one of the patch boundaries is a parabolic line, then the Gauss image degenerates into a *curved spherical biangle*. Indeed, the normals of quadratic patches along their (at most 3) parabolic lines are constant. If $\mathbf{p}$ is regular, then also $\mathbf{p}^*$ is regular, provided that Δ is sufficiently small. Consequently, at most one of the three patch boundaries is a parabolic line for each non-developable patch, since parabolic lines on quadratic patches intersect in singular points.
   – The Gauss images of developable patches are arcs of spherical conics.
   The collection of all Gauss images G_i covers a certain subset of the unit sphere. Each point may be covered several times. Each of the Gauss image G_i can be represented by the equations of (at most) three cones and by an auxiliary plane,
   $$G_i = \{\mathbf{n} \in \mathbb{S}^2 : \ \mathbf{n}^\top A_i^{(1)} \mathbf{n} \geq 0 \wedge \mathbf{n}^\top A_i^{(2)} \mathbf{n} \geq 0 \wedge \mathbf{n}^\top A_i^{(3)} \mathbf{n} \geq 0 \wedge \mathbf{n}^\top \mathbf{w}_i \geq 0\},$$
   where the symmetric matrices $A_i^{(j)}$ represent the cones and the vector $\mathbf{w}_i$ is the normal vector of the auxiliary plane.
4. Finally we compute the inverse Gauss maps.
   – For non-developable surface patches $\mathbf{p}_i^*$ we compute the rational support function H_i, see Section 2.2, and the inverse Gauss map $\Gamma_i = \nabla H_i$, see Eq. (27). H_i and Γ_i are rational functions of degree 3/2 and 4/4, respectively.
   – The Gauss map Γ_i of a developable surface patch $\mathbf{p}_i^*$ assigns to each normal in G_i a line segment on the surface patch.

The union of the support functions defines a multi-valued mapping

$$H^* : \quad \bigcup_{i=1}^{N} G_i \to \mathbb{R} : \quad \mathbf{n} \mapsto \bigcup_{\mathbf{n}\in G_i} \{H_i(\mathbf{n})\} \tag{36}$$

which approximates the support function of the original surface $\mathbf{p}$. The surface defined by $H^*(\mathbf{n}) = 1$, where we extended the domain of each support function segment to the cone $\mathbb{R}G_i$, can be seen as an *approximate dual implicitization* of the original surface, cf. ([8]). (See [6] for information on approximate implicitization.)

The union of the Gauss maps defines a multi-valued mapping

$$\Gamma^* : \quad \bigcup_{i=1}^{N} G_i \to \mathbb{R}^3 : \quad \mathbf{n} \mapsto \bigcup_{\mathbf{n}\in G_i} \{\Gamma_i(\mathbf{n})\} \tag{37}$$

which approximates the inverse Gauss map of the original surface $\mathbf{p}$.

4.3 Isophote Approximation

We compute the isophotes on the approximating surface $\mathbf{p}^*$ for a given light direction $\mathbf{d}$ and different angles ϕ_0. This is done by applying the mapping Γ^* to the points of the corresponding circles $\mathbf{n}^\top \mathbf{d} = \cos\phi_0$ on the unit sphere. The circles are represented as rational quadratic curves and the intersections with the boundaries of the domains G_i are found by numerically solving quartic equations.

Several examples are shown in Figures 2–4. It can be seen that the quality of the isophotes improves if a larger number of quadratic patches (and hence a smaller box size Δ) is used.

We consider an arbitrary but fixed value of $\phi_0 \in [0, \pi]$. The isophotes can be described with the help of the *brightness functions*

$$\beta(u,v) = \mathbf{d}^\top \mathbf{N}(u,v) \quad \text{and} \quad \beta^*(u,v) = \mathbf{d}^\top \mathbf{N}^*(u,v) \tag{38}$$

of $\mathbf{p}$ and $\mathbf{p}^*$, where $\mathbf{N} = (\mathbf{p}_u \times \mathbf{p}_v)/||\mathbf{p}_u \times \mathbf{p}_v||$ is the field of unit normals of the given surface $\mathbf{p}$ and $\mathbf{N}^*$, which is similarly defined, is the field of unit normals of the approximating surface $\mathbf{p}^*$. The level sets

$$L = \beta^{-1}(\cos\phi_0) \quad \text{and} \quad L^* = (\beta^*)^{-1}(\cos\phi_0) \tag{39}$$

in the parameter domain define the isophotes

$$\mathbf{p}(L) \quad \text{and} \quad \mathbf{p}^*(L^*), \tag{40}$$

on the exact and on the approximating surface, respectively.

For any two closed sets $A, B \subseteq \mathbb{R}^k$, let

$$\mathrm{HD}(A,B) = \max_{\mathbf{x}\in A} \min_{\mathbf{y}\in B} ||\mathbf{x} - \mathbf{y}|| \tag{41}$$

be their one-sided Hausdorff distance. We will use it to analyze the convergence of the isophotes.

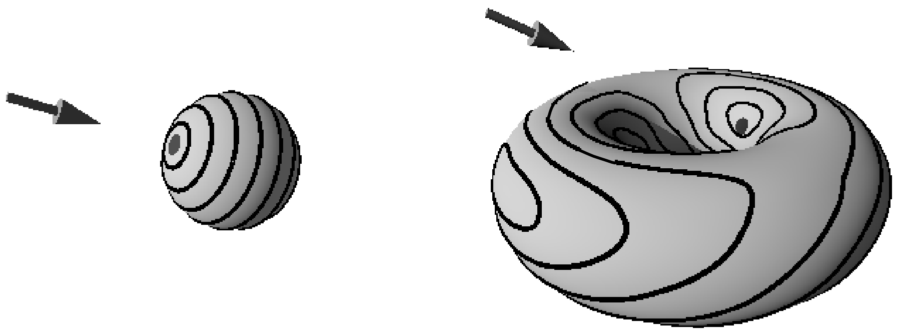


Fig. 2. The circles on the unit sphere are mapped into the isophotes of a quadratic approximation of a torus surface

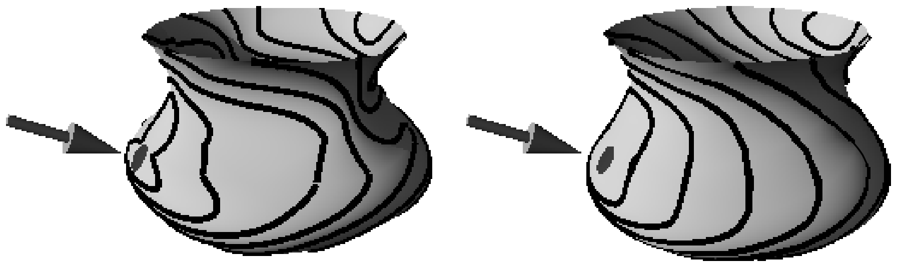


Fig. 3. Isophotes on a vase-shaped surface, which is approximated by 32 (left) and 100 (right) quadratic patches

Theorem 1. *Consider a C^3 smooth surface patch* $\mathbf{p}$ *whose domain* $\Omega = [0,1]^2$ *contains neither parameter values of parabolic points of the surface nor points where the surface normal is parallel to the light direction* $\mathbf{d}$. *We approximate the surface using the quasi-interpolant described in Section 4.1 and use the result to compute the isophote for a given value of* $\cos\phi_0$. *There exists a constant* C, *which depends on the given surface* $\mathbf{p}$, *on the light direction* $\mathbf{d}$ *and the angle* ϕ, *such that*

$$\max\{\mathrm{HD}(\mathbf{p}(L), \mathbf{p}^*(L^* \cup \partial\Omega)),\ \mathrm{HD}(\mathbf{p}^*(L^*), \mathbf{p}(L \cup \partial\Omega))\} \leq C\Delta^2 \tag{42}$$

where Δ *is the box size used for the quasi-interpolants.*

Proof. The brightness functions β and β^* are C^2 smooth and merely continuous, respectively. However, β^* is piecewise differentiable (within each triangle of the criss-cross triangulation). Since the first derivatives of $\mathbf{p}^*$ have uniform quadratic convergence, we may conclude that there exists a constant C_1 such that

$$\forall (u,v) \in \Omega : \quad |(\beta - \beta^*)(u,v)| < C_1\Delta^2. \tag{43}$$

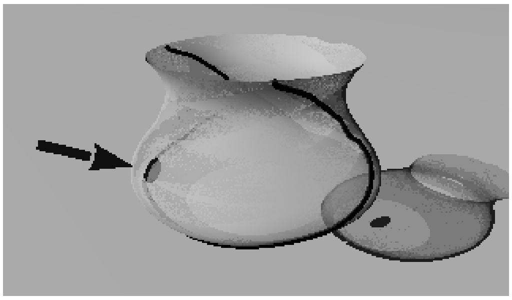


Fig. 4. The contour generator of a vase approximated by 100 quadratic patches and its image under the parallel projection. The contour generator is projected into the contour of the surface.

Since parabolic points and points with normal **d** were excluded from Ω, the length of the gradient $\nabla\beta$ (with respect to the surface parameters u, v) is strictly positive, i.e., there exists a constant C_2 such that

$$\forall (u,v) \in \Omega : \quad ||(\nabla\beta)(u,v)|| > C_2. \tag{44}$$

As the first and second derivatives of the quadratic approximation converge to the derivatives of **p**, we obtain that

$$\forall (u,v) \in \Omega^0 : \quad ||(\nabla\beta^*)(u,v)|| > \frac{C_2}{2} \tag{45}$$

for sufficiently small values of Δ, where Ω^0 is the domain which is obtained by excluding the lines of the criss-cross triangulation.

In order to prove (42) for the first of the two one-sided Hausdorff distances, we consider an arbitrary point (u_0, v_0) in the parameter domain which belongs to the level set L. The value of the brightness function β^* at this point satisfies

$$\cos\phi_0 - C_1\Delta^2 \leq \beta^*(u_0, v_0) \leq \cos\phi_0 + C_1\Delta^2. \tag{46}$$

Starting at this point, we create a curve $\mathbf{c}(s)$ by integrating the normalized gradient field $\nabla\beta^*/||\nabla\beta^*||$. This curve is then given by an arc length parameterization, since it is the integral curve of a field of unit vectors. While it is not C^1 smooth (as the gradient $\nabla\beta^*$ is not C^1), it is C^1 except for its intersections with the lines of the criss-cross triangulation. The restriction $(\beta^* \circ \mathbf{c})(s)$ of the brightness function β^* to this curve defines a continuous and monotonic function of the arc length parameter. Its derivative with respect to s—which is defined everywhere except for the intersections with the lines in the criss-cross triangulation—satisfies

$$(\beta^* \circ \mathbf{c})'(s) = \frac{(\nabla\beta^*)(\mathbf{c}(s))}{||(\nabla\beta^*)(\mathbf{c}(s))||} \cdot (\nabla\beta^*)(\mathbf{c}(s)) = (\nabla\beta^*)(\mathbf{c}(s)) > \frac{C_2}{2}. \tag{47}$$

If $\beta^*(u_0, v_0) < \cos\phi_0$, then we travel along the curve $\mathbf{c}(s)$ into the direction of the gradient, which increases the value of β^*, until we hit either the level set L^* or the domain boundary $\partial\Omega$. However, since the derivative satisfies (47) and the deviation of $\beta^*(u_0, v_0)$ from $\cos\phi_0$ is not larger than $C^1\Delta^2$, the travel distance is bounded by

$$C_1\Delta^2\,\frac{2}{C_2}. \tag{48}$$

Otherwise, if $\beta^*(u_0, v_0) > \cos\phi_0$, then we travel along the curve $\mathbf{c}(s)$ into the direction of the negative gradient and use a similar argument to bound the travel distance.

Summing, we find a point $(u, v) \in L^* \cup \partial B$ such that

$$||(u, v) - (u_0, v_0)|| \le C_1\Delta^2\,\frac{2}{C_2}. \tag{49}$$

Now we apply the mappings $\mathbf{p}$ and $\mathbf{p}^*$ and use the triangle inequality,

$$||\mathbf{p}^*(u, v) - \mathbf{p}(u_0, v_0)|| \le ||\mathbf{p}^*(u, v) - \mathbf{p}(u, v)|| + ||\mathbf{p}(u, v) - \mathbf{p}^*(u_0, v_0)||.$$

The first term on the right hand side is bounded by $C_3\Delta^3$, due to the third order of approximation. The second term is bounded by $2C_4(C_1/C_2)\Delta^2$, where C_4 is a global bound on the norm of the Jacobian of $\mathbf{p}$. This proves the result for the first Hausdorff distance in (42). The second Hausdorff distance can be dealt with similarly, but this time using the smooth integral curves of the vector field $\nabla\beta/||\nabla\beta||$. $\square$

5 Conclusion

We discussed the support functions of several special classes of free-form surfaces. Based on observations concerning quadratic polynomial surfaces, which possess rational support functions of degree 3/2, and on a quasi-interpolation scheme for bivariate quadratic splines, we formulated an algorithm for approximate isophote computation and analyzed its convergence.

Instead of quadratic polynomial surfaces, one might use other classes of approximating surfaces which still admit closed-form representations of the inverse Gauss maps (e.g., using square roots). This could give approximation schemes for isophotes with an even higher rate of convergence, which may be a possible topic of further research.

Another possible application for our results is mesh contour smoothing: see [27]. This motivates the further investigation of surfaces with simple contours, such as quadratic polynomial surfaces.

Acknowledgement. M. Aigner and B. Jüttler gratefully acknowledge financial support by the Austrian Science Fund through the National Research Network S92 “Industrial Geometry” and by grant WTZ 11/2007 of the Austrian-Spanish programme for scientific cooperation. The second author acknowledges the support of the Spanish grant MTM2008-04699-C03-03. M. L. Sampoli acknowledges the financial support of the Italian GNCS through the “Senior visiting program 2008”.

References

1. Aigner, M., Jüttler, B., Gonzalez-Vega, L., Schicho, J.: Parameterizing surfaces with certain special support functions, including offsets of quadrics and rationally supported surfaces. J. Symb. Comp. 44, 180–191 (2009)
2. Bastl, B., Jüttler, B., Kosinka, J., Lávička, M.: Computing exact rational offsets of quadratic triangular Bézier surface patches. Comp.-Aided Design 40, 197–209 (2008)
3. de Boor, C., Höllig, K., Riemenschneider, S.: Box Splines. Springer, Heidelberg (1993)
4. Degen, W.L.F.: The types of triangular Bézier surfaces. In: Mullineux, G. (ed.) The Mathematics of Surfaces IV. The Institute of Mathematics and its Applications Conference Series, vol. 38, pp. 153–171. Clarendon Press, Oxford (1996)
5. Dietz, R., Hoschek, J., Jüttler, B.: An algebraic approach to curves and surfaces on the sphere and on other quadrics. Comp. Aided Geom. Design 10, 211–229 (1993)
6. Dokken, T.: The GAIA project on intersection and implicitization. In: Jüttler, B., Piene, R. (eds.) Geometric Modeling and Algebraic Geometry, pp. 5–30. Springer, Heidelberg (2008)
7. Elber, G., Johnstone, J.K., Kim, M.-S., Seong, J.-K.: The Kernel of a Freeform Surface and its duality with the Convex Hull of its Tangential Surface. Int. J. Shape Modeling 12, 129–142 (2006)
8. Foucher, F., Sablonnière, P.: Approximating partial derivatives of first and second order by quadratic spline quasi-interpolants on uniform meshes. Mathematics and Computers in Simulation 77, 202–208 (2008)
9. Gravesen, J.: Surfaces parametrised by the normals. Computing 79, 175–183 (2007)
10. Gravesen, J., Jüttler, B., Šír, Z.: On Rationally Supported Surfaces. Comp. Aided Geom. Design 25, 320–331 (2008)
11. Hahmann, S.: Visualization techniques for surface analysis. In: Bajaj, C. (ed.) Advanced Visualization Techniques, pp. 49–74. Wiley, Chichester (1999)
12. Hahmann, S., Belyaev, A., Buse, L., Elber, G., Mourrain, B., Rössl, C.: Shape Interrogation. In: Floriani, L., Spagnuolo, M. (eds.) Shape Analysis and Structuring, pp. 1–52. Springer, Heidelberg (2007)
13. Hoschek, J.: Dual Bézier curves and surfaces. In: Barnhill, R.E., Boehm, W. (eds.) Surfaces in Computer Aided Geometric Design, pp. 147–156. North Holland, Amsterdam (1983)
14. Hoschek, J., Lasser, D.: Fundamentals of Computer Aided Geometric Design. AK Peters, Wellesley (1993)
15. Johansen, H., Løberg, M., Piene, R.: Monoid hypersurfaces. In: Jüttler, B., Piene, R. (eds.) Geometric Modeling and Algebraic Geometry, pp. 55–78. Springer, Heidelberg (2008)
16. Lávička, M., Bastl, B.: Rational hypersurfaces with rational convolutions. Comp. Aided Geom. Design 24, 410–427 (2007)
17. Lichtenauer, J., Hendriks, E., Reinders, M.: Isophote properties as features for object detection. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. 2, pp. 649–654 (2005)
18. Morse, B.S., Schwartzwald, D.: Isophote-based interpolation. In: Proc. Int. Conference on Image Processing, vol. 3, pp. 227–231 (1998)
19. Mühlthaler, H., Pottmann, H.: Computing the Minkowski sum of ruled surfaces. Graphical Models 65, 369–384 (2003)

20. Peternell, M., Pottmann, H.: A Laguerre geometric approach to rational offsets. Comput. Aided Geom. Design 15, 223–249 (1998)
21. Peters, J., Reif, U.: The 42 equivalence classes of quadratic surfaces in affine n-space. Comp. Aided Geom. Design 15, 459–473 (1998)
22. Poeschl, T.: Detecting surface irregularities using isophotes. Comput. Aided Geom. Design 1, 163–168 (1984)
23. Pottmann, H.: Rational curves and surfaces with rational offsets. Computer Aided Geometric Design 12, 175–192 (1995)
24. Pottmann, H., Lü, W., Ravani, B.: Rational Ruled Surfaces and Their Offsets. Graphical Models and Image Processing 58, 544–552 (1996)
25. Pottmann, H., Steiner, T., Hofer, M., Haider, C., Hanbury, A.: The Isophotic Metric and its Application to Feature Sensitive Morphology on Surfaces. In: Pajdla, T., Matas, J. (eds.) ECCV 2004. LNCS, vol. 3024, pp. 560–572. Springer, Heidelberg (2004)
26. Sabin, M.: A Class of Surfaces Closed under Five Important Geometric Operations, Report no. VTO/MS/207, British aircraft corporation (1974), http://www.damtp.cam.ac.uk/user/na/people/Malcolm/vtoms/vtos.html
27. Wang, L., Tu, C.H., Wang, W., Meng, X.X., Chan, B., Yan, D.M.: Silhouette smoothing for real-time rendering of mesh surfaces. IEEE Trans. on Vis. Comp. Graphics 14, 640–652 (2008)
28. Zhan, Y., Wang, M., Li, M.: An Isophote-Oriented Image Interpolation Method. In: Proc. Int. Symp. Computer Science and Computational Technology, pp. 723–726 (2008)

Swept Volume Parameterization for Isogeometric Analysis

M. Aigner[1], C. Heinrich[1,2], B. Jüttler[1], E. Pilgerstorfer[1], B. Simeon[3], and A.-V. Vuong[3]

[1] Johannes Kepler University, Institute of Applied Geometry, Linz, Austria
[2] Siemens Corporate Technology, München, Germany
[3] Technische Universität München, Zentrum Mathematik, Germany

Abstract. Isogeometric Analysis uses NURBS representations of the domain for performing numerical simulations. The first part of this paper presents a variational framework for generating NURBS parameterizations of swept volumes. The class of these volumes covers a number of interesting free-form shapes, such as blades of turbines and propellers, ship hulls or wings of airplanes. The second part of the paper reports the results of isogeometric analysis which were obtained with the help of the generated NURBS volume parameterizations. In particular we discuss the influence of the chosen parameterization and the incorporation of boundary conditions.

Keywords: NURBS volume parameterization, Isogeometric Analysis, swept volume.

1 Introduction

The concept of isogeometric analysis, which was introduced by Hughes et al. [1], provides an opportunity for bridging the gap between Computer Aided Design (CAD) and numerical simulation based on the finite element method (FEM). Its potential has been demonstrated in a substantial number of publications, which also discuss related issues such as efficient techniques for numerical integration and the incorporation of boundary conditions [2,3,4,5,6,7,8,9,10,11,12].

The European project EXCITING [13] aims at applying Isogeometric Analysis to free-form objects arising in real-world applications, in particular in the transportation industry. In this paper we report on our first experiences with this approach. In particular we will focus on the following problem: Given a three-dimensional free-form object, find a parameterization by (one or several) NURBS volumes.

NURBS volumes have been discussed in the classical literature in Computer Aided Geometric design, see [14] and the references cited therein. The existing literature on volume parameterizations by NURBS concentrates mostly on applications to object modeling via free-form deformations. In this context, the given object (often represented as a triangular mesh) is embedded into a simple NURBS volume, which represents (e.g.) the bounding box of the object. By modifying the control points and weights of the NURBS volume one can then edit the shape of the embedded object. This or similar editing capabilities are available in many modeling systems.

In order to obtain NURBS volume parameterizations which are suitable for isogeometric analysis, we have to solve a different problem. The boundary surfaces of the object are given as (trimmed or untrimmed) NURBS surfaces. A volume parameterization which respects the given boundary surfaces has to be generated. While it is a very challenging open problem to solve this task for general CAD objects, it is possible to obtain reasonable results for special classes of free-form objects. Nevertheless, these classes already cover a number of interesting applications.

The first part of the paper presents a method for generating NURBS parameterizations of swept volumes, which are obtained by sweeping a closed curve through space. These volumes are also known as generalized cylinders, and there exists an extensive literature discussing them, e.g. [15,16,17]. The second part reports results of isogeometric analysis which we obtained with the help of the generated NURBS volume parameterizations.

2 Swept Volume Parameterization

We will describe a variational framework for generating the control points of a NURBS volume from given boundary conditions and one or more guiding curves.

2.1 NURBS Representation of Swept Volumes

A non-uniform rational B-spline volume (NURBS volume) is defined by the parametric representation

$$\mathbf{F}(r,s,t) = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} R_{ijk}(r,s,t)w_{ijk}\mathbf{d}_{ijk}, \quad (r,s,t)\in[0,1]^3, \tag{1}$$

where the domain is the unit cube in $\mathbb{R}^3$. More generally, the domain can be chosen as any axis-aligned box in $\mathbb{R}^3$, but for the purposes of the present paper it suffices to consider the unit cube.

The blending functions

$$R_{ijk}(r,s,t) = \frac{N_{i,\mathcal{R}}(r)N_{j,\mathcal{S}}(s)N_{k,\mathcal{T}}(t)}{\sum_{i'\in\mathcal{I}}\sum_{j'\in\mathcal{J}}\sum_{k'\in\mathcal{K}} w_{i'j'k'}N_{i',\mathcal{R}}(r)N_{j',\mathcal{S}}(s)N_{k',\mathcal{T}}(t)}$$

are called the rational spline basis functions associated with the weights w_{ijk}. The functions $N_{i,\mathcal{R}}(r)$, $N_{j,\mathcal{S}}(s)$ and $N_{k,\mathcal{T}}(t)$ are B-splines of certain degrees with respect to three given knot vectors $\mathcal{R}$, $\mathcal{S}$ and $\mathcal{T}$ with degree–fold boundary knots 0 and 1. The index sets $\mathcal{I},\mathcal{J},\mathcal{K}\subset\mathbb{Z}$ of the control points are determined by the knot sequences and degrees of the B-splines.

The vector–valued coefficients $\mathbf{d}_{ijk}\in\mathbb{R}^3$ are called de Boor points or *control points.* The three-dimensional grid determined by them is called the de Boor net

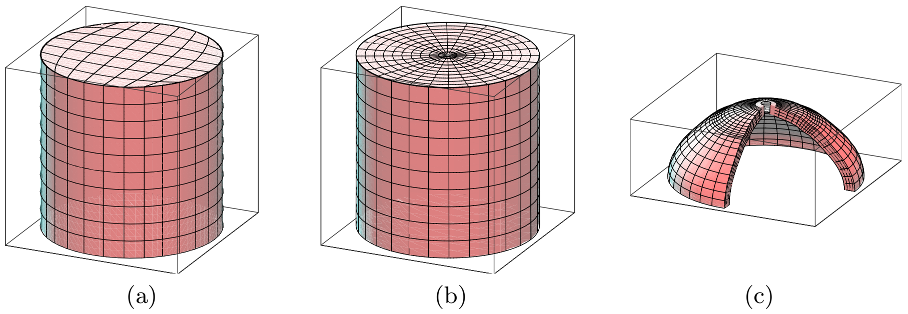


Fig. 1. Parameterization of three simple swept volumes. The cylinders (a) and (b) are based on different representations of the circular patch.

or control net. In order to simplify the notation, we denote the vector of control points by

$$\mathbf{d} = (\mathbf{d}_{ijk})_{i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}}. \tag{2}$$

In addition, each control point has an associated scalar value w_{ijk}, which is called its *weight*. In the remainder of this paper we will assume that these weights are given; they are not subject to the optimization process described below. More precisely, we assume that they are determined by the given planar shape which is moved through space.

See [14,18] for more information on rational spline techniques.

The parameterization of a general three-dimensional volume by a collection of tensor-product spline volumes ("patches") is a non-trivial problem, and one cannot expect to find a general method that can deal successfully with all cases. In this paper we concentrate on the special class of *swept volumes*.

These volumes are obtained by moving a two-dimensional shape through space, where the geometry of the moving shape may be subject to an evolution during the sweep. The motion of the shape is guided by one or more guiding curves. This class of volumes includes a large number of objects with significant industrial applications, ranging from simple shapes (cylinders, spherical shells) to more complicated ones (turbine blades, aircraft wings, ship hulls). Three examples of simple shapes are presented in Fig. 1.

In the following we assume that t, which will be called the *sweep parameter*, is associated with the motion of the two-dimensional shape. The remaining two parameters r and s parameterize the two-dimensional shape as a surface patch. More precisely, the surface

$$\mathbf{F}(r, s, t'), \qquad (r, s) \in [0, 1]^2, \tag{3}$$

which is obtained by considering a constant value of $t' \in [0, 1]$, will be called (one instance of) the *moving surface*. On the other hand, the curve

$$\mathbf{F}(r', s', t), \qquad t \in [0, 1], \tag{4}$$

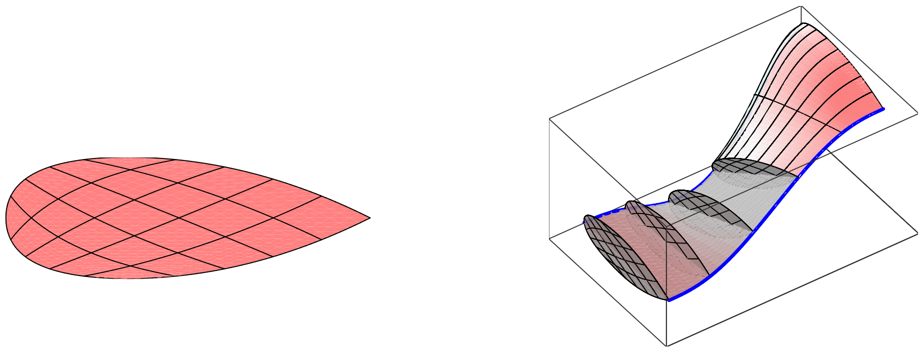

(a) (b)

Fig. 2. Parameterization of the two-dimensional shape as a tensor-product patch with three singular vertices (a). Two guiding curves (blue), intermediate moving surfaces and a segment of the final parameterization (b).

which is obtained by considering constant values of $(r', s') \in [0,1]^2$, will be considered as the *trajectory* of a point.

The variational framework will be illustrated by an example of a blade-like NURBS volume, see Fig. 2. The underlying parameterization of the planar shape as a tensor-product patch with three singular vertices is shown in Fig. 2a.

2.2 Boundary Conditions

Due to the multiplicity of the boundary knots, the two boundary nets

$$(\mathbf{d}_{i\,j\,\min\mathcal{K}})_{i\in\mathcal{I},j\in\mathcal{J}} \quad \text{and} \quad (\mathbf{d}_{i\,j\,\max\mathcal{K}})_{i\in\mathcal{I},j\in\mathcal{J}} \tag{5}$$

of the grid of control points determine the two boundary surfaces

$$\mathbf{F}(r,s,0) \quad \text{and} \quad \mathbf{F}(r,s,1), \quad (r,s)\in[0,1]^2, \tag{6}$$

of the NURBS volume. Consequently, if these two boundary surfaces are specified by the user, then the corresponding subset of the set of control points $\mathbf{d}$ can be eliminated from the set of unknowns; it is already determined by the boundary conditions.

In addition, the next two nets

$$(\mathbf{d}_{i\,j(\min\mathcal{K}+1)})_{i\in\mathcal{I},j\in\mathcal{J}} \quad \text{and} \quad (\mathbf{d}_{i\,j(\max\mathcal{K}-1)})_{i\in\mathcal{I},j\in\mathcal{J}} \tag{7}$$

of control points determine the derivatives with respect to the sweep parameter t along the two boundary faces. Sometimes it is necessary to specify the direction of these derivatives, e.g., for composing two B-spline volumes with C^1 continuity. If a vector specifying the direction of these derivatives at $t = 0$ is given, then the second family of control points $\mathbf{d}_{ij(\min\mathcal{K}+1)}$ is obtained by moving a copy of the boundary control points $\mathbf{d}_{ij\,\min\mathcal{K}}$ along this vector. A similar construction can be applied at the other boundary, where $t = 1$. The distance between the first and second family of control points can either be prescribed or can also be subject to the optimization procedure described below.

2.3 Guiding Curves and Reference Shape

Guiding curves. In addition to the boundary conditions, we assume that n *guiding curves*

$$\mathbf{c}_\ell : [0,1] \to \mathbb{R}^3, \quad t \mapsto \mathbf{c}_\ell(t), \quad \ell = 1, \ldots, n, \tag{8}$$

are given, which are to specify the motion of the moving two-dimensional shape through space. The motion will be governed by the guiding curves and by several shape constraints.

The moving surfaces $\mathbf{F}(r, s, t')$, which are obtained for constant values $t' \in [0,1]$ and $(r, s) \in [0,1]^2$ are to follow the motion of the points $\mathbf{c}_\ell(t')$ of the guiding curves. For each guiding curve $\mathbf{c}_\ell$, we choose parameter values $(\tilde{r}_\ell, \tilde{s}_\ell)$ of an associated point in the two-dimensional shape. In order to define the NURBS volume, we minimize the distance

$$f_{\mathrm{A}}(\mathbf{d}) = \sum_{\ell=1}^{n} \int_0^1 ||\mathbf{F}(\tilde{r}_\ell, \tilde{s}_\ell, t) - \mathbf{c}_\ell(t)||^2 \mathrm{d}t. \tag{9}$$

between the guiding curves and the associated points. The right-hand side in (9) will be called the *approximation term* of the objective function.

Reference shape. In principle one can use any point $(\tilde{r}_\ell, \tilde{s}_\ell)$ in the parameter domain as parameters of the associated points. However, it is more appropriate to select certain special points, e.g. the center of gravity or points on the boundary of the moving surface.

More precisely, we consider a planar *reference shape*

$$\mathbf{R} : [0,1]^2 \to \mathbb{R}^2 \tag{10}$$

of the moving surface, see Fig. 3. The reference shape is a parameterization of the plane which represents the expected average shape of the moving surface. The given guiding curves $\mathbf{c}_\ell$ are now associated with certain points $\mathbf{R}(\tilde{r}_\ell, \tilde{s}_\ell)$ of the reference position. These points will be called the *anchors* of the guiding curves.

For instance, in the case of the blade-like NURBS volume (Fig. 2), we use two guiding curves and associate them with the two extremal points of the two-dimensional reference shape.

Influence functions. For each guiding curve $\mathbf{c}_\ell$ we define an *influence function*

$$\alpha_\ell : [0,1]^2 \to [0,1] \tag{11}$$

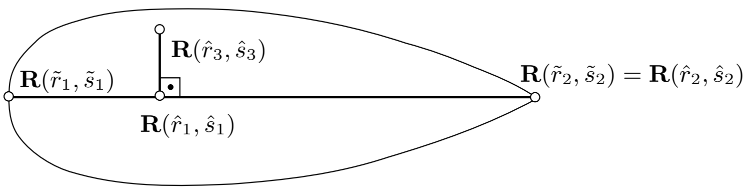

$\mathbf{R}(\hat{r}_3,\hat{s}_3)$
$\mathbf{R}(\tilde{r}_1,\tilde{s}_1)$
$\mathbf{R}(\tilde{r}_2,\tilde{s}_2)=\mathbf{R}(\hat{r}_2,\hat{s}_2)$
$\mathbf{R}(\hat{r}_1,\hat{s}_1)$

Fig. 3. Reference shape for two guiding curves. The points $\mathbf{R}(\tilde{r}_i,\tilde{s}_i)$ serve as anchors of the guiding curves. The points $\mathbf{R}(\hat{r}_i,\hat{s}_i)$ will be used later to define the first shape term.

such that the weight $\alpha_\ell(r,s)$ controls the influence of the ℓth guiding curve to the trajectory $\mathbf{F}(r,s,t)$, $t\in[0,1]$ and

$$\sum_{\ell=1}^{n}\alpha_\ell(r,s)\equiv 1. \tag{12}$$

More precisely, the point $\mathbf{F}(r,s,t)$ is associated with the weighted average

$$\hat{\mathbf{c}}(r,s,t)=\sum_{\ell=1}^{n}\alpha_\ell(r,s)\,\mathbf{c}_\ell(t) \tag{13}$$

of guiding curves. The choice of the influence functions depends on the number n of guiding curves, as follows.

– If $n=1$, the cross section sweeps along a single guiding curve $\mathbf{c}_1(t)$, hence we choose $\alpha_1(r,s)\equiv 1$.
– If $n=2$, then the weights $\alpha_1(r,s)$ and $\alpha_2(r,s)$ are computed by orthogonal projection of $\mathbf{R}(r,s)$ onto the line segment connecting the points with parameters $(\tilde{r}_1,\tilde{s}_1)$ and $(\tilde{r}_2,\tilde{s}_2)$, see Fig. 3. The ratio of the projected point with respect to the line segment determines the values of $\alpha_1(r,s)$, $\alpha_2(r,s)$. The points
$$\hat{\mathbf{c}}(r,s,t)=\alpha_1(r,s)\mathbf{c}_1(t)+\alpha_2(r,s)\mathbf{c}_2(t) \tag{14}$$
form the ruled surface which is spanned by the two curves $\mathbf{c}_1(t)$ and $\mathbf{c}_2(t)$, see Fig. 2(b).
– If $n=3$, then the weights $\alpha_\ell(r,s)$ are chosen as the barycentric coordinates of the point $\mathbf{R}(r,s)$ with respect to the triangle formed by the points with parameters $(\tilde{r}_\ell,\tilde{s}_\ell)$, $\ell=1,2,3$.
– If $n>3$, then one can use one of the various generalizations of barycentric coordinates to closed planar polygons with more than three vertices, see e.g. [19].

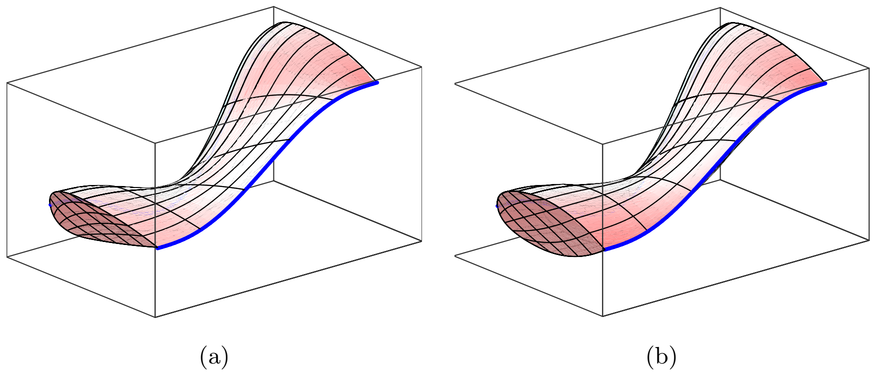


Fig. 4. Parameterization of a blade. In (a), the orthogonality is averaged between the two guiding curves. As the guiding curves are not parallel, the moving surfaces are non-planar. In (b) the cross sections are forced to be orthogonal to an averaged guiding curve ($\alpha_1 = \alpha_2 = 0.5$), hence the planarity is better preserved.

2.4 Controlling the Shape

Orthogonality condition. In order to ensure a constant shape of the moving surface, it should travel in the normal plane of the n guiding curves $(\mathbf{c}_\ell)_{\ell=1}^n$. If $n = 1$, then all points of the moving surface are expected to travel in the normal plane of the single guiding curve. However, more than one guiding curve may be given, and this condition is not well defined if $n > 1$. We resolve this ambiguity by using the weighted average of the guiding curves.

More precisely, the point $\mathbf{F}(r, s, t)$ is expected to travel in the normal plane of the weighted average $\hat{\mathbf{c}}(r, s, t)$ of guiding curves, see Eq. (13). This is achieved by using the *orthogonality term*

$$f_\mathrm{O}(\mathbf{d}) = \iiint\limits_{[0,1]^3} \left((\mathbf{F}(r,s,t) - \hat{\mathbf{c}}(r,s,t)) \cdot \frac{\partial_t \hat{\mathbf{c}}(r,s,t)}{||\partial_t \hat{\mathbf{c}}(r,s,t)||} \right)^2 \mathrm{d}r\,\mathrm{d}s\,\mathrm{d}t \qquad (15)$$

of the objective function, where ∂_t indicates differentiation with respect to the sweep parameter t. By minimizing this term, the point $\mathbf{F}(r, s, t)$ of the moving surface is restricted to the normal plane of the weighted average (14) of guiding curves.

Note that minimizing (15) produces a volume parameterization where the moving surfaces are as orthogonal as possible to all guiding curves. As their associated tangent directions do not coincide in general, the moving surfaces are forced to deviate from planarity, see 4(a). However, if one wishes to achieve planar cross sections, one may choose the coefficients $\alpha_\ell(r, s)$ as constants. The weighted average (13) then defines a single space curve. The moving surfaces then remain approximately in the normal plane of this curve, see Fig. 4(b).

Rotation minimization. The previous two conditions do not prevent the moving surfaces from rotating around the tangent vectors of the guiding curves. In order to avoid this undesirable twist, we propose to use an additional term which forces the rotation of the parameterization of the sweeping surface to be minimal.

If no rotation around the tangent is present, then the trajectory of a point $\mathbf{F}(r,s,t)$ intersects all moving surfaces orthogonally. Consequently, the rotation vanishes if

$$||\partial_t \mathbf{F}(r,s,t) \times \frac{\partial_t \hat{\mathbf{c}}(r,s,t)}{||\partial_t \hat{\mathbf{c}}(r,s,t)||}||^2 = 0. \tag{16}$$

In order to minimize the rotation along all guiding curves we integrate again over the parameter domain and obtain the *rotation minimizing term*

$$f_{\mathrm{RM}}(\mathbf{d}) = \iiint_{[0,1]^3} ||\partial_t \mathbf{F}(r,s,t) \times \frac{\partial_t \hat{\mathbf{c}}(r,s,t)}{||\partial_t \hat{\mathbf{c}}(r,s,t)||}||^2 \mathrm{d}r\, \mathrm{d}s\, \mathrm{d}t. \tag{17}$$

See [20] for more information on rotation-minimizing frames of space curves.

Shape control. The terms which we introduced so far try to keep the shape of the moving surface constant during the motion along the guiding curves. However, this is not always appropriate, e.g., if the distance between the guiding curves changes during the motion. In this situation, it is desirable to have a tool for controlling the change of the shape during the motion.

We present three methods that allow us to influence the shape (e.g. the ratio of certain lengths) of the moving surface.

1. *Ratio of width and height.* Consider again the reference shape (see (10) and Figure 3), where we choose three points $\mathbf{R}(\hat{r}_j, \hat{s}_j)$, $j = 1,2,3$, such that

$$(\mathbf{R}(\hat{r}_2,\hat{s}_2) - \mathbf{R}(\hat{r}_1,\hat{s}_1)) \times \mathbf{N} = \mu\,(\mathbf{R}(\hat{r}_3,\hat{s}_3) - \mathbf{R}(\hat{r}_1,\hat{s}_1))\,, \tag{18}$$

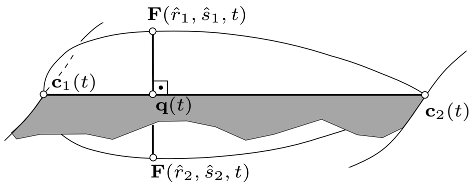
where $\mathbf{N}$ is the unit normal vector of the plane containing the reference shape which points into the direction of the sweep and μ is a real number. Consequently, the three points form a right triangle and the ratio of the lengths of the two legs is equal to μ. We now use the corresponding points of the moving surface in order to define the *first shape term*

$$f_{\mathrm{S}}^{(1)}(\mathbf{d}) = \int_0^1 \|(\mathbf{F}(\hat{r}_2,\hat{s}_2,t) - \mathbf{F}(\hat{r}_1,\hat{s}_1,t)) \times \frac{\partial_t \hat{\mathbf{c}}(\hat{r}_1,\hat{s}_1,t)}{\|\partial_t \hat{\mathbf{c}}(\hat{r}_1,\hat{s}_1,t)\|} - \mu(\mathbf{F}(\hat{r}_3,\hat{s}_3,t) - \mathbf{F}(\hat{r}_1,\hat{s}_1,t))\|^2 \mathrm{d}t. \tag{19}$$

The normal vector $\mathbf{N}$ has been replaced with the unit tangent vector of the guiding curve $\hat{\mathbf{c}}(\hat{r}_1,\hat{s}_1,t)$, which is associated with the apex of the right angle, cf. (13).

2. The *ratio of three collinear points* can be controlled in a similar way. We consider three collinear points $\mathbf{R}(\hat{r}_j,\hat{s}_j)$, $j = 1,2,3$ of the reference shape, which satisfy

$$\mathbf{R}(\hat{r}_3,\hat{s}_3) = \lambda \mathbf{R}(\hat{r}_1,\hat{s}_1) + (1-\lambda)\mathbf{R}(\hat{r}_2,\hat{s}_2). \tag{20}$$



$\mathbf{F}(\hat{r}_1, \hat{s}_1, t)$
$\mathbf{c}_1(t)$
$\mathbf{q}(t)$
$\mathbf{c}_2(t)$
$\mathbf{F}(\hat{r}_2, \hat{s}_2, t)$

Fig. 5. Cross section moving along two guiding curves and associated ruled surface

These three points can be used to define the *second shape term*

$$f_{\mathrm{S}}^{(2)}(\mathbf{d}) = \int_0^1 \|\lambda \mathbf{F}(\hat{r}_1, \hat{s}_1, t) + (1-\lambda)\mathbf{F}(\hat{r}_2, \hat{s}_2, t) - \mathbf{F}(\hat{r}_3, \hat{s}_3, t)\|^2 \mathrm{d}t. \tag{21}$$

3. *Height control.* If the number of guiding curves satisfies $n \geq 2$, then the height of the moving surface can be controlled directly. We consider two points $\mathbf{R}(\hat{r}_1, \hat{s}_1)$ and $\mathbf{R}(\hat{r}_2, \hat{s}_2)$ of the reference shape such that the line segment connecting them intersects the line connecting the two anchors $\mathbf{R}(\tilde{r}_1, \tilde{s}_1)$ and $\mathbf{R}(\tilde{r}_2, \tilde{s}_2)$ orthogonally at a point

$$(1-\beta)\mathbf{R}(\tilde{r}_1, \tilde{s}_1) + \beta \mathbf{R}(\tilde{r}_2 . \tilde{s}_2), \tag{22}$$

Let $\mathbf{N}(t)$ be the unit normal vector of the ruled surface generated by the two guiding curves $\mathbf{c}_1$ and $\mathbf{c}_2$ at the point

$$\mathbf{q}(t) = (1-\beta)\mathbf{c}_1(t) + \beta \mathbf{c}_2(t), \tag{23}$$

see Fig. 5, and let ν be the desired distance of the two points $\mathbf{F}(\hat{r}_1, \hat{s}_1, t)$ and $\mathbf{F}(\hat{r}_2, \hat{s}_2, t)$. The distance can be controlled using the *third shape term*

$$f_S^{(3)}(\mathbf{d}) = \int_0^1 ((\mathbf{F}(\hat{r}_1, \hat{s}_1, t) - \mathbf{F}(\hat{r}_2, \hat{s}_2, t)) \cdot \mathbf{N}(t) - \nu)^2 dt. \tag{24}$$

The length ν can be chosen as a constant, or it can be chosen according to the distance between the two points which are associated with the guiding curves,

$$\nu(t) = \frac{\|(\mathbf{R}(\hat{r}_1, \hat{s}_1) - \mathbf{R}(\hat{r}_2, \hat{s}_2)\|}{\|(\mathbf{R}(\tilde{r}_1, \tilde{s}_1) - \mathbf{R}(\tilde{r}_2, \tilde{s}_2)\|} \|(\mathbf{F}(\tilde{r}_1, \tilde{s}_1, t) - \mathbf{F}(\tilde{r}_2, \tilde{s}_2, t)\| \tag{25}$$

Similarly, in the case of only one guiding curve, one may use t-dependent values of μ and λ, in order to change the ratio of the lengths during the sweep.

Regularity. In order to obtain a regular B-spline volume, we introduce the regularity term

$$f_{\mathrm{R}}(\mathbf{d}) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \sum_{\{k,k+1\} \subset \mathcal{K}} \|\mathbf{d}_{ijk} - \mathbf{d}_{ijk+1}\|^2. \tag{26}$$

Alternatively one may consider

$$f'_{\mathrm{R}}(\mathbf{d}) = \iiint\limits_{[0,1]^3} \|\partial_t \mathbf{F}(r,s,t)\|^2 \mathrm{d}r\, \mathrm{d}s\, \mathrm{d}t. \tag{27}$$

These two terms are related to the lengths of the trajectories, and their minimization leads to shorter trajectories. This may help to avoid unwanted oscillations.

2.5 Variational Design

We define the objective function as a linear combination of the terms

$$f(\mathbf{d}) = \omega_{\mathrm{A}} f_{\mathrm{A}}(\mathbf{d}) + \omega_{\mathrm{O}} f_{\mathrm{O}}(\mathbf{d}) + \omega_{\mathrm{R}M} f_{\mathrm{R}M}(\mathbf{d}) + \omega_{\mathrm{S}}^{(i)} f_{\mathrm{S}}^{(i)}(\mathbf{d}) + \omega_{\mathrm{R}} f_{\mathrm{R}}(\mathbf{d}) \tag{28}$$

with non-negative weights ω_*. The index i specifies the number of the shape term which is used, where the third term can be used only if $n \geq 2$. The weights can be used to control the influence of the individual terms.

In order to simplify the computation, we use numerical integration in order to evaluate the integrals in the objective function and their derivatives with respect to the control points. As a necessary condition for a minimum of (28), the first derivatives of f with respect to all unknowns have to vanish. Since the objective function is a quadratic function of the unknowns $\mathbf{d}$, this yields a linear system of equations for the components of the control points $\mathbf{d}_{ijk}$. Consequently, the solution

$$\mathbf{d}^* = \arg\min_{\mathbf{d}} f \tag{29}$$

of the parameterization problem can be obtained in one step by solving a linear system of equations for the vector of unknowns.

2.6 Examples

We conclude this section of the paper by presenting two examples.

Blade (continued). We continue the blade example and use it to demonstrate the influence of the third shape term. Two parameterizations of a blade-shaped volume are shown in Fig. 6. In this example, we specify only one boundary face of the sweep volume as a boundary condition (the one in the back of the image), and we use two guiding curves. In Fig. 6(a), the ratio of the height to the width of the cross section is increased during the sweep. In Fig. 6(b), the ratio is decreased to zero.

In the previous example, the distance between the two guiding curves was roughly the same for all parameter values t. Now we consider another example, where the distance decreases from 3 to 1.5, i.e. it shrinks by a factor of two. Figure 7(a) shows the behaviour of the cross section without any scaling. For the parameter value $t = 1$, the height is still 1, which corresponds to the original

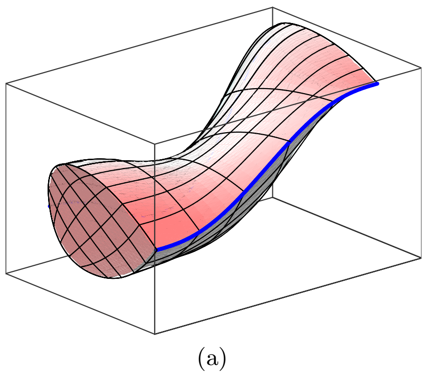


Fig. 6. Controlling the shape of the moving surface during the sweep. The ratio of height and width is doubled in (a) and decreased in (b).

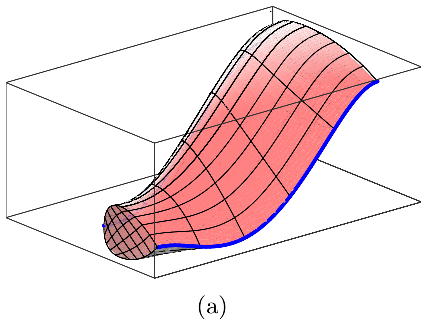


Fig. 7. In (a), the height of the moving surface has not been adapted to the distance of the guiding curves, which causes some distortion. This distortion can be avoided by using the third shape term, as shown in (b).

height. In contrast, in Fig. 7(b) the height decreases and the shape of the moving surface is preserved better.

Table support structure. As another example, which is motivated by a figure in [20], we consider the space curve $\mathbf{c}(t) = (r\cos(t), r\sin(t), \cos(\alpha t)), t \in [0, 2\pi]$ which we use as the guiding curve for a swept volume. Figure 8 shows this curve. The parameter α specifies the number of oscillations of the curve. In our case we choose $\alpha = 4$.

Starting from this curve we want to parameterize the volume that is covered when moving a quadrilateral along the guiding curve $\mathbf{c}(t)$. The parameterization $\mathbf{F}(r, s, t)$ of the volume shall fulfill the conditions described in the previous section.

Note that the guiding curve $\mathbf{c}(t)$ possesses certain symmetries. Hence we parameterize only a segment of the volume between two extremal points and use the symmetries to obtain the entire volume. In order to ensure that the volumes

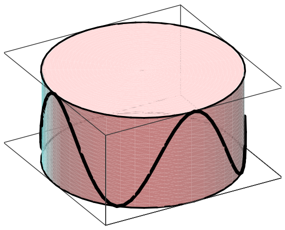
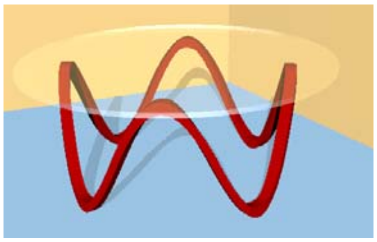



Fig. 8. Support structure of a table (right) and its guiding curve (left)

can be pieced together with C^1 continuity, we prescribe boundary conditions as explained in Section 2.2.

The moving surfaces are parameterized as bilinear patches, while the degree in the sweep direction equals two. After piecing together the rotated segments of the volume we obtain the support structure of the table which is shown in Fig. 8.

3 Isogeometric Analysis for Swept Volumes

In this section, we outline the main features of Isogeometric Analysis, compare it with the classical Finite Element Method (FEM), and then focus on three-dimensional geometries generated by the swept volume technique.

3.1 Weak Form and Geometry Function

Both FEM and Isogeometric Analysis have the same theoretical foundation, namely the weak form of a partial differential equation. For ease of presentation, we consider Poisson's equation

$$-\Delta u = f \quad \text{in } \Omega \tag{30}$$

as an illustrative model problem. Here, $\Omega \subset \mathbb{R}^3$ is a Lipschitz domain with boundary $\partial\Omega$, $f : \Omega \to \mathbb{R}$ is a given source term, and the unknown function $u : \Omega \to \mathbb{R}$ shall satisfy the Dirichlet boundary condition

$$u = u_0 \quad \text{on } \partial\Omega. \tag{31}$$

The discussion of Neumann boundary conditions is postponed to the end of this section.

The weak form of the PDE (30) is obtained by multiplication with test functions v and integration over Ω. More specifically, one defines the function space

$$V := \{v \in H^1(\Omega),\, v = 0 \text{ on } \partial\Omega\}, \tag{32}$$

which consists of all functions $v \in L_2(\Omega)$ that possess weak and square-integrable first derivatives and that vanish on the boundary. For functions $u, v \in H^1(\Omega)$, the bilinear form

$$a(u,v) := \int_{\Omega} \nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x} \tag{33}$$

is well-defined, and even more, it is symmetric and coercive. Setting

$$\langle l, v \rangle := \int_{\Omega} f v \,\mathrm{d}\mathbf{x} \tag{34}$$

as linear form for the integration of the right hand side, the solution $u \in H^1(\Omega)$ is then characterized by the weak form

$$a(u,v) = \langle l, v \rangle \quad \text{for all } v \in V \tag{35}$$

and the boundary condition $u = u_0$ (in the sense of traces).

As a prelude to the idea of Isogeometric Analysis, suppose now that the physical domain Ω is parameterized by a global geometry function

$$\mathbf{F} : \Omega_0 \to \Omega, \quad \mathbf{F}(\boldsymbol{\xi}) = \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{36}$$

Below we will apply NURBS and the swept volume technique to define $\mathbf{F}$, but for the moment the geometry function is simply an invertible C^1-mapping from the parameter domain $\Omega_0 \subset \mathbb{R}^3$ to the physical domain. Integrals over Ω can be transformed into integrals over Ω_0 by means of the well-known integration rule

$$\int_{\Omega} w(\mathbf{x}) \,\mathrm{d}\mathbf{x} = \int_{\Omega_0} w(\mathbf{F}(\boldsymbol{\xi})) \, |\det \, \mathbf{DF}(\boldsymbol{\xi})| \,\mathrm{d}\boldsymbol{\xi} \tag{37}$$

with 3×3 Jacobian matrix $\mathbf{DF}(\boldsymbol{\xi}) = (\partial F_i / \partial \xi_j)_{i,j=1,2,3}$. For the differentiation, the chain rule applied to $u(\mathbf{x}) = u(\mathbf{F}(\boldsymbol{\xi}))$ yields, using a row vector notation for the gradient ∇u,

$$\nabla_{\mathbf{x}} \, u(\mathbf{x}) = \nabla_{\xi} \, u(\boldsymbol{\xi}) \cdot \mathbf{DF}(\boldsymbol{\xi})^{-1}. \tag{38}$$

Summarizing, the integrals in the weak form (35) satisfy the transformation rules

$$\int_{\Omega} \nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x} = \int_{\Omega_0} (\nabla u \, \mathbf{DF}(\boldsymbol{\xi})^{-1}) \cdot (\nabla v \, \mathbf{DF}(\boldsymbol{\xi})^{-1}) \, |\det \, \mathbf{DF}(\boldsymbol{\xi})| \,\mathrm{d}\boldsymbol{\xi} \tag{39}$$

and

$$\int_{\Omega} f v \,\mathrm{d}\mathbf{x} = \int_{\Omega_0} (fv)(\mathbf{F}(\boldsymbol{\xi})) \, |\det \, \mathbf{DF}(\boldsymbol{\xi})| \,\mathrm{d}\boldsymbol{\xi} \, . \tag{40}$$

Obviously, the geometry function, which is in general nonlinear, leads to more complicated expressions in the integrals. We will come back to this point below.

3.2 Galerkin Projection

The Galerkin projection replaces the infinite dimensional space V by a finite dimensional subspace $V_h \subset V$, with the subscript h indicating the relation to a spatial grid. Let $\phi_1, \dots, \phi_n$ be a basis of V_h, then the numerical approximation u_h is constructed as linear combination

$$u_h = \phi_0 + \sum_{i=1}^{n} q_i \phi_i \tag{41}$$

with unknown real coefficients q_i and a given function ϕ_0 that satisfies $\phi_0 = u_0$ on the boundary $\partial\Omega$. Below we will address the issue of boundary conditions in more detail. For the moment, however, we simplify the discussion by assuming $u_0 = 0 = \phi_0$.

Upon inserting u_h into the weak form (35) and testing with $v = \phi_i$ for $i = 1, \dots, n$, one obtains the linear system

$$\mathbf{A}\mathbf{q} = \mathbf{b} \tag{42}$$

with $n \times n$ stiffness matrix $\mathbf{A} = (a(\phi_i, \phi_j))_{i,j=1,\dots,n}$ and right hand side vector $\mathbf{b} = (\langle l, \phi_i \rangle)_{i=1,\dots,n}$. Since the matrix $\mathbf{A}$ inherits the properties of the bilinear form a, it is straightforward to show that $\mathbf{A}$ is symmetric positive definite, and thus the numerical solution $\mathbf{q}$ or u_h, respectively, is well-defined.

In the classical FEM, the subspace V_h consists of piecewise polynomials with global C^0-continuity. It is not appropriate to discuss the FEM in full detail here, but in our context, three features are of particular importance: the concept of nodal bases, local shape functions, and the isoparametric approach, cf. [21].

A finite element mesh in three dimensions consists of grid points or nodes $\mathbf{z}_j$ and tetrahedral or hexahedral elements T_k such that the physical domain Ω is approximated by

$$\Omega_h = \bigcup_k T_k. \tag{43}$$

A nodal basis $(\phi_1, \dots, \phi_n)$ is characterized by the favorable property $\phi_i(\mathbf{z}_j) = \delta_{ij}$, which means that

$$u_h(\mathbf{z}_j) = \sum_{i=1}^{n} q_i \phi_i(\mathbf{z}_j) = q_j. \tag{44}$$

In other words, the coefficient q_j stands for the numerical solution in $\mathbf{z}_j$ and thus carries physical significance. This concept of a nodal basis can be generalized to the partition of unity, which is the property

$$\sum_{i=1}^{n} \phi_i = 1. \tag{45}$$

Shape functions are a very useful technique to unify the treatment of the polynomials in each finite element T_k by the transformation to a reference element T_0. Let ϕ_j be a basis function with support $S \supset T_k$. Restricted to T_k, ϕ_j can be written as a polynomial p_{k_j}, and in case of a nodal basis this polynomial is one at a specific node, say $\mathbf{z}_j$, and zero at all other nodes of T_k. Instead of using $p_{k_j}(\mathbf{x})$ for $\mathbf{x} \in T_k$, the shape function

$$N_l(\boldsymbol{\eta}) = p_{k_j}(\mathbf{x}(\boldsymbol{\eta})) \quad \text{for } \boldsymbol{\eta} \in T_0 \tag{46}$$

allows the evaluation with respect to the reference element T_0. As an example, consider a tetrahedron

$$T_0 = \left\{ \boldsymbol{\eta} \in \mathbb{R}^3 : \, 0 \leq \eta_i \leq 1, \; i = 1, 2, 3, \;\; \text{and} \; \sum_{i=1}^{3} \eta_i \leq 1 \right\} \tag{47}$$

and linear shape functions

$$N_1(\boldsymbol{\eta}) = \eta_1, \;\; N_2(\boldsymbol{\eta}) = \eta_2, \;\; N_3(\boldsymbol{\eta}) = \eta_3, \;\; N_4(\boldsymbol{\eta}) = 1 - \eta_1 - \eta_2 - \eta_3. \tag{48}$$

Correspondingly, the integrations for the assembly of the stiffness matrix and the load vector are carried out by summation over all involved elements and a transformation $\mathbf{G} : T_0 \to T_k$ from the reference element,

$$A_{ij} = a(\phi_i, \phi_j) = \sum_k \int_{T_k} \nabla \phi_i \cdot \nabla \phi_j \, d\mathbf{x} \tag{49}$$

and

$$\int_{T_k} \nabla \phi_i \cdot \nabla \phi_j \, \mathrm{d}\mathbf{x} = \int_{T_0} (\nabla N_m \, \mathbf{DG}(\boldsymbol{\eta})^{-1}) \cdot (\nabla N_l \, \mathbf{DG}(\boldsymbol{\eta})^{-1}) \, |\det \, \mathbf{DG}(\boldsymbol{\eta})| \, d\boldsymbol{\eta}. \tag{50}$$

Though one observes some similarities with the transformation rule (39), it should be stressed that there are two major differences: The integral (50) refers to a single element with simple geometry, and the mapping $\mathbf{G}$ is either linear or, in case of the isoparametric approach, polynomial. Due to the simple structure of the shape functions and the polygonal shape of the elements, the integration of (50) is straightforward and can be implemented in terms of standard quadrature rules or sometimes even via closed form integration.

For the approximation of curved boundaries, the isoparametric approach applies the shape functions both for defining the basis functions and for describing the physical domain. Thus, the mapping $\mathbf{G} : T_0 \to T_k$ from above is written as

$$\mathbf{x} = \mathbf{G}(\boldsymbol{\eta}) = \sum_{l=1}^{L} N_l(\boldsymbol{\eta}) \mathbf{z}_{k_l}, \tag{51}$$

where $\mathbf{z}_{k_l}$ stands for the nodes of the element T_k. In each element, one has therefore the local representation

$$\mathbf{x} = \sum_{l=1}^{L} N_l(\boldsymbol{\eta}) \mathbf{z}_{k_l}, \quad u_h(\mathbf{x}) = \sum_{l=1}^{L} N_l(\boldsymbol{\eta}) q_{k_l} = \sum_{l=1}^{L} N_l(\mathbf{G}^{-1}(\mathbf{x})) q_{k_l}. \tag{52}$$

In practice, isoparametric elements employ quadratic or at most cubic Lagrange-type shape functions, and only edges or faces of elements along a curved boundary are treated this way. In other words, interior element boundaries remain flat faces. Contributions from isoparametric elements in the stiffness matrix are also computed via (50).

While isoparametric finite elements approximate the boundary by a C^0-interpolant, isogeometric analysis exactly represents the boundary by using a geometry description which is directly related to the CAD representation. The basic idea is to formulate the Galerkin projection with respect to basis functions defined on the parameter domain Ω_0 and to use the geometry function $\mathbf{F}$ from (36) as a global push-forward operator to map these functions to the physical domain Ω. Let $(\psi_1, \ldots, \psi_n)$ be a set of linear independent functions on Ω_0. By setting $\phi_i := \psi_i \circ \mathbf{F}^{-1}$, each function is pushed forward to the physical domain Ω. In other words,

$$V_h = \text{span } \{\psi_i \circ \mathbf{F}^{-1}\}_{i=1,\ldots,n} \tag{53}$$

is the finite dimensional subspace for the projection.

Two features are particularly important for an isogeometric method:

(i) The geometry function $\mathbf{F}$ is typically inherited from the CAD description. In this paper, we concentrate on a single patch parameterization in terms of trivariate NURBS, but other options such as volume meshes generated from trimmed surfaces or from T-Spline surfaces are currently under investigation [8].

(ii) The second ingredient is the choice of the functions $\psi_1, \ldots, \psi_n$ for the Galerkin projection. Hughes et. al [1] select those NURBS that describe the geometry, and mesh refinement steps or degree elevation enlarge the subspace while still preserving the original geometry. This is in analogy to the isoparametric approach, but on a global level. However, as long as the geometry function is exact and used as in the transformation rule (39), other choices for $\psi_1, \ldots, \psi_n$ will also preserve the geometry. For instance, one could think of B-Splines instead of NURBS and thus avoid the rational terms.

For swept volumes, the geometry function is of the form

$$\mathbf{F}(\boldsymbol{\xi}) = \mathbf{F}(r,s,t) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} R_{ijk}(r,s,t)\mathbf{d}_{ijk} \tag{54}$$

with trivariate NURBS R_{ijk} defined on the patch $\Omega_0 = [0,1]^3$ and control points $\mathbf{d}_{ijk} \in \mathbb{R}^3$. Like in [1], we use the same functions R_{ijk} as basis functions and thus have

$$V_h \subset \text{span } \{R_{ijk} \circ \mathbf{F}^{-1}\}_{i\in\mathcal{I}, j\in\mathcal{J}, k\in\mathcal{K}} \,. \tag{55}$$

Note that the boundary condition $u = u_0$ has also to be taken into account, and for this reason we write V_h as a subset of the span. Accordingly, the numerical solution is given by

$$u_h(\mathbf{x}) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} R_{ijk}(\mathbf{F}^{-1}(\mathbf{x}))\, q_{ijk} \tag{56}$$

where some of the coefficients q_{ijk} are determined from the boundary condition.

A comparison with isoparametric finite elements leads to the following observations:

(i) The knot vectors partition the patch into a computational mesh, and adopting the finite element terminology, we can call three-dimensional knot spans also elements (in the parameter domain). However, the support of the basis functions is in general larger than in the FEM case.
(ii) The NURBS do not form a nodal basis, and thus single coefficients q_{ijk} do not represent approximations in specific grid points. On the other hand, the partition of unity property (45) is satisfied.
(iii) Depending on the chosen degree and the knot multiplicity in the NURBS data, global smoothness of class C^1 or higher is easily achieved in Isogeometric Analysis.

Note also that both the FEM and Isogeometric Analysis coincide for an important special case. For degree $p = 1$ in all three coordinate directions, the geometry function (54) generates a regular assembly of hexahedral finite elements, and the corresponding R_{ijk} reduce to trilinear basis functions in each element. Thus, the wide-spread trilinear hexahedral finite element is part of Isogeometric Analysis.

While the idea of Isogeometric Analysis is impressive, its actual implementation requires additional effort in order to come up with powerful algorithms. For this reason, we address some of the major issues in the following.

3.3 Boundary Conditions, Quadrature, Refinement

In standard FEM, the treatment of Dirichlet boundary conditions is greatly simplified by the nodal basis property. Recall Poisson's equation (30) and the boundary condition $u = u_0$ on $\partial\Omega$. For inhomogeneous $u_0 \neq 0$, the numerical solution is split into $u_h = \phi_0 + \sum_{i=1}^{n} q_i \phi_i$, and the function ϕ_0 has to be chosen such that $\phi_0 = u_0$ on $\partial\Omega$. Now let $\mathbf{z}_j$ for $j = 1, \ldots, m$ denote all nodes on the boundary and χ_j the corresponding basis function with $\chi_i(\mathbf{z}_j) = \delta_{ij}$. By construction,

$$\phi_0 := \sum_{j=1}^{m} u_0(\mathbf{z}_j)\chi_j \tag{57}$$

interpolates u_0 in the nodes and is thus an appropriate choice for incorporating the boundary condition.

Based on the interpolation (57), the next steps are straightforward. One inserts u_h in the weak form and moves the term involving ϕ_0 to the right hand side. While the definition of the stiffness matrix in (42) remains the same as for zero boundary conditions, the load vector is modified to

$$\mathbf{b} = (\langle l, \phi_i\rangle - a(\phi_0, \phi_i))_{i=1,\ldots,n}. \tag{58}$$

This direct incorporation of boundary conditions is usually performed at the linear algebra level, i.e., the stiffness matrix is first generated including the nodes on the boundary as additional degrees of freedom, and then the matrix is condensed and the contributions from ϕ_0 are moved to the right hand side.

Two alternatives are also common in FEM codes. In both cases, the basis for generating the stiffness matrix includes the nodes on the boundary, i.e.,

$$\tilde{V}_h = \text{span}\,\{\phi_1, \ldots, \phi_n, \chi_1, \ldots, \chi_m\} \tag{59}$$

is used in the Galerkin projection, which corresponds to

$$u_h = \sum_{j=1}^{m} w_j \chi_j + \sum_{i=1}^{n} q_i \phi_i. \tag{60}$$

The interpolation conditions $w_j = u_0(\mathbf{z}_j)$ are then explicitly enforced. The first alternative simply replaces the rows and columns that belong to the w_j coefficients by ones on the diagonal and zeros elsewhere. The corresponding right hand side entries are set to $u_0(\mathbf{z}_j)$. This leads to an enlarged stiffness matrix of dimension $(n+m) \times (n+m)$.

The second alternative is related to the concept of weak boundary conditions. The equations $w_j = u_0(\mathbf{z}_j)$ can be viewed as m linear constraints for the node vector

$$\tilde{\mathbf{q}} := (q_1, \ldots, q_n, w_1, \ldots, w_m)^\top. \tag{61}$$

In matrix-vector notation, this is equivalent to

$$\mathbf{B}\tilde{\mathbf{q}} = \mathbf{c} \tag{62}$$

with an $m \times (m+n)$ Boolean matrix $\mathbf{B}$ and a right hand side vector $\mathbf{c}$ determined by $c_j := u_0(\mathbf{z}_j), j = 1, \ldots, m$. Overall, the constraint (62) combined with the discretized weak form results in the linear system

$$\begin{pmatrix} \tilde{\mathbf{A}} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{q}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}, \tag{63}$$

which has a saddle point structure. The additional unknowns $\boldsymbol{\lambda} \in \mathbb{R}^m$ are discrete Lagrange multipliers. Note that the $(n+m) \times (n+m)$ matrix $\tilde{\mathbf{A}}$ is generated from the Galerkin projection with enlarged space $\tilde{V}_h$.

In total, this second alternative yields thus a system of $n+2m$ linear equations, which seems rather expensive. However, this approach is the most flexible one since it can be extended to a weak formulation

$$\int_{\partial\Omega} (u - u_0)\mu \, d\mathbf{s} = 0 \quad \text{for all } \mu \in M_h \tag{64}$$

of the boundary conditions. Such a weak formulation is of great advantage in case of coupling conditions for multi-physics and multi-domain problems, and it is also closely related to domain-decomposition methods. Compare also [4] on the advantages of weak boundary conditions in fluid mechanics applications. However, it should be stressed that the choice of the space M_h for the test functions μ in (64) requires some care, cf. the inf-sup condition in mixed and hybrid finite element methods.

If we consider the above techniques in combination with Isogeometric Analysis and the swept volume meshes, it turns out that the lack of a nodal basis is a drawback and renders the incorporation of boundary conditions more involved. More specifically, zero Dirichlet boundary conditions are the easiest case and simply require the determination of those basis functions R_{ijk} that do not vanish on the boundary. The corresponding solution coefficients q_{ijk} are then set to zero, which can be accomplished by the first alternative above. Non-zero boundary conditions $u = u_0$, however, demand additional measures and will be subject of future work. One option is to derive an analogue to interpolation (57) in terms of the basis functions. Another one is to use quasi-interpolation operators that project the boundary condition into the spline space. Alternatively, the weak form (64) can be used to discretize the boundary in a more general way.

If we have to deal with mixed Dirichlet and Neumann boundary conditions, i.e.,

$$u = u_0 \quad \text{on } \Gamma_D, \qquad \frac{\partial u}{\partial \mathbf{n}} = h \quad \text{on } \Gamma_N \tag{65}$$

with outward normal vector $\mathbf{n}$ and $\partial\Omega = \Gamma_D \cup \Gamma_N$, the situation is basically the same. One first determines the knot spans in the patch Ω_0 that correspond to Γ_D and Γ_N and then identifies the non-vanishing NURBS that are involved. The Neumann boundary condition yields an additional surface integral in the weak form (35), which can be subsumed under the linear form on the right hand side via

$$\langle l, v\rangle := \int_\Omega f v \,\mathrm{d}\mathbf{x} + \int_{\Gamma_N} h v \, d\mathbf{s}. \tag{66}$$

Accordingly, the load vector $\mathbf{b}$ is computed by projection of (66).

As discussed in the very beginning of this section, the evaluation of integrals over Ω can be replaced by integrals over the parameter domain Ω_0 via the transformation rules (39) and (40). More specifically, consider the right hand side vector $\mathbf{b}$, whose entries consist of integrals

$$b_l = \int_\Omega f \phi_l \,\mathrm{d}\mathbf{x} = \int_{\Omega_0} f(\mathbf{F}(\boldsymbol{\xi}))\phi_l(\mathbf{F}(\boldsymbol{\xi}))\, |\det \mathbf{DF}(\boldsymbol{\xi})| \,\mathrm{d}\boldsymbol{\xi}\,. \tag{67}$$

Now assume that the parameter domain is discretized into a mesh

$$\Omega_0 = \bigcup_\kappa B_\kappa \tag{68}$$

with hexahedral elements B_κ that are defined by the knot spans in the three coordinate directions. Then, as in the finite element approach (49), the integral (67) is split into

$$b_i = \sum_\kappa \int_{B_\kappa} f(\mathbf{F}(\boldsymbol{\xi}))\psi_l(\boldsymbol{\xi})\, |\det \mathbf{DF}(\boldsymbol{\xi})| \,\mathrm{d}\boldsymbol{\xi}\,. \tag{69}$$

In the case of swept volumes, the basis functions ψ_ℓ are the tri-variate NURBS R_{ijk}, and numerical quadrature is employed to approximate the integrals, see [11] for a discussion of specific quadrature rules. In this context, it is important to take both the larger support of the basis functions and the increased smoothness into account, which means that the Gaussian quadrature rules used in the standard FEM are not optimal in Isogeometric Analysis. The same reasoning applies to the computation of the stiffness matrix.

Finally, we shortly comment on the options for refining the grid in Isogeometric Analysis. The overall goal of refinement is to enlarge the finite dimensional subspace V_h in a step-by-step procedure, which leads to a sequence

$$V_h = V_1 \subset V_2 \subset V_3 \subset \ldots, \tag{70}$$

and in the limit $V_i \to V$. The process of h-refinement consists of inserting additional knots in the computational mesh. For swept volumes, this means that in one or several of the coordinates r, s, t the corresponding knot vectors are refined. This step changes the representation of the geometry function $\mathbf{F}$ but leaves the geometry and the mapping between Ω_0 and Ω invariant, which is a very important property. Due to the tensor product structure, h-refinement has always a global effect on the mesh. For recent work on T-Splines, which allow local refinement, see [8].

As alternative to h-refinement, p-refinement increases the polynomial degree in each element B_κ. More precisely, if the initial mesh and geometry description is given in terms of piecewise linear functions, $p = 1$, then p-refinement increases the local smoothness inside each element but leaves the global continuity unchanged, which means that multiple knots are used in the refinement process. When starting with piecewise linear functions the knots of a degree p NURBS are repeated $p-1$ times, and there is basically not much difference to the classical p-FEM. The combination of both degree elevation and knot insertion (in this order) with increased smoothness is called k-refinement. Due to the embedding (70), the original continuity properties at the knots of the mesh belonging to V_1 have to be preserved also in k-refinement, but at the additional knots the smoothness is C^{p-1} for NURBS of degree p.

4 Simulation Examples

Instead of Poisson's equation (30), we study in this section the deformation of three dimensional solids under the assumption of linear elasticity. All geometries are generated by the swept volume method and then used as input data for the experimental isogeometric solver of Hughes at al. [1]. In the simulations, the displacement field $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}), u_3(\mathbf{x}))^\top \in \mathbb{R}^3$ is the unknown quantity, and it satisfies the equilibrium equations

$$\text{div } \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{f} \tag{71}$$

with given volume load $\mathbf{f}$. Hooke's law

$$\boldsymbol{\sigma}(\mathbf{u}) = \lambda(\text{trace } \boldsymbol{\epsilon}(\mathbf{u}))\mathbf{I} + 2\mu\boldsymbol{\epsilon}(\mathbf{u}) \tag{72}$$

defines the stress tensor $\boldsymbol{\sigma}$ in terms of the strain tensor

$$\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\top}) \tag{73}$$

and the Lamé constants λ and μ as material parameters. These parameters are related to Young's modulus E and Poisson ratio ν by

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}. \tag{74}$$

In the weak form (35), the bilinear form a is then replaced by

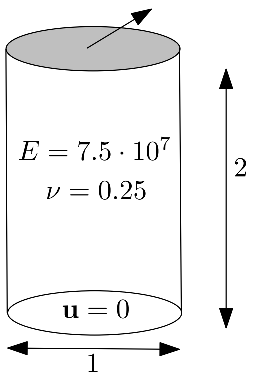
$$a(\mathbf{u},\mathbf{v}) := \int_{\Omega} \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{u})\, d\mathbf{x} \tag{75}$$

with tensor product $\boldsymbol{\sigma} : \boldsymbol{\epsilon} = \text{trace } (\boldsymbol{\sigma}\boldsymbol{\epsilon})$.

4.1 Simulation of Swept Volumes

Cylinder. We start with the study of a cylinder with zero-Dirichlet boundary condition at the base and a surface force in y-direction at the top, see Fig. 9. We distinguish two different possibilities of parameterizing the cylinder shown in Fig. 1. In Section 4.2 we will investigate the effects of the different parameterizations on the numerical simulation in more detail.

However, the plots of the simulation results for both parameterizations give identical results, which are displayed in Fig. 10.



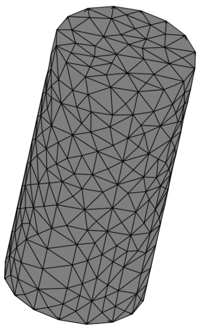


Fig. 9. Left: Cylinder geometry and boundary conditions. Right: COMSOL discretization.

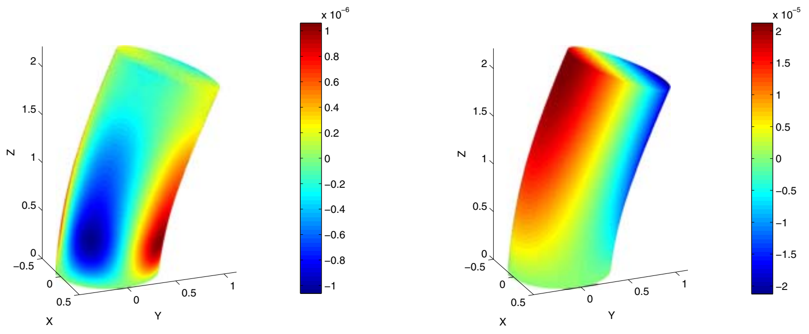


Fig. 10. Displacement components of the solution in x- (left) and z-direction (right)

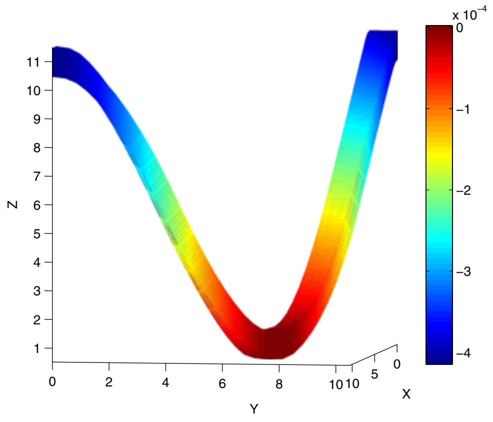


Fig. 11. The table example: Components of the solution in z-direction

Table. Our next simulation example is the supporting structure of a table already introduced in 2.6. Due to the rotational symmetry of the structure we only need to simulate a quarter of it. At the faces on the symmetric planes we apply symmetric boundary conditions, that means zero Dirichlet boundary conditions for displacement orthogonal to the face and zero Neumann boundary conditions for displacement parallel to the plane. The numerical result for the displacement into z-direction can be seen in Fig. 11.

Blade. This example is based on a blade-shaped NURBS volume which has been generated with the help of the techniques in Section 2. The blade is subject to a volume load in the central segment and to zero displacement boundary conditions on the right-hand side. The result of the numerical simulation is visualized in Fig. 12.

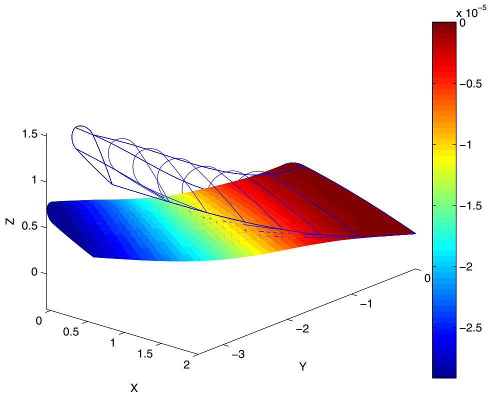


Fig. 12. The blade example: The figure shows the initial shape (wireframe) and the deformed solid (colored), where the displacement has been magnified by a factor of $2.5 \cdot 10^4$. The coloring represents the components of the solution in z-direction.

4.2 Experimental Comparison with a Traditional Simulation Tools

We now compare our results with a numerical approximation obtained by a conventional FEM package. We use COMSOL [22] in combination with linear and quadratic isoparametric tetrahedral elements. A discretization of a cylinder by tetrahedra is shown in Fig. 9. The original geometry is therefore approximated by piecewise polynomials and also changes in every refinement step. For the isogeometric simulation we use a triquadratic parameterization which we already introduced in Section 4.1. This justifies the direct comparison with the quadratic isoparametric approach.

In order to compare the different approximations obtained, we calculate the energy norm

$$||\mathbf{u}_h||_E = \sqrt{\mathbf{q}^\top \mathbf{A}\mathbf{q}} \tag{76}$$

of the numerical approximation. The norm $||\mathbf{u}_h||_E$ is plotted in Fig. 13. As can be seen, by refining the grids all numerical solutions tend to the same maximum value. This confirms the theoretical result that $||\mathbf{u}_h||_E \to ||\mathbf{u}||_E$ from below for any convergent Galerkin projection method. The speed of this convergence may serve as an indicator of the convergence behaviour of the method and can be used to compare different methods.

Note that the discretizations using isoparametric quadratic elements behave better than the one using only linear elements. The second cylinder parameterization and its refinements in comparison show a similar behavior to the quadratic isoparametric approach. Remarkably, the first cylinder parameterization uses significantly fewer degrees of freedom than the other two.

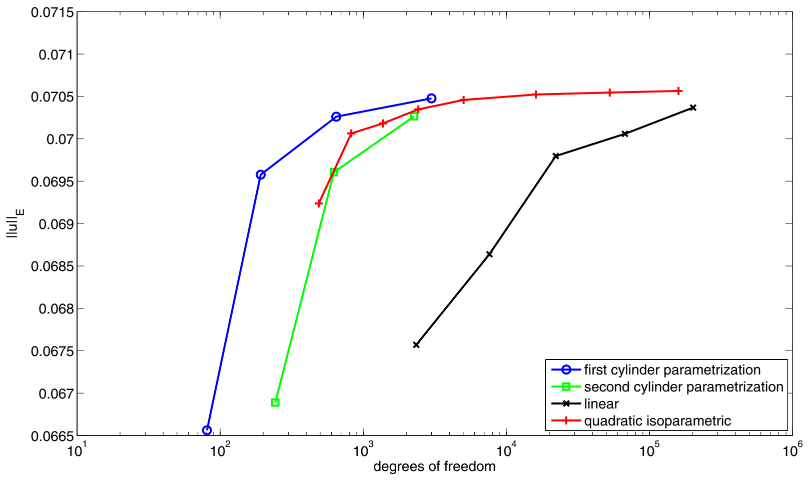


Fig. 13. Comparison between Isogeometric Analysis and Standard-FEM

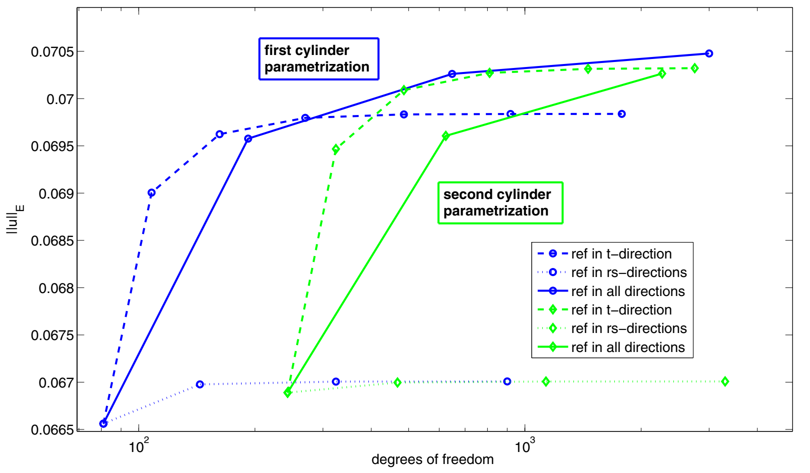


Fig. 14. Comparison between different refinement directions

In Fig. 14 we compare different refinement strategies applied to the two cylinder parameterizations. The rs-parameter directions are parallel to the xy-plane and the t-direction is equal to the z-direction in space coordinates. As expected, the refinement in t-direction strongly affects the energy norm due to the fact that the displacement varies more strongly in this direction than in the other directions.

5 Conclusion

For using isogeometric analysis, the creation of trivariate volumetric NURBS representations is a great challenge. The swept volume framework presented here provides a means for generating, optimizing and refining such volume meshes. Swept volumes lead to a single patch description of the geometry, which can be used to set up and perform an isogeometric simulation. In view of the preliminary numerical results discussed above, the selection of the parameterization deserves specific attention. Since an adept choice of the parameterization leads to significant savings in the number of degrees of freedom required to achieve a certain precision of the numerical solution, there is a clear connection to the approximation properties of the NURBS basis functions in the Galerkin projection. However, at the moment we have no measure to assess or predict the quality of the parameterization in this respect. The numerical results also indicate that isogeometric analysis is a competitive approach as compared to standard FEM with isoparametric quadratic elements.

Acknowledgment. The authors were supported by the 7th Framework Programme of the European Union, project SCP8-218536 “EXCITING”. Special thanks go to Hughes et al. [1] for providing their isogeometric implementation, which was used to generate the test results in this paper. We are also grateful to our student C. Götz for his preliminary work on 3D NURBS volume meshes.

References

1. Hughes, T.J.R., Cottrell, J.A., Bazilevs, Y.: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer methods in applied mechanics and engineering 194, 4135–4195 (2005)
2. Auricchio, F., Beirão da Veiga, L.B., Buffa, A., Lovadina, C., Reali, A., Sangalli, G.: A fully ”locking-free” isogeometric approach for plane linear elasticity problems: A stream function formulation. Computer methods in applied mechanics and engineering 197, 160–172 (2007)
3. Bazilevs, Y., Beirão da Veiga, L., Cottrell, J.A., Hughes, T.J.R., Sangalli, G.: Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes. Mathematical Methods and Models in Applied Sciences 16, 1031–1090 (2006)
4. Bazilevs, Y., Hughes, T.J.R.: Weak imposition of Dirichlet boundary conditions in fluid mechancis. Computer & Fluids 36, 12–26 (2007)
5. Bazilevs, Y., Calo, V.M., Hughes, T.J.R., Zhang, Y.: Isogeometric fluid-structure interaction: theory, algorithms, and compuations. Computational Mechanics 43, 3–37 (2008)
6. Cottrell, J.A., Hughes, T.J.R., Reali, A.: Studies of refinement and continuity in isogeometric structural analysis. Computer methods in applied mechanics and engineering 196, 4160–4183 (2007)
7. Cottrell, J.A., Reali, A., Bazilevs, Y., Hughes, T.J.R.: Isogeometric analysis of structural vibrations. Computer methods in applied mechanics and engineering 195, 5257–5296 (2006)

8. Dörfel, M.R., Jüttler, B., Simeon, B.: Adaptive isogeometric analysis by local h-refinement with T-splines. Computer methods in applied mechanics and engineering (2008) (in press)
9. Elguedj, T., Bazilevs, Y., Calo, V.M., Hughes, T.J.R.: $\overline{B}$ and $\overline{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements. Computer methods in applied mechanics and engineering 197, 2732–2762 (2008)
10. Hughes, T.J.R., Reali, A., Sangalli, G.: Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS. Computer methods in applied mechanics and engineering 197, 4104–4124 (2008)
11. Hughes, T.J.R., Reali, A., Sangalli, G.: Efficient quadrature for NURBS-based isogeometric analysis. Computer methods in applied mechanics and engineering (2009) (in press)
12. Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C.L., Hughes, T.J.R.: Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Computer methods in applied mechanics and engineering 196, 2943–2959 (2007)
13. EXCITING: exact geometry simulation for optimized design of vehicles and vessels. EU Project SCP8-2007-GA-218536, http://www.exciting-project.eu
14. Hoschek, J., Lasser, D.: Fundamentals of Computer Aided Geometric Design. AK Peters, Wellesley (1996)
15. Chang, T.I., Lee, J.H., Kim, M.S., Hong, S.J.: Direct manipulation of generalized cylinders based on b-spline motion. The Visual Computer 14(5), 228–239 (1998)
16. Chuang, J.H., Ahuja, N., Lin, C.C., Tsai, C.H., Chen, C.H.: A potential-based generalized cylinder representation. Computers & Graphics 28(6), 907–918 (2004)
17. Martin, T., Cohen, E., Kirby, R.: Volumetric parameterization and trivariate B-spline fitting using harmonic functions. Computer Aided Geometric Design (2009) (in press)
18. Piegl, L.A., Tiller, W.: The Nurbs Book. Springer, Heidelberg (1995)
19. Floater, M.: Mean value coodinates. Computer Aided Geometric Design 20(11), 19–27 (2003)
20. Wang, W., Jüttler, B., Zheng, D., Liu, Y.: Computation of rotation minimizing frames. ACM Trans. on Graphics 27(1), article no. 2 (2008)
21. Hughes, T.: The Finite Element Method. Prentice Hall, Englewood Cliffs (1987)
22. COMSOL multiphysics user manual, version 3.4 (2007)

Numerical Checking of C^1 for Arbitrary Degree Quadrilateral Subdivision Schemes

Ursula H. Augsdörfer[1], Thomas J. Cashman[1], Neil A. Dodgson[1], and Malcolm A. Sabin[2]

[1] The Computer Laboratory, University of Cambridge, UK
[2] Numerical Geometry Ltd., UK
{uha20,tc270,nad10,mas33}@cl.cam.ac.uk

Abstract. We derive a numerical method to confirm that a subdivision scheme based on quadrilateral meshes is C^1 at the extraordinary points. We base our work on Theorem 5.25 in Peters and Reif's book "Subdivision Surfaces", which expresses it as a condition on the derivatives within the characteristic ring around the EV. This note identifies instead a sufficient condition on the control points in the natural configuration from which the conditions of Theorem 5.25 can be established.

1 Introduction

It is important, when deriving a subdivision scheme, to ensure that the scheme is C^1 everywhere. If the scheme is derived from a tensor product or box-spline the level of continuity can be determined by reference to the underlying basis functions. For other schemes algebraic methods can be applied to determine the level of continuity in the regular regions [1]. The situation changes at extraordinary points, where the analysis must be extended to a more general topology [2,3].

In this note we demonstrate a simple numerical check of C^1 continuity which was derived to test tensor product subdivision schemes of arbitrary degree as described by Cashman *et al.* [4]. It can be used for any subdivision scheme with a stationary subdivision matrix. The check is based on work by Peters and Reif [3] and requires analysis of the characteristic ring which is determined in terms of the support of a subdivision scheme. We first explain the context and then show how C^1 can be proved.

We do not claim priority for these results. However, they do not seem to be in the literature except implicitly in papers by Zorin [5] and Umlauf [6,7], which deal primarily with schemes whose regular regions are triangle grids. This note will help others who have still to prove the same property of their own schemes.

2 Support

The *support region* is the region in the limit surface of a control point which is modified when that control point moves. For a univariate B-spline of degree d, the support width, w, equals $(d + 1)/2$ spans of the control polygon to each

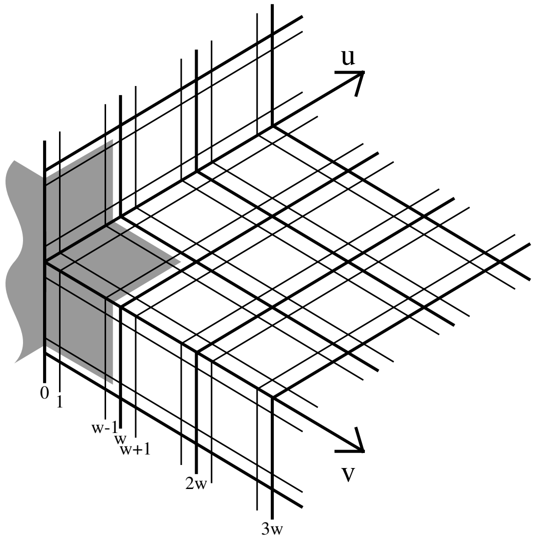


Fig. 1. A schematic figure for a grid around a vertex of arbitrary valency. Thick lines indicate the width, w, of the support of the vertex. Thin lines are boundaries of patches in the limit surface. The support of the vertex is shown schematically highlighted in grey.

side of the control point. For a regular bivariate quadrilateral scheme the support region is the interior of a square of side $2w$. For a vertex of valency n, the support region is the interior of an n-gon of side $2w$.

The support region of an extraordinary vertex (EV) is also the region for which changes in the coefficients associated with it directly influence the shape of the limit surface.

The area around an EV with valency n is discussed in terms of n sectors which are indexed from 1 to n. Figure 1 shows one sector around a vertex, which we assume to be extraordinary, together with partly drawn neighbouring sectors. The support of the vertex is shown schematically highlighted in grey. The thick lines indicate the extent of the support of the vertex. The thin lines can be viewed as control polyhedron edges or as boundaries of patches in the limit surface. The support of the EV falls short of the thick lines, because the corresponding limit surface lines are not influenced by the EV position.

3 The Characteristic Ring

Just outside the support region of the EV the limit surface can be seen to consist of pieces of polynomials of bi-degree d, because each piece sees only a regular configuration of control points.

By eigenanalysis we can determine from the coefficients of the scheme a *natural configuration* [8] which is derived from the column eigenvectors of the two subdominant eigenvalues.

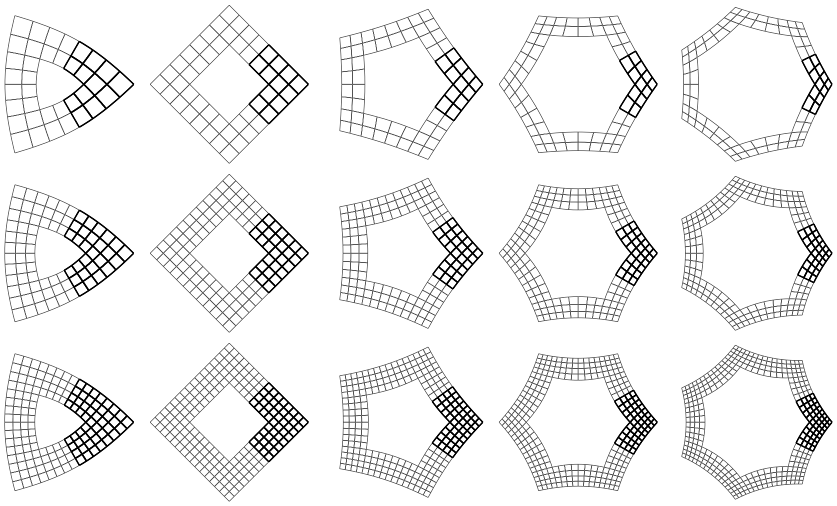


Fig. 2. Normalised characteristic rings for valencies 3 to 7 (left to right) for schemes of degrees 3, 5 and 7 (top to bottom). The first sector is shown using thick lines.

With each refinement of the natural configuration the support of the EV reduces by a factor of 2, exposing a new ring of regular B-spline surface. These spline rings are called characteristic rings [2], and each is a scaled copy of the next one out. We can therefore define the limit surface shape around the EV as being made up of rings of spline pieces.

A characteristic ring is referred to as *normalised* if it is centred on the origin and oriented in such a way that the furthest corner in the first sector is at $(1, 0)$ of the global (x, y)-coordinate system, as shown in Figure 2 for different valencies and different degree schemes. This ring is w bi-polynomial patches thick.

For schemes such as the original Catmull-Clark [9] where the influence of the extraordinary vertex on its neighbours is not modified, the characteristic ring can be thinner, and further in. However, because we regard bounded curvature as essential [10], and this demands modification of the influence of the extraordinary vertex on its new neighbours, we consider only schemes which have this larger value.

The exact boundaries of the characteristic ring are shown more clearly in Figure 3 for the first sector. As before, the thick lines indicate the support w and the thin lines are boundaries of the polynomial pieces.

4 Checking C^1 Continuity

For a scheme to be C^1, the characteristic ring has to be regular and injective [3].

Peter and Reif's Theorem 5.25 [3] states that, given a scheme that is symmetric under both rotation and reflection, then these conditions for C^1 are met if,

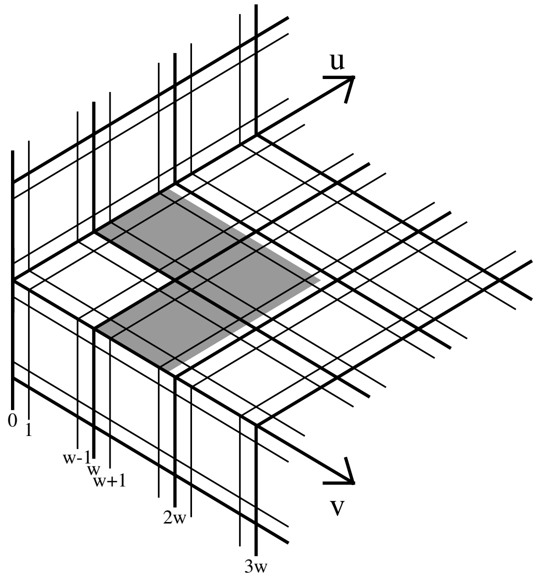


Fig. 3. A schematic figure like Figure 1. The characteristic ring is highlighted in grey for the first sector.

in the first sector of the normalised characteristic ring, the first derivative in the u direction of the local coordinate system is directed within the first quadrant of the global coordinate system.

It is therefore sufficient to prove that the first derivative in the u direction everywhere in the shaded region in Figure 3 has positive x and y components.

Determining the derivative everywhere is an infinite calculation. We have to relate the calculations to something finite. In principle this could be symbolic or algebraic, but the practical approach which we take is to identify a numerical procedure which can be applied to any subdivision scheme. Because each valency has its own set of EV coefficients, this means that each valency is processed separately.

To establish a numerical proof for C^1, we use two facts:

1. The first derivative in u is itself the tensor product of B-splines of degree d in v and $d-1$ in u, with control points being the first differences in the u-direction of the points of the natural configuration.
2. This B-spline satisfies the convex hull property and thus all points lie within the convex hull of its control points.

The derivatives we need are therefore bounded by the convex hull of the first differences of some region of the original polyhedron. If all first differences within that region lie in the first quadrant, then so does their convex hull, and so do the derivatives at all points of the first sector of the spline ring.

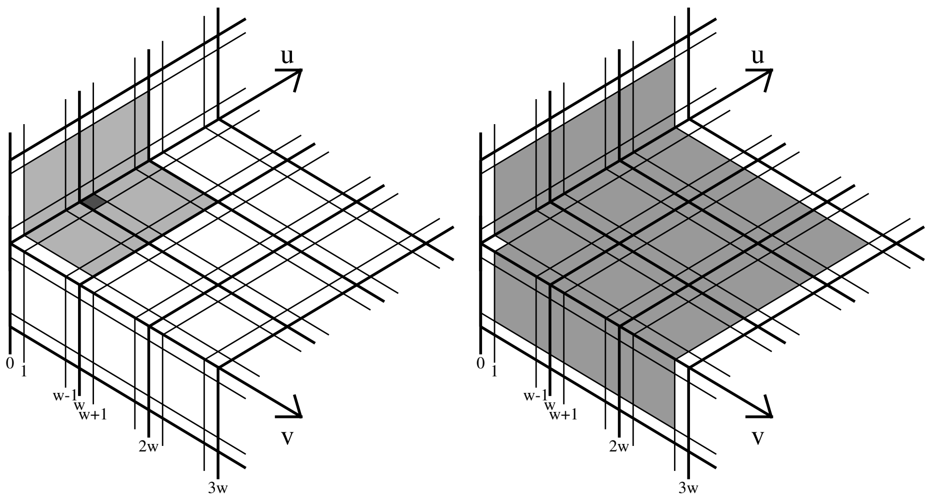


Fig. 4. Schematic figures like Figure 1. Left: One bi-polynomial patch (dark grey), and the set of control vertices which influence the first derivatives within it (light grey). Right: The set of control points influencing the first derivatives in all patches of the spline ring are coloured grey.

5 Region of Analysis

To establish the region of the natural configuration for which differences have to be taken, we resort again to support region arguments.

The bi-polynomial patch, which is dark-shaded in Figure 4 on the left, is influenced by the vertices of the $2w+1 \times 2w+1$ region, which is light-shaded. Therefore, the derivative in this region is influenced by the first differences in this part of the polyhedron.

If we draw similar diagrams for all of the polynomial patches in the first sector of the spline ring, and take the union, we find that the first differences we have to consider are those of the control points in, or on the edge of, the shaded regions shown in Figure 4 on the right. That is, the characteristic ring extended by a border of width w, but excluding the boundary of the extended region.

6 Sharper Bounds for the Region of Analysis

If the EV has high enough valency, analysing the above described region fails to provide the required proof, even when the scheme is in fact C^1. That is, it is a sufficient condition but not a necessary condition.

An example of this problem is shown in Figure 5. On the left, the EV has valency $n = 5$. The points shown as circles all lie within the region of analysis for a bounded curvature variant of the Catmull-Clark subdivision scheme. Computation shows that all differences of these points in the u direction of the local

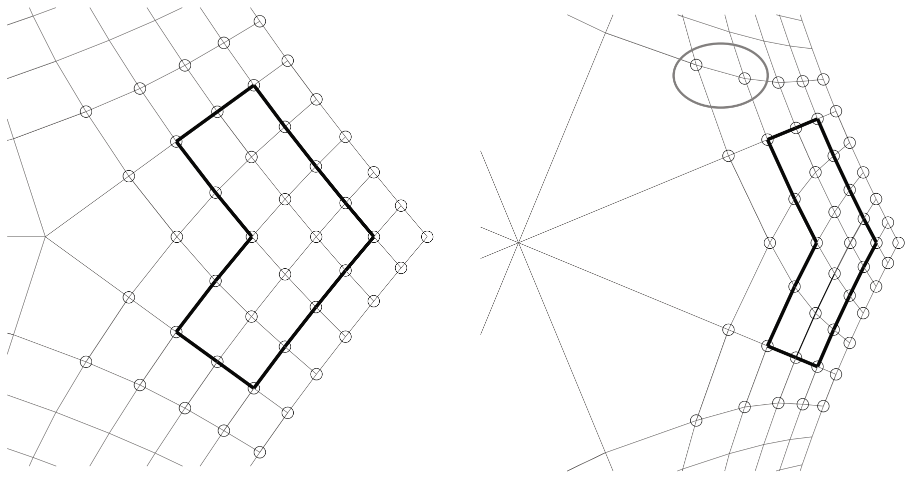


Fig. 5. Left: Part of the mesh around an EV of valency $n = 5$ for a bounded curvature variant of the Catmull-Clark scheme (degree 3). The characteristic ring in the first sector lies within the thick dark lines. Control points influencing the first derivatives in this region and required for the analysis are shown as circles. Right: The mesh around an EV of valency $n = 8$. The vertices for which differences are negative are encircled in grey.

coordinate system are positive. However, for an EV of valency $n = 8$, shown on the right, not all differences are positive. The first difference in the u-direction of vertices encircled in grey has clearly a negative y component, despite the scheme being C^1.

This discrepancy is because within the region of analysis, described in Section 5, lie pieces of mesh well beyond the first sector of the characteristic ring, and the curvature of the grid is sufficient to push these first differences outside the acceptable range.

This can be countered by using sharper bounds, which lie tighter around the first sector of the characteristic ring.

In order to obtain a tighter bound for any subdivision scheme, the region of analysis is subdivided. Because this involves only the regular regions it does not require the implementation of non-regular stencils. However, the EV has had its effect taken into account because we are looking at an eigenvector of the scheme around the EV.

In Figure 6 the region of analysis is shown after one and two subdivision steps (centre and right) next to the original region (left). Vertices which form part of the region of analysis now lie tighter around the first sector, all differences are positive and the scheme is thus proved to be C^1.

For high valencies more than one step of subdivision may be necessary. Up to valency 50 and up to a degree 19 bounded curvature tensor product subdivision scheme a maximum of seven subdivision steps were required.

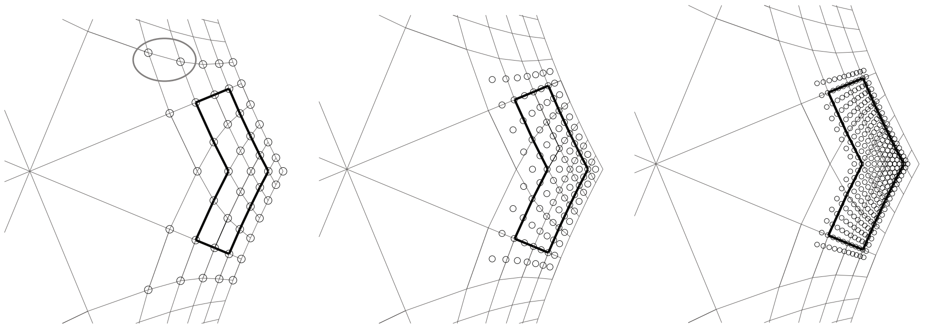


Fig. 6. Left: Part of the mesh around an EV of valency $n = 8$ for a bounded curvature variant of the Catmull-Clark scheme. Control points influencing the first derivatives in the first sector and required for the analysis are shown as circles. Centre: After one step of subdivision. Right: After two steps of subdivision, the newly introduced vertices lie tighter around the first sector of the characteristic ring. All differences are now positive and the scheme provably C^1.

If the confirmation of C^1 does not emerge from this procedure after ten subdivision steps it is worth evaluating the derivative at a few places to see whether a disproof by example can be found. The concave corner of the spline ring sector is a good first choice. This would catch the misbehaviour so carefully constructed in [3], Figure 6.9.

An alternative to this refinement would be to convert the bi-polynomial pieces of the spline ring to Bézier form. This would give tight bounds without iteration, but does not generalise easily beyond B-splines.

7 Simplification of the Test

Using subdivision means we can simplify the test further.

The refined natural configuration in the characteristic ring is a scaled down version of the original natural configuration of the next ring out. Consider the first bi-polynomial patches at the inner edge of the spline ring. The refined polyhedron uses only the vertices of or inside the spline ring, and is given by a convex combination. We can therefore be sure that if the characteristic ring itself satisfies our test, the first two rings of control points outside it, which are a scaled-up copy of the refined grid, will also satisfy it. This argument ripples outward recursively until it covers the region we were originally scanning.

Therefore, we do not need to consider any control vertices outside the characteristic ring, reducing the area which needs to be considered to that shaded in Figure 7 on the left. An example of the region of analysis for degree 9 subdivision scheme, with a support width $w = 5$, is shown in Figure 7 on the right. The higher the degree of the subdivision scheme the more can be gained from this simplification.

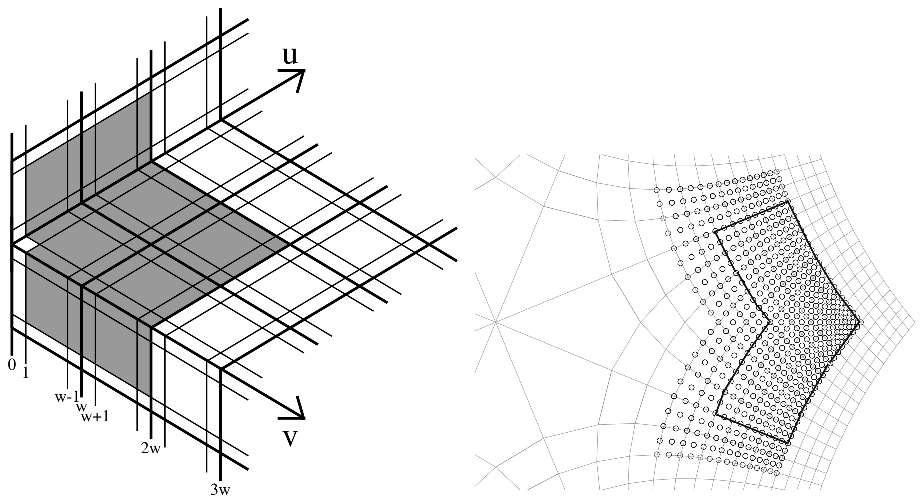


Fig. 7. Left: The schematic figure, like Figure 1, shows the region of the natural configuration for which first differences must be taken. Right: Circles are used to show the region of analysis after one subdivision step for a subdivision scheme of degree 9, which has a support width $w = 5$. The characteristic ring in the first sector lies within the thick dark lines.

8 Other Schemes

These results have been determined for primal binary subdivision tensor product B-splines of any degree. However, the only property of the B-splines that we have used is that they satisfy the convex hull property. The results therefore apply unchanged to all tensor product schemes where the univariate scheme is variation diminishing. The property that the first derivative is derivable as a subdivision scheme with differences as control points will be true for all schemes which have a $(1 + z)/2$ factor in the u direction of the symbol (z-transform) of their mask. Schemes which do not have this property are unlikely to be satisfactory in other respects. Similarly, the condition of symmetry under rotation and reflection can usually be taken for granted. Failure to meet this condition is directly evident from the mask.

It was pointed out by Goldman and DeRose [11] that where the univariate scheme is not variation diminishing it is instead possible to use the fact that the limit of a convergent subdivision scheme lies within an enclosure which is just that of the control polyhedron scaled up by some factor, namely the largest value of the sum of the magnitudes of the basis functions. It is therefore possible to take the bounds on the first differences in the same regions and scale these up before checking whether those bounds always lie within the first quadrant of the global coordinate system.

Where the scheme is not a tensor product, is dual rather than primal, is based on triangles rather than quads, or is not binary, the arguments can be followed in exactly the same way, but new figures would need to be drawn to replace those used here.

Where a scheme is not stationary, but has a stationary limit, the usual arguments (and conditions) can determine whether the continuity is that of the limit scheme [12].

9 Summary

The check that a subdivision scheme has a regular and injective characteristic map is required in order to prove that the scheme is C^1. Peters and Reif's Theorem 5.25 is a good basis for this test [3]. We have identified a standard procedure for applying this theorem numerically to the natural configuration of any bivariate binary scheme. The extension to ternary and higher n-ary schemes is expected to be straightforward.

A Pseudo Code of Procedure

1. Determine the eigenvector of the dominant eigencomponent in the Fourier block ± 1. Check that it is the subdominant component.
2. Evaluate the natural configuration over a large enough region.
3. Refine the region of analysis until all the first differences lie in the first quadrant of the global coordinate system or a predefined depth limit of, say, ten subdivision steps is exceeded.
4. If first differences are positive, Theorem 5.25 can be invoked and the scheme was proven to be C^1.
   Otherwise, no proof of C^1 is available. Evaluate derivative at a few places and disprove C^1 by example.

References

1. Dyn, N.: Analysis of convergence and smoothness by the formalism of laurent polynomials. In: Iske, A., Quack, E., Floater, M. (eds.) Tutorials on Multiresolution in Geometric Modelling, pp. 51–68. Springer, Heidelberg (2002)
2. Reif, U.: A unified approach to subdivision algorithms near extraordinary vertices. CAGD 12(2), 153–174 (1995)
3. Peters, J., Reif, U.: Subdivision Surfaces, p. 107. Springer, Heidelberg (2008)
4. Cashman, T.J., Augsdörfer, U.H., Dodgson, N.A., Sabin, M.A.: NURBS with Extraordinary Points: High-degree, Non-uniform, Rational Subdivision Schemes. ACM Transactions on Graphics 28(3) (2009)
5. Zorin, D.: A method for analysis of c1-continuity of subdivision surfaces. SIAM Journal of Numerical Analysis 37(5), 1677–1708 (2000)
6. Umlauf, G.: A technique for verifying the smoothness of subdivision schemes. In: Lucian, M., Neamtu, M. (eds.) Geometric Modeling and Computing: Seattle 2003, pp. 513–521. Nashboro Press (2004)
7. Ginkel, I., Umlauf, G.: Analyzing a generalized loop subdivision scheme. Computing 79, 353–363 (2007)

8. Ball, A., Storry, D.: Conditions for tangent plane continuity over recursively generated B-spline surfaces. ACM Transactions on Graphics 7(2), 83–108 (1988)
9. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 10(6), 183–188 (1978)
10. Augsdörfer, U.H., Dodgson, N.A., Sabin, M.A.: Tuning subdivision by minimising gaussian curvature variation near extraordinary vertices. Computer Graphics Forum 25(3), 263–272 (2006)
11. Goldman, R.N., DeRose, T.D.: Recursive subdivision without the convex hull property. CAGD 3, 247–265 (1987)
12. Wallner, J., Dyn, N.: Convergence and C^1 analysis of subdivision schemes on manifolds by proximity. CAGD 22(7), 593–622 (2005)

The Invariant Functions of the Rational Bi-cubic Bézier Surfaces

H.E. Bez

Department of Computer Science, Loughborough University, UK
h.e.bez@lboro.ac.uk

Abstract. Patterson's work [1] on the invariants of the rational Bézier paths may be extended to permit weight vectors of mixed-sign [2]. In this more general situation, in addition to Patterson's continuous invariants, a discrete sign-pattern invariant is required to distinguish path geometry. The author's derivation of the invariants differs from that of Patterson's and extends naturally to the rational Bézier surfaces. In this paper it is shown that 13 continuous invariant functions and a discrete, sign-pattern, invariant exist for the bi-cubic surfaces. Explicit functional forms of the invariant functions for the bi-cubics are obtained. The results are viewed from the perspective of the Fundamental Theorem on invariants for Lie groups.

1 Introduction and Scope

Patterson showed that if the weight vector $\omega = (\omega_0, \ldots, \omega_n)$ of a degree n Bézier rational path is transformed to $\omega^* = (\omega_0^*, \ldots, \omega_n^*)$, where $\omega_i > 0$ and $\omega_i^* > 0$, then, for any vertex set, the shape of the path is invariant if:

$$\varsigma_i(\omega^*) = \varsigma_i(\omega) \text{ for all } 1 \leq i \leq n-1$$

where $\varsigma_i(\omega) = \frac{\omega_{i-1}\omega_{i+1}}{\omega_i^2}$. The invariance of the $n-1$ continuous function, ς_i, comprise invariant-geometry conditions for Bézier paths with positive weights. The invariants also provide information about the underlying curve type and enable standardised weight vectors to be determined [3].

In this paper the author's generalisation of Patterson's results, to the case of weight-vectors of mixed-sign being permitted, are summarised. Following this the invariant functions of the bi-cubic Bézier surfaces are derived - again allowing mixed-sign weight vectors. Negative weights are allowed for a number of reasons including:

- completeness and generality,
- the difficulty of avoiding negative weights when parametrising some surfaces, e.g., Dupin cyclides, with a small number of low-degree patches.

It is shown that 13 independent, real-valued invariant functions, denoted ϖ_{ij}, and a sign-pattern invariant, denoted, *sp*, exist for these surfaces. The main result obtained states that if the weight vector ω of a bi-cubic surface, with arbitrary vertices, is transformed to ω^* then the shape of the patch is unchanged if and only if:

$$\varpi_{ij}(\omega^*) = \varpi_{ij}(\omega), \quad \text{and} \quad sp(\omega^*) = sp(\omega).$$

Explicit forms for the ϖ type invariants for the bi-cubics are derived and some further applications are discussed.

2 Mathematical Preliminaries

2.1 Functional Independence

If $f_i : \mathbb{R}^q \to \mathbb{R}$, for $1 \le i \le k$, then the functions $f \equiv (f_1, \ldots, f_k)$ are functionally *dependent* if:

$$F \circ f = 0$$

for some $F : \mathbb{R}^k \to \mathbb{R}$ not identically zero. The functions are *independent* if no such F exists. Equivalently, for dependence, we have $(dF \circ f)df = 0$ or, taking the transpose, the linear system:

$$\begin{bmatrix} \partial_1 f_1 & \partial_1 f_2 & \ldots & \partial_1 f_k \\ \partial_2 f_1 & \partial_2 f_2 & \ldots & \partial_2 f_k \\ \vdots & \vdots & \vdots & \vdots \\ \partial_q f_1 & \partial_q f_2 & \ldots & \partial_q f_k \end{bmatrix} \begin{bmatrix} \partial_1 F \circ f \\ \partial_2 F \circ f \\ \vdots \\ \partial_k F \circ f \end{bmatrix} = 0$$

where ∂_i denotes partial derivative with respect to the ith variable. It follows that if df is of maximal rank, then the only solution is $F \equiv 0$ and the functions $f_1, \ldots, f_k$ are independent.

In this paper we say that the functions $f_1, \ldots, f_k$ are *manifestly* independent if either:

1. each f_i is a function of a variable that does not occur in the other functions, or
2. all except one of $f_1, \ldots, f_k$, is a function of a variable that does not occur in the other functions, and the exceptional function does not depend on any of the variables unique to the others.

For example, the pair $f_i : \mathbb{R}^3 \to \mathbb{R}$, for $i = 1, 2$, defined by:

$$f_1(x, y, z) = x\ z, \qquad f_2(x, y, z) = y\ z$$

is manifestly independent as, respectively, they have variables x and y that do not occur in the other function. It follows that no, nontrivial, F with $F(f_1, f_2) = 0$ can exist. Further, the triple (f_1, f_2, f_3), where $f_3(x, y, z) = z$, is manifestly independent – as f_3 is not a function of either x or y it cannot be expressed as a function of f_1 and f_2. In cases of manifest independence it is not necessary to determine the rank of df.

2.2 General Observations on Invariants

No unique set of invariants exists for a given problem, since further invariants may always be determined as functions of a known set; i.e., if $\iota_1, \dots, \iota_k$ are real-valued invariants and h is a real-valued function of k variables then $h(\iota_1, \dots, \iota_k)$ is an invariant. The invariant problem is therefore one of determining a set of functionally independent invariants that is complete – in the sense that all invariants not in the set may be expressed as a function of those that are. The cardinality of two complete independent sets is the same [6].

2.3 Sign-Pattern

Let $A = \{+, -\}$ and A^* be the set of all vectors of symbols from A. For $\alpha \in A^*$ we define α' to be the complement of α – i.e., the vector obtained by replacing the $+$'s of α by $-$'s and the $-$'s by $+$'s. We say that two elements, $\alpha, \beta \in A^*$ are equivalent (written $\alpha \sim \beta$), or have the same sign-pattern, if $\alpha = \beta$ or $\alpha = \beta'$. For example $(+,+,-,+,-) \sim (-,-,+,-,+)$.

The sign-pattern of a vector $X = (x_1, \dots, x_m) \in \mathbb{R}^m$, with $x_i \neq 0$, may defined to be $sp(X) = [(sign(x_1), \dots, sign(x_m))] \in A^*/\sim$.

2.4 Equivalent Curve and Surface Parametrisations

We denote by $\mathcal{P}$, the set of, suitably defined, regular functions $\sigma : I \to \mathbb{R}^2$, where I is an interval of $\mathbb{R}$, that provide local parametrisations of curves embedded in $\mathbb{R}^2$. The general conditions under which such functions σ_1 and σ_2, on respectively I_1 and I_2, parametrise the same region of a curve are as follows: there must exist a sufficiently smooth invertible function $\phi : I_1 \to I_2$ such that $\sigma_1 = \sigma_2 \circ \phi$.

We denote by $\mathcal{S}$, the set of, suitably defined, regular functions $\sigma : I_1 \times I_2 \to \mathbb{R}^3$, where I_1 and I_2 are intervals of $\mathbb{R}$, that provide local parametrisations of surfaces embedded in $\mathbb{R}^3$. The general conditions under which functions parametrise the same surface are as follows: functions σ_1, σ_2 on, respectively, $I_1 \times I_2$ and $I_1^* \times I_2^*$ parametrise the same surface if there is a sufficiently smooth invertible function $\psi : I_1 \times I_2 \to I_1^* \times I_2^*$ such that $\sigma_1 = \sigma_2 \circ \psi$.

3 Generalised Invariance Results for Rational Bézier Paths

3.1 Summary of the Generalised Invariant-Geometry Conditions

Full details and proofs of the results summarised in this section may be found in [2]. If $b_i(t) = \binom{n}{i} t^i (1-t)^{n-i}$ for $0 \leq i \leq n$ are the Bernstein polynomials of degree n then the rational paths of degree n may be be written in this basis as:

$$\eta[v^*, \omega](t) = \frac{\sum_{i=0}^{n} b_i(t) v_i^*}{\sum_{i=0}^{n} b_i(t) \omega_i}$$

$$= \frac{[1, t, \dots, t^n] B_n v^*}{[1, t, \dots, t^n] B_n \omega}$$

where

- $\omega_i \in \mathbb{R}, 0 \leq i \leq n$ are the 'weights' of the path and $\omega = (\omega_0, \ldots, \omega_n)^T$
- $v^* = [v_0^*, \ldots, v_n^*]^T$, $v_i^* \in \mathbb{R}^m$, are the Bernstein vectors
- B_n is upper-triangular the change-of-basis matrix

$$B_n = \begin{bmatrix} 1 & , & 0 & , & 0 & , \ldots , 0 \\ -\binom{n}{1} & , & \binom{n}{1}\binom{n-1}{0} & , & 0 & , \ldots , 0 \\ \binom{n}{2} & , & (-1)\binom{n}{1}\binom{n-1}{1} & , & \binom{n}{2}\binom{n-2}{1} & , \ldots , 0 \\ \vdots & & & & & \\ (-1)^n & , & (-1)^{n-1}\binom{n}{1} & , & (-1)^{n-2}\binom{n}{2} & , \ldots , 1 \end{bmatrix}$$

defined by

$$[b_0(t), b_1(t), \ldots, b_n(t)] = [1, t, \ldots, t^n]B_n.$$

The rational Bézier paths of degree n may be defined for the subset of the degree n Bernstein paths for which all the weights ω_i are non-zero. In this case the vectors $v_i = \frac{v_i^*}{\omega_i}$ are well-defined and the path may be written in the Bézier form:

$$\eta_B[v, \omega](t) = \frac{[1, t, \ldots, t^n]B_n \Omega_n v}{[1, t, \ldots, t^n]B_n \omega} \quad \text{for } \; 0 \leq t \leq 1 \tag{1}$$

where $\Omega_n = diag(\omega_0, \omega_1, \ldots, \omega_n)$ and v denotes $[v_0, \ldots, v_n]^T$.

Bézier paths have special properties in the case of all positive weights and for this reason discussion is often restricted to this case [3]. In this paper the only limitation on the weight values is that they are non-zero.

From section 2.4, it follows that weight vectors ω, ω^* determine paths with the same geometry on $[0, 1]$ if and only if there exists a re-parametrisation function ϕ such that

$$\eta_B[v, \omega] = \eta_B[v, \omega^*] \circ \phi \qquad \text{for all } \; v \in \mathbb{R}^m. \tag{2}$$

The general solution of (2) – comprising all ϕ functions and weight vector pairs ω and ω^* – determines the invariant-geometry conditions for Bézier paths. Relation (2) may be written

$$\frac{[1, t, \ldots, t^n]B_n \Omega_n \; v}{[1, t, \ldots, t^n]B_n \omega} = \frac{[1, \phi, \ldots, \phi^n]B_n \Omega_n^* \; v}{[1, \phi, \ldots, \phi^n]B_n \omega^*}$$

and, as this must hold for all v, it follows that it is valid to 'cancel' the column of v's to obtain the invariant-geometry functional equation:

$$\frac{[1, t, \ldots, t^n]B_n \Omega_n}{[1, t, \ldots, t^n]B_n \omega} = \frac{[1, \phi, \ldots, \phi^n]B_n \Omega_n^*}{[1, \phi, \ldots, \phi^n]B_n \omega^*}. \tag{3}$$

Denoting the ratio $\frac{\omega_i}{\omega_i^*}$ by ρ_i, it can be shown that the general solutions of (3) that are continuous bijections of $[0, 1]$ are given by:

$$\phi(t) = \frac{\rho_n t}{\rho_{n-1} + (\rho_n - \rho_{n-1})t}$$

where ρ_n and ρ_{n-1} are non-zero and have the same sign. The substitution of these ϕ solutions transform (3) from a functional equation to the following (equivalent) relationships between ω and ω^*.

$$\omega^* = c\, B_n^{-1} {U_\phi}^{-1} B_n \omega \tag{4}$$

$$\Omega_n^* = c\, B_n^{-1} {U_\phi}^{-1} B_n \Omega_n \tag{5}$$

where c is an arbitrary (non-zero) multiplier,

$$U_\phi = \begin{bmatrix} \binom{n}{0}\rho_{n-1}^n & , & 0 & , & 0 & , \ldots , & 0 \\ \binom{n}{1}\rho_{n-1}^{n-1}\delta_n & , & \rho_n\binom{n-1}{0}\rho_{n-1}^{n-1} & , & 0 & , \ldots , & 0 \\ \binom{n}{2}\rho_{n-1}^{n-2}\delta_n^2 & , & \rho_n\binom{n-1}{1}\rho_{n-1}^{n-2}\delta_n & , & \rho_n^2\binom{n-2}{0}\rho_{n-1}^{n-2} & , \ldots , & 0 \\ \vdots & , & \vdots & , & \vdots & , & \vdots \\ \binom{n}{n-1}\rho_{n-1}\delta_n^{n-1} & , & \rho_n\binom{n-1}{n-2}\rho_{n-1}\delta_n^{n-2} & , & \rho_n^2\binom{n-2}{n-3}\rho_{n-1}\delta_n^{n-3} & , \ldots , & 0 \\ \delta_n^n & , & \rho_n\delta_n^{n-1} & , & \rho_n^2\delta_n^{n-2} & , \ldots , & \rho_n^n \end{bmatrix}$$

and $\delta_n = \rho_n - \rho_{n-1}$.

Relation (4) is a necessary and sufficient condition for invariant geometry - as is (5). Relation (4) is the generalization of a known transformation rule to the case of weights of mixed-sign being allowed. Relation (5) leads directly to the determination of the invariant functions.

Specifically, it may be shown that $B_n^{-1} {U_\phi}^{-1} B_n$ is the diagonal matrix

$$B_n^{-1} {U_\phi}^{-1} B_n = diag\left(\frac{1}{\rho_{n-1}^n},\ \frac{1}{\rho_{n-1}^{n-1}\rho_n},\ \frac{1}{\rho_{n-1}^{n-2}\rho_n^2}, \ldots, \frac{1}{\rho_{n-1}\rho_n^{n-1}},\ \frac{1}{\rho_n^n}\right);$$

it follows that:

1. relation (4) is equivalent to the following relationship between ω and ω^*:

$$\omega^* = \pm e^\alpha diag(1, e^\mu, \ldots, e^{(n-1)\mu}, e^{n\mu})\ \omega$$

   where $e^\alpha = \frac{|c|}{\rho_{n-1}^n}$ and $e^\mu = \frac{\rho_{n-1}}{\rho_n}$,

2. relation (5) is equivalent to the following constraints on ω and ω^*:

$$\varpi_i(\omega^*) = \varpi_i(\omega) \quad \text{for all} \quad 1 \le i \le n-1, \quad \text{and} \quad sp(\omega^*) = sp(\omega),$$

   where

$$\varpi_i(\omega) = \frac{\omega_i}{\omega_0}\left(\frac{\omega_{n-1}}{\omega_n}\right)^i,$$

3. relation (5) is equivalent to the following constraints on ω and ω^*:

$$\varsigma_i(\omega^*) = \varsigma_i(\omega) \quad \text{for all} \quad 1 \le i \le n-1, \quad \text{and} \quad sp(\omega^*) = sp(\omega),$$

   where

$$\varsigma_i(\omega) = \frac{\omega_{i-1}\omega_{i+1}}{\omega_i^2}.$$

The ς_i functions are Patterson's continuous invariants.

In addition:

1. The functions ϖ_i for $1 \leq i \leq n-1$ determine a manifestly functionally independent set.

2. The invariance of *sp* is not implied by the invariance of the ϖ_i functions (or the ς_i functions).

3. The ς_i functions do not, except in the cases $n = 2, 3$, determine a manifestly independent set. However a simple inductive proof of independence may be constructed.

The following proposition summarises the generalised invariant-geometry conditions for the rational Bézier paths.

Proposition 1. *(i) If the weight vector ω of a rational Bézier path is transformed to ω^* then the shape of the path is unchanged if and only if:*

$$\varpi_i(\omega^*) = \varpi_i(\omega) \text{ for all } 1 \leq i \leq n-1, \quad \text{and } sp(\omega^*) = sp(\omega)$$

or, there exist $\alpha, \mu \in \mathbb{R}$ such that:

$$\omega^* = \pm e^{\alpha} diag(1, e^{\mu}, \ldots, e^{(n-1)\mu}, e^{n\mu})\, \omega. \tag{6}$$

(ii) All complete, independent sets of invariants for the degree n rational Bézier paths have n elements.

3.2 Generalised Weight-Normalisations for the Bézier Paths

The 2-parameter transformation group defined by (6), allows arbitrary rational Bézier paths of degree n to be re-parametrised to rational forms having two weights transformed to unit modulus. Contrary to the situation for bi-cubic patches, the transformation group for paths allows any pair of weights to be chosen for normalisation.

Example 1. *Normalisation matrices for the cubic Bézier paths. Here, the pair (ω_0, ω_1) is chosen for transformation to unit modulus in the weight vector $\omega = (\omega_0, \omega_1, \omega_2, \omega_3)$. Normalisations of the form $\pm(1, 1, \omega_2^*, \omega_3^*)$ and $\pm(1, -1, \omega_2^*, \omega_3^*)$ occur—according to the signs of ω_0 and ω_1.*

(i) If ω_0 and ω_1 have the same sign then the conditions $e^{\alpha}\omega_0 = 1$ and $e^{\alpha}e^{\mu}\omega_1 = 1$, when $\omega_0 > 0$ and $\omega_1 > 0$, and $e^{\alpha}\omega_0 = -1$ and $e^{\alpha}e^{\mu}\omega_1 = -1$, when $\omega_0 < 0$ and $\omega_1 < 0$ give, from (6), the relation:

$$\begin{bmatrix} \omega_0^* \\ \omega_1^* \\ \omega_2^* \\ \omega_3^* \end{bmatrix} = \pm diag\left(\frac{1}{\omega_0}, \frac{1}{\omega_1}, \frac{\omega_0}{\omega_1^2}, \frac{\omega_0^2}{\omega_1^3}\right) \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \pm \begin{bmatrix} 1 \\ 1 \\ \frac{\omega_0\omega_2}{\omega_1^2} \\ \frac{\omega_0^2\omega_3}{\omega_1^3} \end{bmatrix}.$$

(ii) If ω_0 and ω_1 are of opposite sign then, assuming $\omega_1 < 0$, (6) becomes:

$$\begin{bmatrix} \omega_0^* \\ \omega_1^* \\ \omega_2^* \\ \omega_3^* \end{bmatrix} = \pm diag\left(\frac{1}{\omega_0}, \frac{1}{|\omega_1|}, \frac{\omega_0}{\omega_1^2}, \frac{\omega_0^2}{|\omega_1^3|}\right) \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \pm \begin{bmatrix} 1 \\ -1 \\ \frac{\omega_0\omega_2}{\omega_1^2} \\ \frac{\omega_0^2\omega_3}{|\omega_1^3|} \end{bmatrix}.$$

A similar transformation may be determined for the case $\omega_0 < 0$.

It is easily verified that the conditions:

$$\varsigma_i(\omega^*) = \varsigma_i(\omega), \; for\, i = 1, 2, \; and \; sp(\omega^*) = sp(\omega)$$

hold in all cases.

4 An Invariant-Geometry Functional Equation for the Bi-cubic Bézier Surfaces

If $b_i(t) = \binom{3}{i} t^i (1-t)^{3-i}$ then the rational, degree $(3,3)$ Bernstein surfaces take the form

$$\sigma[v,\omega](t,s) = \frac{\sum_{i=0}^3 \sum_{j=0}^3 b_i(t)b_j(s)v_{ij}^*}{\sum_i^3 \sum_j^3 b_i(t)b_j(s)\omega_{ij}}.$$

on a set $v^* = \{v_{ij}^* : 0 \leq i \leq 3, 0 \leq j \leq 3\}$ of 16 vectors and a vector $\omega = (\omega_{00}, \omega_{01}, \ldots, \omega_{03};\; \omega_{10}, \omega_{11}, \ldots, \omega_{13}; \ldots;\; \omega_{30}, \ldots \omega_{33})$ of 16 'weights'.

Rational Bézier surfaces of degree $(3,3)$ are defined for the subset of degree $(3,3)$ Bernstein surfaces for which all the weights are non-zero. In this case the vectors $v_{ij} = \frac{v_{ij}^*}{\omega_{ij}}$ are all well-defined and the surface may be written in the Bézier form:

$$\sigma_B[v,\omega](t,s) = \frac{\sum_{i=0}^3 \sum_{j=0}^3 b_i(t)b_j(s)\omega_{ij}v_{ij}}{\sum_{i=0}^3 \sum_{j=0}^3 b_i(t)b_j(s)\omega_{ij}}$$

We have

$$\sigma_B : V \times \mathcal{O} \to \mathcal{S}$$

where V is the set of all 4×4-tuples of vectors in $\mathbb{R}^3$ and $\mathcal{O}$ is the set of all 4×4-tuples of non-zero real numbers.

It follows from section 2.4 that weight vectors $\omega, \omega^* \in \mathcal{O}$ determine surfaces with the same geometry, for all $v \in V$, if and only if there exists a re-parametrisation function ψ such that

$$\sigma_B[v,\omega] = \sigma_B[v,\omega^*] \circ \psi \text{ for all } v \in V.$$

The general solution of this functional equation, comprising all ψ functions and ω^* vectors as functions of ω, determines the relationships between the weight vectors ω and ω^* for constant geometry on any vertex set v. The solutions of this functional equation for σ_B may be referred to as the vertex-independent symmetries of σ_B - it is these symmetries that are investigated in this paper.

Note that re-parametrisation functions ψ may always be expressed in the paired form

$$\psi(t,s) = (\phi(t,s), \mu(t,s)).$$

Following a little algebra the degree $(3,3)$ Bézier surface, $\sigma_B[v,\omega]$, may be re-written in the monomial tensor product basis form as:

$$\sigma_B[v,\omega](t,s) = \frac{\mathrm{t}\otimes\mathrm{s}(B_3\otimes B_3)\Omega_{33}\ v}{\mathrm{t}\otimes\mathrm{s}(B_3\otimes B_3)\omega} \tag{7}$$

where:

- $\mathrm{t}\otimes\mathrm{s} = [1,t,t^2,t^3]\otimes[1,s,s^2,s^3]$ is the tensor-product basis of the space of polynomials of degree $(3,3)$ in 2-variables,
- $B_3\otimes B_3$ is the 16×16 lower-triangular, tensor-product matrix

$$B_3\otimes B_3 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
3 & -6 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-1 & 3 & -3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-3 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
9 & -9 & 0 & 0 & -9 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-9 & 18 & -9 & 0 & 9 & -18 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
3 & -9 & 9 & -3 & -3 & 9 & -9 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
3 & 0 & 0 & 0 & -6 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-9 & 9 & 0 & 0 & 18 & -18 & 0 & 0 & -9 & 9 & 0 & 0 & 0 & 0 & 0 & 0\\
9 & -18 & 9 & 0 & -18 & 36 & -18 & 0 & 9 & -18 & 9 & 0 & 0 & 0 & 0 & 0\\
-3 & 9 & -9 & 3 & 6 & -18 & 18 & -6 & -3 & 9 & -9 & 3 & 0 & 0 & 0 & 0\\
-1 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
3 & -3 & 0 & 0 & -9 & 9 & 0 & 0 & 9 & -9 & 0 & 0 & -3 & 3 & 0 & 0\\
-3 & 6 & -3 & 0 & 9 & -18 & 9 & 0 & -9 & 18 & -9 & 0 & 3 & -6 & 3 & 0\\
1 & -3 & 3 & -1 & -3 & 9 & -9 & 3 & 3 & -9 & 9 & -3 & -1 & 3 & -3 & 1
\end{bmatrix},$$

- Ω_{33} is the 16×16 invertible diagonal matrix:

$$\Omega_{33} = diag(\omega_{00},\omega_{01},\ldots,\omega_{03};\ \omega_{10},\omega_{11},\ldots,\omega_{13};\ldots;\ \omega_{30},\ldots\omega_{33}),$$

- $v = (v_{00},\ldots,v_{03};v_{10},\ldots,v_{13};\ldots,:v_{30},\ldots,v_{33})^T$,
- $\omega = (\omega_{00},\ldots,\omega_{03};\omega_{10},\ldots,\omega_{13};\ldots,:\omega_{30},\ldots,\omega_{33})^T$.

The monomial expression (equation (7)) for $\sigma_B[v,\omega]$ is the generalization of the monomial representation of Bézier curves (equation (1)) and is essential to the approach now taken to determine the invariant-geometry conditions for the bi-cubic surfaces.

If $\psi(t,s) = (\phi(t,s),\mu(t,s))$ is a re-parametrisation function, then $\sigma_B[v,\omega]\circ\psi$ is given by

$$\sigma_B[v,\omega](\phi,\mu) = \frac{\phi\otimes\mu(B_3\otimes B_3)\Omega_{33}\ v}{\phi\otimes\mu(B_3\otimes B_3)\omega}$$

and the invariant-geometry functional equation for Bézier surfaces, of degree $(3,3)$ in (t,s), may be written as:

$$\frac{\phi\otimes\mu(B_3\otimes B_3)\Omega^*_{33}\ v}{\phi\otimes\mu(B_3\otimes B_3)\omega^*} = \frac{\mathrm{t}\otimes\mathrm{s}(B_3\otimes B_3)\Omega_{33}\ v}{\mathrm{t}\otimes\mathrm{s}(B_3\otimes B_3)\omega} \quad \text{for all } \ v\in V \tag{8}$$

where Ω^*_{33} is the 16×16 diagonal matrix defined by:

$$\Omega^*_{33} = diag(\omega^*_{00}, \omega^*_{01}, \ldots, \omega^*_{03};\ \omega^*_{10}, \omega^*_{11}, \ldots, \omega^*_{13};\ \ldots; \omega^*_{30}, \ldots, \omega^*_{33})$$

and ω^* are the 16×1 column vector

$$\omega^* = (\omega^*_{00}, \omega^*_{01}, \ldots, \omega^*_{03};\ \omega^*_{10}, \omega^*_{11}, \ldots, \omega^*_{13};\ \ldots; \omega^*_{30}, \ldots, \omega^*_{33})^T.$$

As (8) required to hold for all v, it follows that the functional equation for invariant-geometry reduces to:

$$\frac{\phi \otimes \mu (B_3 \otimes B_3)\Omega^*_{33}}{\phi \otimes \mu (B_3 \otimes B_3)\omega^*} = \frac{\mathrm{t} \otimes \mathrm{s}(B_3 \otimes B_3)\Omega_{33}}{\mathrm{t} \otimes \mathrm{s}(B_3 \otimes B_3)\omega}. \tag{9}$$

5 The Functional Solutions for the Bi-cubics

The ratios $\frac{\omega_{ij}}{\omega^*_{ij}}$ occur naturally in the solution of equation (9), and are denoted henceforth by ρ_{ij}.

All the $\psi \equiv (\phi, \mu)$ solutions of equation (9) are required; while it is known that pairs of Möbius functions $(\phi(t), \mu(s)) = (\frac{at}{1+(a-1)t}, \frac{bs}{1+(b-1)s})$ are solutions, the following Lemma establishes that there are no others.

Lemma 1. *If $\psi = (\phi, \mu)$ is invertible and satisfies equation (9), for the weight vectors (ω, ω^*), then:*

$$\phi(t,s) = \frac{a\,t}{1+(a-1)t} \quad and \quad \mu(t,s) = \frac{b\,s}{1+(b-1)s}$$

where $a = \frac{\rho_{3,3}}{\rho_{2,3}} > 0$ and $b = \frac{\rho_{3,3}}{\rho_{3,2}} > 0$.

Proof. The 16th, the 15th and the 12th columns of the matrix $B_3 \otimes B_3$ are, respectively:

$$[\{0\}^{15}, 1]^T, \quad [\{0\}^{14}, 3, -3]^T \quad \text{and} \quad [\{0\}^{11}, 3, \{0\}^3, -3]^T$$

where $\{0\}^3 = 0, 0, 0$ etc. The quantities $\mathrm{t} \otimes \mathrm{s}(B_3 \otimes B_3)\omega$ and $\phi \otimes \mu(B_3 \otimes B_3)\omega^*$ are scalar; denoting them, respectively, by P and Q we obtain from equation (9):

$$P\ \phi \otimes \mu(B_3 \otimes B_3)\Omega^*_{33} = Q\ \mathrm{t} \otimes \mathrm{s}(B_3 \otimes B_3)\Omega_{33}. \tag{10}$$

Comparing elements $(1, 16)$, $(1, 15)$ and $(1, 12)$ on each side of the functional equation array (10) we obtain:

- $P\ \phi^3\mu^3\omega^*_{33} = Q\ t^3s^3\omega_{33}$
- $P\ [\phi^3\mu^2 - \phi^3\mu^3]\omega^*_{3,2} = Q\ [t^3s^2 - t^3s^3]\omega_{3,2}$
- $P\ [\phi^2\mu^3 - \phi^3\mu^3]\omega^*_{2,3} = Q\ [t^2s^3 - t^3s^3]\omega_{2,3}.$

The ratio of the first two of these relations gives:

$$\mu(t,s) = \frac{\rho_{3,3}\ s}{\rho_{3,2} + (\rho_{3,3} - \rho_{3,2})s},$$

similarly the first and third give

$$\phi(t,s) = \frac{\rho_{3,3}\ t}{\rho_{2,3} + (\rho_{3,3} - \rho_{2,3})t}$$

i.e., the solutions for μ and ϕ are both Möbius functions of a single variable - as required. Further, we have

1. $\phi(0) = 0, \phi(1) = 1, \mu(0) = 0, \mu(1) = 1$
2. $\phi'(t) = \frac{\rho_{3,3}\rho_{2,3}}{(\rho_{2,3}+(\rho_{3,3}-\rho_{2,3})t)^2}$ and $\mu'(t) = \frac{\rho_{3,3}\rho_{3,2}}{(\rho_{3,2}+(\rho_{3,3}-\rho_{3,2})s)^2}$

and the invertibility of ψ is equivalent to $\phi'(t) \neq 0$ for all $t \in [0,1]$ and $\mu'(s) \neq 0$ for all $s \in [0,1]$. It follows that:

– $\rho_{3,3}\rho_{2,3} > 0$, equivalently $\rho_{3,3}$ and $\rho_{2,3}$ have the same sign

– $\rho_{3,3}\rho_{3,2} > 0$, equivalently $\rho_{3,3}$ and $\rho_{3,2}$ have the same sign

i.e., $\rho_{3,3}, \rho_{2,3}$ and $\rho_{3,2}$ all have the same sign. Equivalently $a > 0$ and $b > 0$ where a and b are as defined in the hypothesis.

5.1 Reduced Invariant-Geometry Conditions for the Bi-cubics

Lemma 2. *The invariant-geometry condition (9) for the degree $(3,3)$ Bézier surfaces is equivalent to:*

$$\omega^* = c\ \ (B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)\ \omega \tag{11}$$

or

$$\Omega^*_{33} = c\ \ (B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)\Omega_{33} \tag{12}$$

where c is an arbitrary non-zero multiplier, U_ϕ is the 4×4 matrix:

$$U_\phi = \begin{bmatrix} \rho_{23}^3 & , & 0 & , & 0 & , & 0 \\ 3\rho_{23}^2(\rho_{33}-\rho_{23}) & , & \rho_{33}\rho_{23}^2 & , & 0 & , & 0 \\ 3\rho_{23}(\rho_{33}-\rho_{23})^2 & , & 2\rho_{33}\rho_{23}(\rho_{33}-\rho_{23}) & , & \rho_{33}^2\rho_{23} & , & 0 \\ (\rho_{33}-\rho_{23})^3 & , & \rho_{33}(\rho_{33}-\rho_{23})^2 & , & \rho_{33}^2(\rho_{33}-\rho_{23}) & , & \rho_{33}^3 \end{bmatrix}$$

and

$$U_\mu = \begin{bmatrix} \rho_{32}^3 & , & 0 & , & 0 & , & 0 \\ 3\rho_{32}^2(\rho_{33}-\rho_{32}) & , & \rho_{33}\rho_{32}^2 & , & 0 & , & 0 \\ 3\rho_{32}(\rho_{33}-\rho_{32})^2 & , & 2\rho_{33}\rho_{32}(\rho_{33}-\rho_{32}) & , & \rho_{33}^2\rho_{32} & , & 0 \\ (\rho_{33}-\rho_{32})^3 & , & \rho_{33}(\rho_{33}-\rho_{32})^2 & , & \rho_{33}^2(\rho_{33}-\rho_{32}) & , & \rho_{33}^3 \end{bmatrix}.$$

Proof. From Lemma 1 we may write $\phi = \frac{\phi_N}{\phi_D}$ where $\phi_N = at$ and $\phi_D = 1 + (a - 1)t$, similarly $\mu = \frac{\mu_N}{\mu_D}$ with $\mu_N = bs$ and $\mu_D = 1 + (b-1)s$.

It follows that:

$$\begin{aligned}[1, \phi, \phi^2, \phi^3] &= \frac{1}{\phi_D^3}[\phi_D^3, \phi_D^2\phi_N, \phi_D\phi_N^2, \phi_N^3]. \\ &= \frac{1}{\phi_D^3}[(1+(a-1)t)^3, (1+(a-1)t)^2 at, (1+(a-1)t)(at)^2, (at)^3] \\ &= \frac{1}{\phi_D^3}[1, t, t^2, t^3]U_\phi\end{aligned}$$

and

$$[1, \mu, \mu^2, \mu^3] = \frac{1}{\phi_D^3}[1, s, s^2, s^3]U_\mu$$

where U_ϕ and U_μ take the form given in the statement of the Lemma. Hence

$$\begin{aligned}\sigma_B[v, \omega^*](\phi, \mu) &= \frac{\phi \otimes \mu(B_3 \otimes B_3)\Omega_{33}^* v}{\phi \otimes \mu(B_3 \otimes B_3)\omega^*} \\ &= \frac{\mathrm{t} \otimes \mathrm{s}(U_\phi \otimes U_\mu)(B_3 \otimes B_3)\Omega_{33}^* v}{\mathrm{t} \otimes \mathrm{s}(U_\phi \otimes U_\mu)(B_3 \otimes B_3)\omega^*}\end{aligned}$$

and the invariant-geometry condition (9) transforms to:

$$\frac{\mathrm{t} \otimes \mathrm{s}(U_\phi \otimes U_\mu)(B_3 \otimes B_3)\Omega_{33}^*}{\mathrm{t} \otimes \mathrm{s}(U_\phi \otimes U_\mu)(B_3 \otimes B_3)\omega^*} = \frac{\mathrm{t} \otimes \mathrm{s}(B_3 \otimes B_3)\Omega_{33}}{\mathrm{t} \otimes \mathrm{s}(B_3 \otimes B_3)\omega}.$$

As this should hold for all t, s, we have

$$\frac{(U_\phi \otimes U_\mu)(B_3 \otimes B_3)\Omega_{33}^*}{(U_\phi \otimes U_\mu)(B_3 \otimes B_3)\omega^*} = \frac{(B_3 \otimes B_3)\Omega_{33}}{(B_3 \otimes B_3)\omega}.$$

Hence $c\ (B_3 \otimes B_3)\ \omega = (U_\phi \otimes U_\mu)(B_3 \otimes B_3)\ \omega^*$ and $c\ (B_3 \otimes B_3)\Omega_{33} = (U_\phi \otimes U_\mu)(B_3 \otimes B_3)\Omega_{33}^*$, where c is an arbitrary non-zero multiplier; equivalently

$$\omega^* = c\ \ (B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)\ \omega$$

and

$$\Omega_{33}^* = c\ \ (B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)\Omega_{33}.$$

The equivalence of the two above relations follows from the diagonal nature of the matrices $B_3^{-1}U_\phi^{-1}B_3$ and $B_3^{-1}U_\mu^{-1}B_3$. It is easy to show that

$$B_3^{-1}U_\phi^{-1}B_3 = diag(1, \frac{1}{a}, \frac{1}{a^2}, \frac{1}{a^3}) \text{ and } B_3^{-1}U_\mu^{-1}B_3 = diag(1, \frac{1}{b}, \frac{1}{b^2}, \frac{1}{b^3})$$

hence $(B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)$ is diagonal. As A is diagonal, for any column x we have $x^* = Ax$ if and only if $diag(x^*) = A\ diag(x)$; hence (11) if and only if (12).

5.2 Invariant-Geometry Transformations and the Invariant Functions for the Bi-cubics

Lemma 3. *The invariant-geometry condition (9) for the degree* $(3,3)$ *Bézier surfaces is equivalent to:*

$$\omega^* = \pm e^\alpha diag(1, e^\gamma, \ldots, e^{3\gamma}; e^\beta, e^{\beta+\gamma}, \ldots, e^{\beta+3\gamma}; \ldots; e^{3\beta}, e^{3\beta+\gamma}, \ldots, e^{3\beta+3\gamma})\,\omega$$

where $\alpha, \beta, \gamma \in \mathbb{R}$.

Proof. It follows from the proof of 2 that $(B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)$ is the 16×16 diagonal matrix:

$$(B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3) = diag(\; 1, \frac{1}{b}, \frac{1}{b^2}, \frac{1}{b^3}; \frac{1}{a}, \frac{1}{ab}, \frac{1}{ab^2}, \frac{1}{ab^3}; \frac{1}{a^2}, \frac{1}{a^2b}, \frac{1}{a^2b^2}, \frac{1}{a^2b^3}; \frac{1}{a^3}, \frac{1}{a^3b}, \frac{1}{a^3b^2}, \frac{1}{a^3b^3}).$$

From Lemma 1 we have $a > 0$ and $b > 0$; hence all elements of the matrix $(B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)$ are positive and we may write, from equation (11):

$$\omega^* = \pm e^\alpha diag(1, e^\gamma, \ldots, e^{3\gamma}; e^\beta, e^{\beta+\gamma}, \ldots, e^{\beta+3\gamma}; \ldots; e^{3\beta}, e^{3\beta+\gamma}, \ldots, e^{3\beta+3\gamma})\,\omega$$

where α, being defined by $e^\alpha = |c|$, is arbitrary, $e^\beta = \frac{1}{a}$ and $e^\gamma = \frac{1}{b}$.

The transformations of Lemma 3 determine a 3-parameter subgroup of the general linear group $GL(16, \mathbb{R})$. They may be written as $\omega^*_{ij} = \pm e^\alpha (e^\gamma)^i (e^\beta)^j\, \omega_{ij}$ and amount to a generalization of the weight transformations implied by the expression for re-parametrised Bézier surfaces (see [3] page 194), to the case of mixed-sign weight vectors. The arbitrary multiplier $\pm e^\alpha$ corresponds to universal scaling of the weights.

Denote the set of pairs (i, j) of natural numbers satisfying the relations:

$$i = 0 \;\text{ and }\; 1 \le j \le 3; \;\; 1 \le i \le 2 \;\text{ and }\; 0 \le j \le 3; \;\; i = 3 \;\text{ and }\; 0 \le j \le 1$$

by $\mathcal{I}$. The set $\mathcal{I}$ contains 13 pairs.

Lemma 4. *The invariant-geometry condition (9) for the degree* $(3,3)$ *Bézier surfaces is equivalent to the following conditions on* ω *and* ω^**:*

$$sp(\omega^*) = sp(\omega)$$

and

$$\varpi_{ij}(\omega^*) = \varpi_{ij}(\omega) \;\; \textit{for all} \;\; (i,j) \in \mathcal{I}$$

where the 13 *functions,* ϖ_{ij}*, are defined by*

$$\varpi_{ij}(\omega) = \frac{\omega_{ij}}{\omega_{00}} \left\{ \left[\frac{\omega_{2,3}}{\omega_{33}}\right]^i \left[\frac{\omega_{3,2}}{\omega_{33}}\right]^j \right\}.$$

Proof. We have $\Omega_{33}^* = c \ (B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)\Omega_{33}$ where the diagonal matrix $(B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)$ has positive elements (see proof of Lemma 3). Given that c can take either sign, it follows that the signs of all the diagonal elements of Ω_{33}^* are the same as the signs of those of Ω_{33} or are all of opposite sign. Hence $sp(\omega^*) = sp(\omega)$. Rearranging (12) we obtain

$$\frac{1}{c}\, I_{16} = (B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)\ \Omega_{33}{\Omega_{33}^*}^{-1}$$

where I_{16} is the 16×16 identity matrix. Substituting

$$\Omega_{33}{\Omega_{33}^*}^{-1} = diag(\rho_{00}, \rho_{01}, \ldots, \rho_{33})$$

and the explicit diagonal form for $(B_3^{-1}U_\phi^{-1}B_3) \otimes (B_3^{-1}U_\mu^{-1}B_3)$ (see proof of Lemma 3) gives

$$\frac{1}{c}\, I_{16} = diag(\ 1, \frac{1}{b}, \frac{1}{b^2}, \frac{1}{b^3}; \frac{1}{a}, \frac{1}{ab}, \frac{1}{ab^2}, \frac{1}{ab^3}; \frac{1}{a^2}, \frac{1}{a^2b}, \frac{1}{a^2b^2}, \frac{1}{a^2b^3}; \frac{1}{a^3}, \frac{1}{a^3b}, \frac{1}{a^3b^2}, \frac{1}{a^3b^3})\ diag(\rho_{00}, \rho_{01}, \ldots, \rho_{33})$$

i.e.,

$$\tfrac{1}{c}\, I_{16} = diag(\ \rho_{00}\ , \tfrac{\rho_{01}}{b}\ , \tfrac{\rho_{02}}{b^2}\ , \tfrac{\rho_{03}}{b^3}; \tfrac{\rho_{10}}{a}\ , \tfrac{\rho_{11}}{ab}\ , \tfrac{\rho_{12}}{ab^2}\ , \tfrac{\rho_{13}}{ab^3}; \tfrac{\rho_{20}}{a^2}\ , \tfrac{\rho_{21}}{a^2b}\ , \tfrac{\rho_{22}}{a^2b^2}\ , \tfrac{\rho_{23}}{a^2b^3}; \tfrac{\rho_{30}}{a^3}\ , \tfrac{\rho_{31}}{a^3b}\ , \tfrac{\rho_{32}}{a^3b^2}\ , \tfrac{\rho_{33}}{a^3b^3}).$$

The above relationship implies immediately that all the elements of the diagonal matrix on the right-hand-side are equal. We therefore have:

$$\rho_{0,0} = \frac{\rho_{0,j}}{b^j} \text{ for } 1 \le j \le 3, \quad \rho_{0,0} = \frac{\rho_{1,j}}{ab^j} \text{ for } 0 \le j \le 3$$
$$\rho_{0,0} = \frac{\rho_{2,j}}{a^2b^j} \text{ for } 0 \le j \le 3, \quad \rho_{0,0} = \frac{\rho_{3,j}}{a^3b^j} \text{ for } 0 \le j \le 3.$$

There is some redundancy in these equations due to the relations:

$$\frac{\rho_{3,2}}{a^3b^2} = \frac{\rho_{3,3}}{a^3b^3} = \frac{\rho_{2,3}}{a^2b^3}.$$

It follows that the terms corresponding to $j = 2$ and $j = 3$ in the final set of relations above can be omitted - as they are both identical to the 3rd term, $\frac{\rho_{2,3}}{a^2b^3}$, of the previous set. We therefore have:

$$\rho_{0,0} = \frac{\rho_{0,j}}{b^j} \text{ for } 1 \le j \le 3, \quad \rho_{0,0} = \frac{\rho_{1,j}}{ab^j} \text{ for } 0 \le j \le 3$$
$$\rho_{0,0} = \frac{\rho_{2,j}}{a^2b^j} \text{ for } 0 \le j \le 3, \quad \rho_{0,0} = \frac{\rho_{3,j}}{a^3b^j} \text{ for } 0 \le j \le 1.$$

Given that $a = \frac{\rho_{3,3}}{\rho_{2,3}}$ and $b = \frac{\rho_{3,3}}{\rho_{3,2}}$, it follows that:

$$\frac{\rho_{0,j}}{\rho_{0,0}}\left[\frac{\rho_{3,2}}{\rho_{3,3}}\right]^j = 1 \text{ for } 1 \le j \le 3, \quad \frac{\rho_{1,j}}{\rho_{0,0}}\left(\frac{\rho_{2,3}}{\rho_{3,3}}\right)\left[\frac{\rho_{3,2}}{\rho_{3,3}}\right]^j = 1 \text{ for } 0 \le j \le 3$$
$$\frac{\rho_{2,j}}{\rho_{0,0}}\left(\frac{\rho_{2,3}}{\rho_{3,3}}\right)^2\left[\frac{\rho_{3,2}}{\rho_{3,3}}\right]^j = 1 \text{ for } 0 \le j \le 3, \quad \frac{\rho_{3,j}}{\rho_{0,0}}\left(\frac{\rho_{2,3}}{\rho_{3,3}}\right)^3\left[\frac{\rho_{3,2}}{\rho_{3,3}}\right]^j = 1 \text{ for } 0 \le j \le 1$$

and that the 13 functions defined by:

$$\varpi_{0,j}(\omega) = \frac{\omega_{0,j}}{\omega_{0,0}}\left[\frac{\omega_{3,2}}{\omega_{3,3}}\right]^j \quad \text{for } 1 \leq j \leq 3, \quad \varpi_{1,j}(\omega) = \frac{\omega_{1,j}}{\omega_{0,0}}\left(\frac{\omega_{2,3}}{\omega_{3,3}}\right)\left[\frac{\omega_{3,2}}{\omega_{3,3}}\right]^j \quad \text{for } 0 \leq j \leq 3$$

$$\varpi_{2,j}(\omega) = \frac{\omega_{2,j}}{\omega_{0,0}}\left(\frac{\omega_{2,3}}{\omega_{3,3}}\right)^2\left[\frac{\omega_{3,2}}{\omega_{3,3}}\right]^j \quad \text{for } 0 \leq j \leq 3, \quad \varpi_{3,j}(\omega) = \frac{\omega_{3,j}}{\omega_{0,0}}\left(\frac{\omega_{2,3}}{\omega_{3,3}}\right)^3\left[\frac{\omega_{3,2}}{\omega_{3,3}}\right]^j \quad \text{for } 0 \leq j \leq 1$$

are invariant. The invariants may be written as:

$$\varpi_{ij}(\omega) = \frac{\omega_{ij}}{\omega_{00}}\left\{\left[\frac{\omega_{2,3}}{\omega_{33}}\right]^i\left[\frac{\omega_{3,2}}{\omega_{33}}\right]^j\right\}$$

for $i = 0$ and $1 \leq j \leq 3$; $1 \leq i \leq 2$ and $0 \leq j \leq 3$; $i = 3$ and $0 \leq j \leq 1$.

The first two Corollaries, below, are immediate from Lemmas 3 and 4 respectively and provide tests for equivalence of shape parametrised on $[0,1] \times [0,1]$.

Corollary 1. *Two degree $(3,3)$ Bézier functions with weight vectors ω and ω^* and identical vertex sets v, parametrise the same region of a surface on the domain $[0,1] \times [0,1]$, for all $v \in V$, if and only if:*

$$sp(\omega^*) = sp(\omega) \quad \text{and} \quad \varpi_{ij}(\omega^*) = \varpi_{ij}(\omega) \quad \text{for all } (i,j) \in \mathcal{I}.$$

Corollary 2. *Two degree $(3,3)$ Bézier functions with weight vectors ω and ω^* and identical vertex sets v, parametrise the same region of a surface on the domain $[0,1] \times [0,1]$, for all $v \in V$, if and only if there exist $\alpha, \beta, \gamma \in \mathbb{R}$ such that:*

$$\begin{aligned}\omega^* = \pm e^{\alpha} diag(\ & 1, e^{\beta}, e^{2\beta}, e^{3\beta}, e^{\gamma}, e^{\gamma+\beta}, e^{\gamma+2\beta}, e^{\gamma+3\beta},\\ & e^{2\gamma}, e^{2\gamma+\beta}, e^{2\gamma+2\beta}, e^{2\gamma+3\beta}, e^{3\gamma}, e^{3\gamma+\beta}, e^{3\gamma+2\beta}, e^{3\gamma+3\beta})\,\omega.\end{aligned}$$

Corollary 3 is concerned with the minimum possible representation of the weights of a bi-cubic.

Corollary 3. *In general, no more than 3 weights of a bi-cubic may be scaled to unit modulus without changing the shape of the surface.*

6 Properties of the Invariant Functions for the Bi-cubics

From the functional forms of ϖ_{ij}, it can be seen that all except ϖ_{23} is a function of a variable, specifically $\omega_{i,j}$, that does not occur in the other functions. It can also be seen that the invariant ϖ_{23} is not a function of the variables that occur uniquely in the other invariants. Hence the functions $\{\varpi_{ij} : (i,j) \in \mathcal{I}\}$ are (manifestly) functionally independent.

The independence of the sign-pattern invariant *sp* from the type ϖ functions may be demonstrated as follows: let ω have components ω_{ij} and define ω^* by

$$\omega^*_{ij} = (-)^{i+j}\omega_{ij}.$$

It is easy to check that $\varpi_{ij}(\omega^*) = \varpi_{ij}(\omega)$ for $(i,j) \in \mathcal{I}$ but, clearly, $sp(\omega^*) \neq sp(\omega)$. Hence the invariance of *sp* is not implied by the invariance of the ϖ_{ij} functions. It follows that the sign-pattern *sp* is an essential, additional independent invariant, for discriminating the geometric equivalence of Bézier surfaces.

Proposition 2. *(i) The functions* $\{\varpi_{ij} : (i,j) \in \mathcal{I}\} \cup \{sp\}$ *constitute a complete, functionally independent set of invariants for the degree* $(3,3)$ *Bézier surfaces. (ii) All complete, functionally independent sets of invariants for the degree* $(3,3)$ *Bézier surfaces have* 14 *elements.*

Proof. By Lemma 4 a complete set of invariants has no more than 14 elements, but $\{\varpi_{ij} : (i,j) \in \mathcal{I}\} \cup \{sp\}$ are independent – hence (i). Invariance of cardinality [6] gives (ii).

7 Canonical-Form Invariants for the Bi-cubic Bézier Surfaces

Complete, independent sets of invariants, that bear a close relationship to the canonical form $\frac{\omega_{i-1}\omega_{i+1}}{\omega_i^2}$, for Bézier curves, are shown to exist for the bi-cubic Bézier surfaces. These invariants have a more immediate geometric interpretation - but do not determine a manifestly independent set. In this section only surfaces with strictly positive weights are considered – hence only the type ϖ invariants appear. Extension of the results to surfaces with weights of mixed-sign requires only the inclusion of the sign-pattern invariant *sp*.

Proposition 3. *The functions* $\varpi_1^c, \ldots, \varpi_{13}^c$ *defined by:*

$$\begin{array}{lll}
\varpi_1^c(\omega) = \frac{\omega_{00}\omega_{20}}{\omega_{10}^2}, & \varpi_2^c(\omega) = \frac{\omega_{00}\omega_{22}}{\omega_{11}^2}, & \varpi_3^c(\omega) = \frac{\omega_{02}\omega_{22}}{\omega_{12}^2}, \\
\varpi_4^c(\omega) = \frac{\omega_{00}\omega_{02}}{\omega_{01}^2}, & \varpi_5^c(\omega) = \frac{\omega_{02}\omega_{20}}{\omega_{11}^2}, & \varpi_6^c(\omega) = \frac{\omega_{20}\omega_{22}}{\omega_{21}^2}, \\
\varpi_7^c(\omega) = \frac{\omega_{10}\omega_{30}}{\omega_{20}^2}, & \varpi_8^c(\omega) = \frac{\omega_{01}\omega_{03}}{\omega_{02}^2}, & \varpi_9^c(\omega) = \frac{\omega_{11}\omega_{33}}{\omega_{22}^2}, \\
\varpi_{10}^c(\omega) = \frac{\omega_{30}\omega_{32}}{\omega_{31}^2}, & \varpi_{11}^c(\omega) = \frac{\omega_{03}\omega_{23}}{\omega_{13}^2}, & \varpi_{12}^c(\omega) = \frac{\omega_{31}\omega_{33}}{\omega_{32}^2}, \\
\varpi_{13}^c(\omega) = \frac{\omega_{13}\omega_{33}}{\omega_{23}^2} & &
\end{array}$$

comprise a complete set of functionally independent invariants for the bi-cubic Bézier surfaces with strictly positive weights.

Proof. The properties of invariance, independence and completeness need to be established for the functions $\{\varpi_1^c, \ldots, \varpi_6^c\}$. (i) Invariance: for example, ϖ_4^c transforms as:

$$\frac{e^{\alpha}\omega_{00}\; e^{\alpha+2\mu}\omega_{02}}{e^{2\alpha}e^{2\mu}\omega_{01}^2} = \frac{\omega_{00}\omega_{02}}{\omega_{01}^2} = \varpi_4^c(\omega).$$

Similarly the remaining ϖ functions. (ii) Independence: it may be shown, using a proprietary symbolic linear algebra package, that the 16 × 13 matrix $d\varpi^c$, defined by:

$$d\varpi^c(\omega) = \begin{bmatrix}
\frac{\partial\varpi_1^c}{\partial\omega_{00}} & \frac{\partial\varpi_2^c}{\partial\omega_{00}} & \frac{\partial\varpi_3^c}{\partial\omega_{00}} & \frac{\partial\varpi_4^c}{\partial\omega_{00}} & \frac{\partial\varpi_5^c}{\partial\omega_{00}} & \frac{\partial\varpi_6^c}{\partial\omega_{00}} & \frac{\partial\varpi_7^c}{\partial\omega_{00}} & \frac{\partial\varpi_8^c}{\partial\omega_{00}} & \frac{\partial\varpi_9^c}{\partial\omega_{00}} & \frac{\partial\varpi_{10}^c}{\partial\omega_{00}} & \frac{\partial\varpi_{11}^c}{\partial\omega_{00}} & \frac{\partial\varpi_{12}^c}{\partial\omega_{00}} & \frac{\partial\varpi_{13}^c}{\partial\omega_{00}} \\
\frac{\partial\varpi_1^c}{\partial\omega_{01}} & \frac{\partial\varpi_2^c}{\partial\omega_{01}} & \frac{\partial\varpi_3^c}{\partial\omega_{01}} & \frac{\partial\varpi_4^c}{\partial\omega_{01}} & \frac{\partial\varpi_5^c}{\partial\omega_{01}} & \frac{\partial\varpi_6^c}{\partial\omega_{01}} & \frac{\partial\varpi_7^c}{\partial\omega_{01}} & \frac{\partial\varpi_8^c}{\partial\omega_{01}} & \frac{\partial\varpi_9^c}{\partial\omega_{01}} & \frac{\partial\varpi_{10}^c}{\partial\omega_{01}} & \frac{\partial\varpi_{11}^c}{\partial\omega_{01}} & \frac{\partial\varpi_{12}^c}{\partial\omega_{01}} & \frac{\partial\varpi_{13}^c}{\partial\omega_{01}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & & & & & & & \\
\frac{\partial\varpi_1^c}{\partial\omega_{33}} & \frac{\partial\varpi_2^c}{\partial\omega_{33}} & \frac{\partial\varpi_3^c}{\partial\omega_{33}} & \frac{\partial\varpi_4^c}{\partial\omega_{33}} & \frac{\partial\varpi_5^c}{\partial\omega_{33}} & \frac{\partial\varpi_6^c}{\partial\omega_{33}} & \frac{\partial\varpi_7^c}{\partial\omega_{33}} & \frac{\partial\varpi_8^c}{\partial\omega_{33}} & \frac{\partial\varpi_9^c}{\partial\omega_{33}} & \frac{\partial\varpi_{10}^c}{\partial\omega_{33}} & \frac{\partial\varpi_{11}^c}{\partial\omega_{33}} & \frac{\partial\varpi_{12}^c}{\partial\omega_{33}} & \frac{\partial\varpi_{13}^c}{\partial\omega_{33}}
\end{bmatrix}$$

and having the explicit form $d\varpi^c(\omega) = [d\varpi^c(\omega)_L; d\varpi^c(\omega)_R]$ where

$$d\varpi^c(\omega)_L = \begin{matrix}
\frac{\omega_{02}}{\omega_{10}^2} & \frac{\omega_{22}}{\omega_{11}^2} & 0 & \frac{\omega_{02}}{\omega_{01}^2} & 0 & 0 \\
0 & 0 & 0 & -2\frac{\omega_{00}\omega_{02}}{\omega_{01}^3} & 0 & 0 \\
0 & 0 & \frac{\omega_{22}}{\omega_{12}^2} & \frac{\omega_{00}}{\omega_{01}^2} & \frac{\omega_{20}}{\omega_{11}^2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
-2\frac{\omega_{00}\omega_{20}}{\omega_{10}^3} & 0 & 0 & 0 & 0 & 0 \\
0 & -2\frac{\omega_{00}\omega_{22}}{\omega_{11}^3} & 0 & 0 & -2\frac{\omega_{02}\omega_{20}}{\omega_{11}^3} & 0 \\
0 & 0 & -2\frac{\omega_{02}\omega_{22}}{\omega_{12}^3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\omega_{00}}{\omega_{10}^2} & 0 & 0 & 0 & \frac{\omega_{02}}{\omega_{11}^2} & \frac{\omega_{22}}{\omega_{21}^2} \\
0 & 0 & 0 & 0 & 0 & -2\frac{\omega_{20}\omega_{22}}{\omega_{21}^3} \\
0 & \frac{\omega_{00}}{\omega_{11}^2} & \frac{\omega_{02}}{\omega_{12}^2} & 0 & 0 & \frac{\omega_{20}}{\omega_{21}^2} \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

and

$$d\varpi^c(\omega)_R = \begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\omega_{03}}{\omega_{02}^2} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{-2\omega_{01}\omega_{03}}{\omega_{02}^2} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\omega_{01}}{\omega_{02}^2} & 0 & 0 & \frac{\omega_{23}}{\omega_{13}^2} & 0 & 0 \\
0 & \frac{\omega_{01}}{\omega_{02}^2} & 0 & 0 & \frac{\omega_{23}}{\omega_{13}^2} & 0 & 0 \\
0 & 0 & \frac{\omega_{33}}{\omega_{22}^2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{-2\omega_{03}\omega_{23}}{\omega_{13}^2} & 0 & \frac{\omega_{33}}{\omega_{23}^2} \\
\frac{-2\omega_{10}\omega_{30}}{\omega_{20}^3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{-2\omega_{11}\omega_{33}}{\omega_{22}^3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{\omega_{03}}{\omega_{13}^2} & 0 & \frac{-2\omega_{13}\omega_{33}}{\omega_{23}^3} \\
\frac{\omega_{01}}{\omega_{20}^2} & 0 & 0 & \frac{\omega_{32}}{\omega_{31}^2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{-2\omega_{30}\omega_{32}}{\omega_{31}^3} & 0 & \frac{\omega_{33}}{\omega_{32}^2} & 0 \\
0 & 0 & 0 & \frac{\omega_{30}}{\omega_{31}^2} & 0 & \frac{-2\omega_{31}\omega_{33}}{\omega_{32}^3} & 0 \\
0 & 0 & \frac{\omega_{11}}{\omega_{22}^2} & 0 & 0 & \frac{\omega_{31}}{\omega_{32}^2} & \frac{\omega_{13}}{\omega_{23}^2}
\end{matrix}$$

is of maximal rank – i.e., 13. It follows that the functions $\varpi_1, \ldots, \varpi_{13}$ are functionally independent. Completeness follows from the invariance of cardinality property.

The invariants $\varpi_1^c, \ldots, \varpi_6^c$ are associated with the lower left 9 vertices (see Figure 1), and correspond to an 'embedded' bi-quadratic subset. It can be shown that these 6 functions determine an independent set of invariant functions for the rational bi-quadratic Bézier surfaces [4].

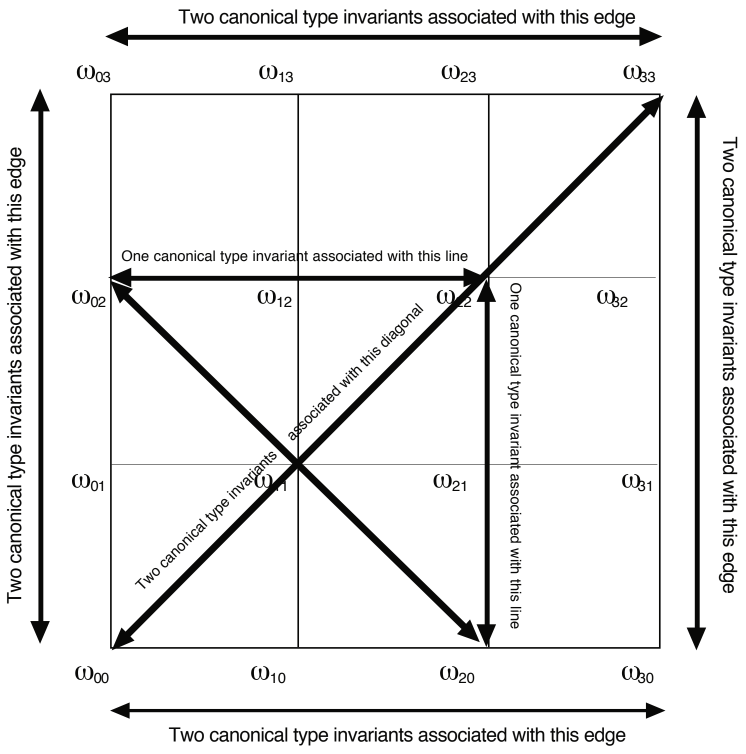

Two canonical type invariants associated with this edge
ω03
ω13
ω23
ω33
Two canonical type invariants associated with this edge
Two canonical type invariants associated with this edge
One canonical type invariant associated with this line
ω02
ω12
ω22
ω32
Two canonical type invariants associated with this diagonal
One canonical type invariant associated with this line
ω01
ω11
ω21
ω31
ω00
ω10
ω20
ω30
Two canonical type invariants associated with this edge

Fig. 1. Edges associated with the type ϖ^c invariants of the bi-cubic Bézier surfaces

8 A Lie Group Perspective

8.1 The Fundamental Theorem

The results obtained may be seen in the context of the Fundamental Theorem on the invariants of Lie groups [6]. The theorem, in essence, states that:

'If G is a Lie group acting on an k-dimensional manifold X with q-dimensional orbits, then for each point $x \in X$ there exist $k - q$ functions $\iota_1, \ldots, \iota_{k-q}$ that are invariant and functionally independent in some neighbourhood N_x of x. Any other invariant ι, defined near x, may be uniquely expressed as $\iota = F(\iota_1, \ldots, \iota_{k-q})$ for some F.'

The invariants of the Fundamental Theorem are 'local'; i.e., they distinguish the orbits near x in the sense that two points $x, x^* \in N_x$ lie on the same orbit if and only if all the fundamental invariants agree: i.e.,

$$\iota_1(x) = \iota_1(x^*), \ldots, \iota_{k-q}(x) = \iota_{k-q}(x^*).$$

The theorem provides information on the cardinality of complete, independent sets of invariants – but does not provide a means to determine their explicit functional forms.

8.2 Application to Bézier Paths and Bi-cubic Surfaces

(i) In the context of Bézier paths, the manifold is the disconnected $(n+1)$-dimensional weight-space:

$$\mathcal{O}_n = \{\omega = (\omega_0, \ldots, \omega_n) : \omega_i \neq 0\} \subset \mathbb{R}^{n+1}$$

with 2^{n+1} connected components. The transformation group is the 2-dimensional Lie group defined by the matrices $\pm e^{\alpha} diag(1, e^{\mu}, \ldots, e^{(n-1)\mu}, e^{n\mu})$. As the stabilizer of the group action on $\mathcal{O}_n$ is trivial, it follows that the orbits have the same dimension as the group – i.e., 2. The theorem states that in these circumstances $(n+1) - 2 = (n-1)$ 'local' invariants exist – these correspond to the type ϖ (or type ς) functions derived in this paper. The function *sp* is a global invariant and, therefore, not addressed by the theorem; *sp* distinguishes orbits globally – i.e., on the components of the disconnected manifold $\mathcal{O}_n$, where the type ς (or type ϖ) are not discriminating.

(ii) For the bi-cubic surfaces the manifold is the disconnected 16 dimensional weight-space:

$$\mathcal{O}_{3,3} = \{(\omega_{00}, \ldots, \omega_{33}) : \omega_{ij} \neq 0\} \subset \mathbb{R}^{16}$$

with 2^{16} components. The transformation group is the three-dimensional Lie subgroup of $GL(16, \mathbb{R})$ defined by the matrices:

$$\pm e^{\alpha} diag(1, e^{\gamma}, \ldots, e^{3\gamma}; e^{\beta}, e^{\beta+\gamma}, \ldots, e^{\beta+3\gamma}; \ldots; e^{3\beta}, e^{3\beta+\gamma}, \ldots, e^{3\beta+3\gamma}).$$

The stabilizer is trivial and the orbits are therefore of dimension 3. The theorem states that in these circumstances $16-3$ 'local' invariants exist – these correspond to the 13 type ϖ functions derived for the bi-cubics. The invariant *sp* is global, and distinguishes orbits on the components of $\mathcal{O}_{3,3}$.

9 Examples

The examples pertain to the $\frac{1}{4}$-cyclide defined by: (i) the geometric parameters $a = 6, b = 4\sqrt{2}, m = 3$, and (ii) the angular limits $\theta_0 = 0, \theta_1 = \pi$ and $\phi_0 = 0, \phi_1 = \pi$. Figure 2 shows a rational bi-cubic parametrisation of this patch induced by the author's construction [5]. The weights, ω_{ij}, of the induced parametrisation are:

$$\begin{array}{l} \omega_{00} = 4 \ , \omega_{01} = \frac{4}{3} \ , \omega_{02} = \frac{8}{3} \ , \ \omega_{03} = 8 \\ \omega_{10} = \frac{4}{3} \ , \omega_{11} = \frac{4}{3} \ , \omega_{12} = \frac{8}{9} \ , \ \omega_{13} = \frac{8}{3} \\ \omega_{20} = \frac{8}{3} \ , \omega_{21} = \frac{8}{9} \ , \omega_{22} = \frac{4}{9} \ , \ \omega_{23} = \frac{4}{3} \\ \omega_{30} = 8 \ , \omega_{31} = \frac{8}{3} \ , \omega_{32} = \frac{4}{3} \ , \ \omega_{33} = 4; \end{array}$$

the co-ordinates of the vertices, v_{ij}, of the parametrisation are given in the Appendix to the paper.

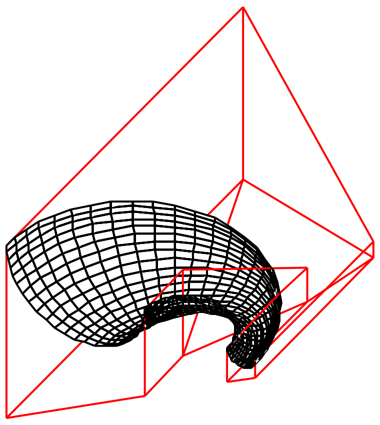


Fig. 2. Induced bi-cubic $\frac{1}{4}$ cyclide parametrisation with Bézier polygon

Example 2. *If $\omega^*_{i,j}$ are defined by*

$$\begin{array}{llll}
\omega^*_{00} = 4 , & \omega^*_{01} = \frac{8}{3} , & \omega^*_{02} = \frac{32}{3} , & \omega^*_{03} = 64 \\
\omega^*_{10} = \frac{2}{3} , & \omega^*_{11} = \frac{4}{3} , & \omega^*_{12} = \frac{16}{9} , & \omega^*_{13} = \frac{32}{3} \\
\omega^*_{20} = \frac{2}{3} , & \omega^*_{21} = \frac{4}{9} , & \omega^*_{22} = \frac{4}{9} , & \omega^*_{23} = \frac{8}{3} \\
\omega^*_{30} = 1 , & \omega^*_{31} = \frac{2}{3} , & \omega^*_{32} = \frac{2}{3} , & \omega^*_{33} = 4
\end{array}$$

then $\varpi_i(\omega) = \varpi_i(\omega^*)$ for all $1 \leq i \leq 13$. Hence, by Corollary 1, $\{(v_{ij}, \omega_{ij}) : 0 \leq i, j \leq 3\}$ and $\{(v_{ij}, \omega^*_{ij}) : 0 \leq i, j \leq 3\}$ determine the same cyclide patch.

Example 3. *The 16×16 diagonal matrix below transforms the weights $\omega_{00}, \omega_{03}, \omega_{30}$, of any bi-cubic surface, to unit modulus.*

$$diag(\frac{1}{|\omega_{00}|}, \left(\frac{1}{|\omega_{00}|}\right)^{\frac{2}{3}} \left(\frac{1}{|\omega_{03}|}\right)^{\frac{1}{3}}, \left(\frac{1}{|\omega_{00}|}\right)^{\frac{1}{3}} \left(\frac{1}{|\omega_{03}|}\right)^{\frac{2}{3}}, \frac{1}{|\omega_{03}|}, \left(\frac{1}{|\omega_{00}|}\right)^{\frac{2}{3}} \left(\frac{1}{|\omega_{30}|}\right)^{\frac{1}{3}},$$

$$\left(\frac{1}{|\omega_{00}|}\right)^{\frac{1}{3}} \left(\frac{1}{|\omega_{30}|}\right)^{\frac{1}{3}} \left(\frac{1}{|\omega_{03}|}\right)^{\frac{1}{3}}, \left(\frac{1}{|\omega_{30}|}\right)^{\frac{1}{3}} \left(\frac{1}{|\omega_{03}|}\right)^{\frac{2}{3}}, \left(\frac{|\omega_{00}|}{|\omega_{30}|}\right)^{\frac{1}{3}} \frac{1}{|\omega_{03}|},$$

$$\left(\frac{1}{|\omega_{00}|}\right)^{\frac{1}{3}} \left(\frac{1}{|\omega_{30}|}\right)^{\frac{2}{3}}, \left(\frac{1}{|\omega_{30}|}\right)^{\frac{2}{3}} \left(\frac{1}{|\omega_{03}|}\right)^{\frac{1}{3}}, |\omega_{00}|^{\frac{1}{3}} \left(\frac{1}{|\omega_{30}|}\right)^{\frac{2}{3}} \left(\frac{1}{|\omega_{03}|}\right)^{\frac{2}{3}},$$

$$\left(\frac{|\omega_{00}|}{|\omega_{30}|}\right)^{\frac{2}{3}} \frac{1}{|\omega_{03}|}, \frac{1}{|\omega_{30}|}, \frac{1}{|\omega_{30}|} \left(\frac{|\omega_{00}|}{|\omega_{03}|}\right)^{\frac{1}{3}}, \frac{1}{|\omega_{30}|} \left(\frac{|\omega_{00}|}{|\omega_{03}|}\right)^{\frac{2}{3}}, \left(\frac{|\omega_{00}|}{|\omega_{30}|}\right) \frac{1}{|\omega_{03}|});$$

for the cyclide patch we obtain:

$$\begin{array}{llll}
\omega_{00} = 1 , & \omega_{01} = \frac{1}{6} \sqrt[3]{4} , & \omega_{02} = \frac{1}{3} \sqrt[3]{2} , & \omega_{03} = 1 \\
\omega_{10} = \frac{1}{6} \sqrt[3]{4} , & \omega_{11} = \frac{1}{18} \sqrt[3]{2} , & \omega_{12} = \frac{1}{9} , & \omega_{13} = \frac{1}{6} \sqrt[3]{4} \\
\omega_{20} = \frac{1}{3} \sqrt[3]{2} , & \omega_{21} = \frac{1}{9} , & \omega_{22} = \frac{1}{36} \sqrt[3]{4} , & \omega_{23} = \frac{1}{12} \sqrt[3]{2} \\
\omega_{30} = 1 , & \omega_{31} = \frac{1}{6} \sqrt[3]{4} , & \omega_{32} = \frac{1}{12} \sqrt[3]{2} , & \omega_{33} = \frac{1}{4} .
\end{array}$$

10 Further Work

It is anticipated that the computational approach will generalize to to the determination of the invariant-geometry conditions of: (i) rectangular Bézier surfaces of arbitrary degree (n, m), (ii) Bézier triangles of arbitrary degree and (iii) Bézier volumes of arbitrary degree.

References

1. Patterson, R.R.: Projective transformations of the parameter of a Bernstein-Bézier curve. ACM Trans. on Graphics 4(4), 276–290 (1985)
2. Bez, H.E.: Generalised invariant-geometry conditions for the rational Bézier paths. International Journal of Computer Mathematics (to appear)
3. Farin, G.: NURBS from Projective Geometry to Practical Use, 2nd edn. A.K.Peters (1999)
4. Bez, H.E.: Invariant-geometry conditions for the rational bi-quadratic Bézier surfaces (submitted for publication) (2008)
5. Bez, H.E.: Bounded domain bi-quadratic rational parametrisations of Dupin cyclides. International Journal of Computer Mathematics 85(7), 1097–1111 (2008)
6. Olver, P.J.: Classical Invariant Theory. London Mathematical Society Student Texts, vol. 44. Cambridge University Press, Cambridge (1999)

Appendix

The vertices, $v_{ij} = (x_{ij}, y_{ij}, z_{ij})$, of the cyclide patch used in the applications are:

$$\begin{array}{llllllll}
x_{00} = 5, & x_{01} = \,, & x_{02} = 7, & x_{03} = 7, & x_{10} = 5, & x_{11} = 5, & x_{12} = 7, & x_{13} = 7 \\
x_{20} = -1, & x_{21} = -1, & x_{22} = -11, & x_{23} = -11, & x_{30} = -1, & x_{31} = -1, & x_{32} = -11, & x_{33} = -11
\end{array}$$

$$\begin{array}{llllllll}
y_{00} = 0, & y_{01} = 0, & y_{02} = 0, & y_{03} = 0, & y_{10} = 6\sqrt{2}, & y_{11} = 6\sqrt{2}, & y_{12} = 9\sqrt{2}, & y_{13} = 9\sqrt{2} \\
y_{20} = 3\sqrt{2}, & y_{21} = 3\sqrt{2}, & y_{22} = 18\sqrt{2}, & y_{23} = 18\sqrt{2}, & y_{30} = 0, & y_{31} = 0, & y_{32} = 0, & y_{33} = 0
\end{array}$$

$$\begin{array}{llllllll}
z_{00} = 0, & z_{01} = -2\sqrt{2}, & z_{02} = -\sqrt{2}, & z_{03} = 0, & z_{10} = 0, & z_{11} = -2\sqrt{2}, & z_{12} = -\sqrt{2}, & z_{13} = 0 \\
z_{20} = 0, & z_{21} = -5\sqrt{2}, & z_{22} = -10\sqrt{2}, & z_{23} = 0, & z_{30} = 0, & z_{31} = -5\sqrt{2}, & z_{32} = -10\sqrt{2}, & z_{33} = 0.
\end{array}$$

Crazy Cuts: Dissecting Planar Shapes into Two Identical Parts

A.M. Bruckstein[1,⋆] and D. Shaked[2]

[1] Computer Science Department, Technion-IIT, Haifa, Israel
[2] HP-labs, Haifa, Israel
freddy@cs.technion.ac.il, doron.shaked@hp.com

Abstract. We analyze a well known type of puzzle in planar geometry: given a planar shape, it is required to find a cut that divides the shape into two identical parts (up to rotation and translation). Clearly not all shapes can be so dissected and for some shapes that appear in puzzles the cutting curve is quite surprising and difficult to find. In this paper we first analyze the inverse problem of assembling planar shapes from two identical parts having partially 'matching' boundaries and then use the insights gained on this topic to derive an efficient algorithm to solve the dissection puzzle in quite general situations.

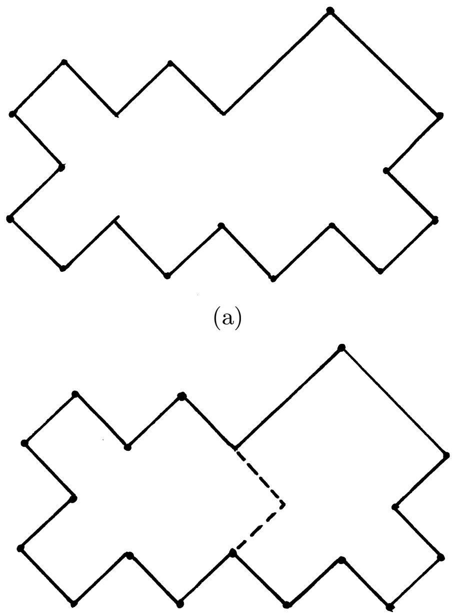
1 Introduction

Consider the planar shape depicted in Figure 1a. The goal is to find a cutting curve that divides this shape into two identical shapes. The solution is seen in Figure 1b, and is not trivial to find. Several other examples of crazy-cut dissection puzzles are shown in Figure 2. The question we address in this paper is the following: given a planar shape, determine whether a simple cutting curve exists dissecting the shape into two identical parts and, if it exists, find the cutting curve efficiently. To answer this question we shall first analyze the inverse problem of assembling a planar shape from two identical shapes that have partially 'matching' boundaries. This problem may be regarded as solving a simple jigsaw puzzle of two pieces (with no drawings on them).

2 Solving Two Piece Jigsaw Puzzles

A simple planar shape (with no holes) may be represented by the closed planar curve of its boundary. Suppose we have a Euclidean invariant description of the closed boundary of the two (identical) shapes we must put together. The description can be the curvature vs arclength 'invariant signature' description $k(s)$, where $s \in [0, L]$ is the arclength along the boundary (L being the total length of the planar boundary that will be assumed measurable, and $s = 0$ being an arbitrarily selected starting point on the boundary). If the boundary

⋆ Corresponding author.


(a)

(b)

Fig. 1. (a) The shape and (b) the 'crazy-cut' into two identical parts (after Martin Gardner)

is non-smooth we can defined $k(s)$ as having δ- function components describing sharp angles at breakpoint or we might assume that we have an equivalent description of the boundary via $k(s)$ between breakpoints along with the turn angle information at each breakpoint. If we have to deal with polygonal planar shapes, the 'boundary signature' may be a sequence of edge-length (l_i) and the turn angles (φ_i) at each vertex of the polygon (see Figure 3). For shapes digitized on $\mathbb{Z}^2$ the natural boundary signature is the so-called crack-code of the boundary, tracing the boundary of the shape built from adjacent square pixels of size (1×1) having integer-coordinate vertices.

We may now ask what characterizes, in terms of the invariant signature function, for instance the Euclidian-invariant curvature $k(s)$, matching portions of boundaries of two given shapes S_I and S_{II}. If $k^I(s)$ and $k^{II}(s)$ describe the boundaries of the two shapes in a clockwise traversal from arbitrary initial conditions and the portion between s^I_A and s^I_B on the boundary of S_I matches the portion between s^{II}_A to s^{II}_B we shall have (see Figure 4) that:

$$k^I(s) = -k^{II}(\Sigma - s) \quad s \in [s^I_A, s^I_B].$$

since clearly along the common boundary portions of the shapes S_I and S_{II} we have the same traversal rate (as arclength traversal implies unit speed clockwise

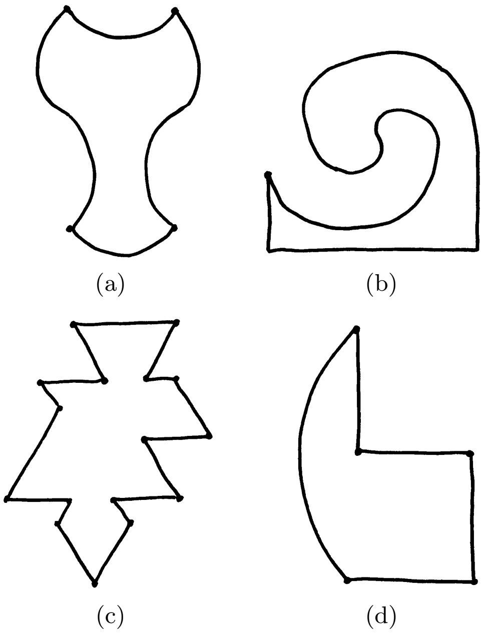


Fig. 2. Some crazy-cut challenges

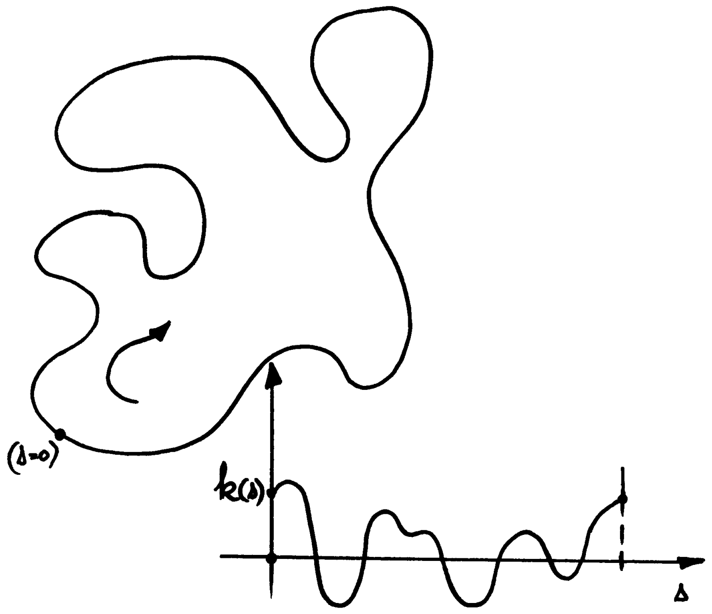
travel along the boundary!) but the velocity vector at each point turns in opposite directions. Note that we also have

$$\Sigma - s_A^I = s_B^{II}$$
$$\Sigma - s_B^I = s_A^{II}$$

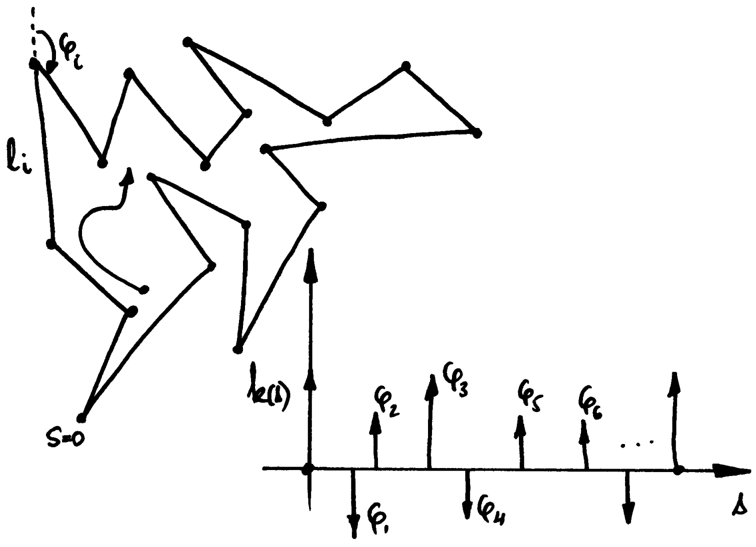
and therefore

$$\Sigma = s_A^I + s_B^{II} = s_A^{II} + s_B^I.$$

Indeed if we plot $k^I(s)$ as a periodic function of s with period equal to L_I, the length of the boundary of shape S_I, and similarly $k^{II}(s)$ as a periodic function of s with period $L_{II}, \ldots$ the picture looks like that illustrated in Figure 5. We see that for both S_I and S_{II} we can regard the signature function $k(s)$ as being composed of alternating parts $\mathbf{P_I}$ and $\mathbf{J_I}$ and $\mathbf{P_{II}}$ and $\mathbf{J_{II}}$ where $\mathbf{J_I}$ and $\mathbf{J_{II}}$, the matching portions, have same length L_J and are (up/down) and (left/right) mirror reflections of each other. The joint object, after 'docking' the two jigsaw puzzle pieces together will have a signature function that can be described by $\mathbf{P_I}$ followed by $\mathbf{P_{II}}$ with a total length of $L_I + L_{II} - 2L_J$.

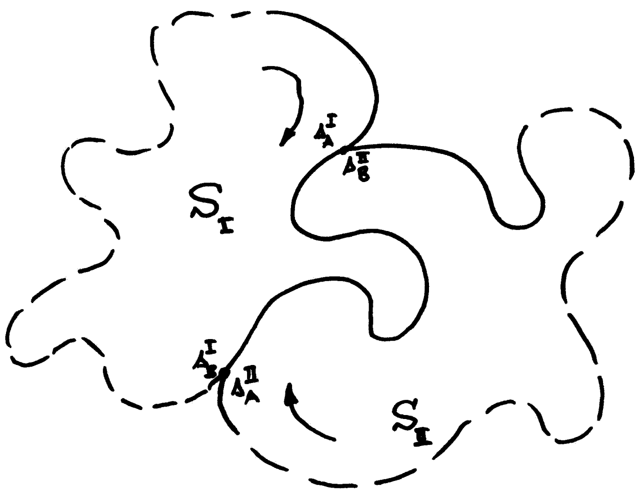


(a)

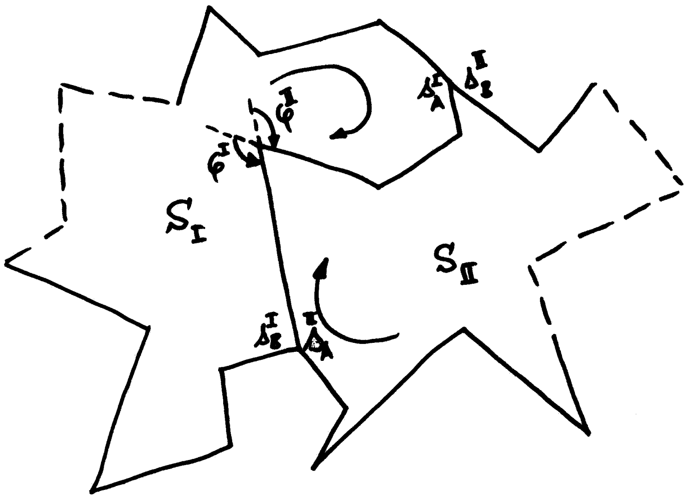


(b)

Fig. 3. (a) Euclidean invariant boundary signatures for shape description (smooth case). (b) Polygonal case.



(a)



(b)

Fig. 4. (a) Smooth shape dockings (b) Polygonal shapes

3 Self Docking of Shapes

Up to this point the discussion was for two general shapes S_I and S_{II}. However we are interested in matching identical shapes, i.e. $S_I \equiv S_{II} \equiv S$. In this case we shall have to have for $k(s)$, the invariant signature description of the boundary of S, that

$$k(s) = -k(\Sigma - s) \quad \text{for} \quad s \in [0, \Delta] \tag{1}$$

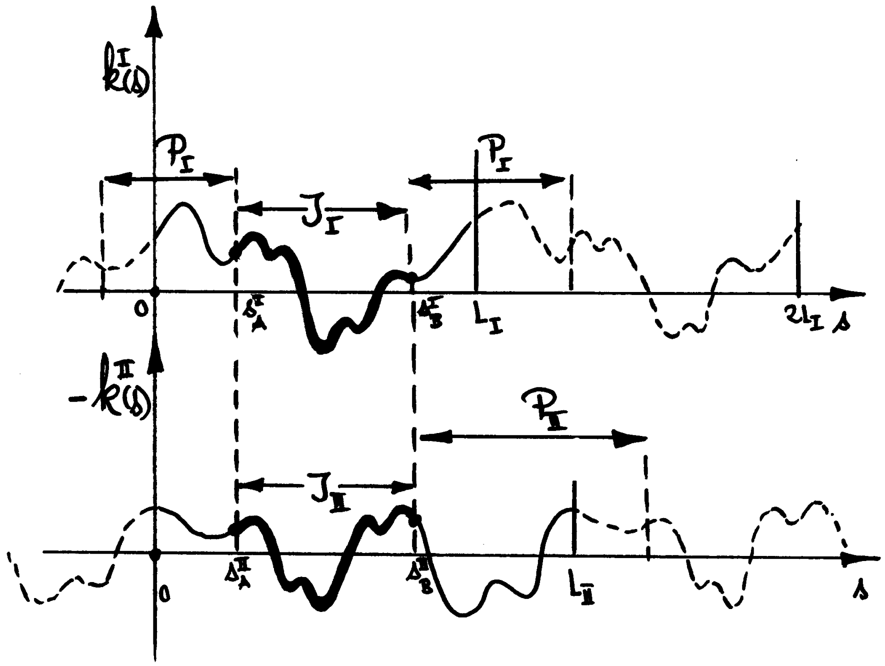


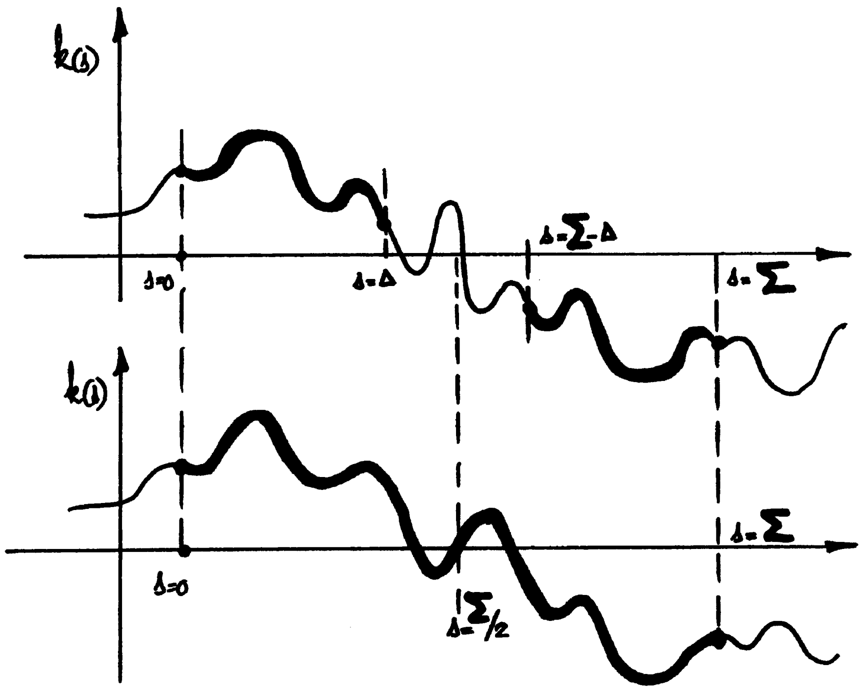
Fig. 5. The signatures of two shapes S_I and S_{II} showing the docking portions $\mathbf{J_I}$ and $\mathbf{J_{II}}$ and the portions $\mathbf{P_I}$ and $\mathbf{P_{II}}$ that will make the boundary of the joint shape

(where we decided w.l.o.g. to start the arclength parametrization at $s_A = 0$). Here, however, something interesting can be observed: if the intervals $[0, \Delta]$ and $[\Sigma - \Delta, \Sigma]$ are disjoint it means that there are two distinct portions on the shape's boundary that can be matched, but if the intervals overlap (i.e. $\Delta > \frac{\Sigma}{2}$ and consequently $\Sigma - \Delta < \frac{\Sigma}{2}$) we necessarily get that $k(s) = -k(\Sigma - s)$ for all the interval $[0, \Sigma]$. Indeed for $s = \frac{\Sigma}{2}$ in (1) we'll have that $k(\frac{\Sigma}{2}) = -k(\frac{\Sigma}{2}) = 0$, and more importantly that $k(\tilde{s}) = -k(\Sigma - \tilde{s})$ for $\tilde{s} \in [\Delta, \Sigma]$ too by realizing that this is simply reading equation (1) with sides interchanged, i.e. redefining the arclength $\tilde{s}$ via $\tilde{s} = \Sigma - s$. (See Figure 6). Note that in the above discussion we assume that the interval $[0, \Sigma]$ is maximal, in the sense that $k(0-\varepsilon) \neq -k(\Sigma+\varepsilon)$. The above considerations prove the following

Self Docking Dichotomy Lemma

A given planar shape either 'docks' to itself over totally disjoint matching portions of its boundary or over the exact same portion of its boundary and it cannot possibly have selfdockings that match over boundary portions that are only partially disjoint (i.e. different boundary intervals that have a common boundary portion).

(a)



(b)

Fig. 6. Self docking dichotomy: (a) disjoint matching intervals (b) same boundary portion matching

Also note that if the shape 'docks' to itself on the same portion of its boundary we shall always have at the midpoint of the match 'an inflexion' point of zero curvature.

The consequences of these observations are far-reaching indeed, with regard to the boundary of the composite shape obtained after docking two identical parts together. If the docking was over the same portion of the boundaries of the component shapes the outer boundary will necessarily be the concatenation of two identical boundary curves, (see Figure 7). In this case the cut curve is completely 'out of sight', i.e hidden inside the composite shape and in fact any symmetrical cut from M to M' (in Figure 7) will yield a possible solution.

So far we have seen that the case of self-docking along the same portion of the boundary is the trivial case of cutting a shape that has two identical curves joining together to form the composite boundary. The interesting case occurs when the self docking is along disjoint portions of the boundary. This case is illustrated in Figure 8. Let us call the matching portions $\mathbf{J}$ and $\bar{\mathbf{J}}$, and we assume that $\mathbf{J}$ and $\bar{\mathbf{J}}$ are different segments of $k(s)$, i.e. they correspond to two intervals $[s_A, s_B]$ and $[\bar{s}_A, \bar{s}_B]$ that are of the same length but disjoint in

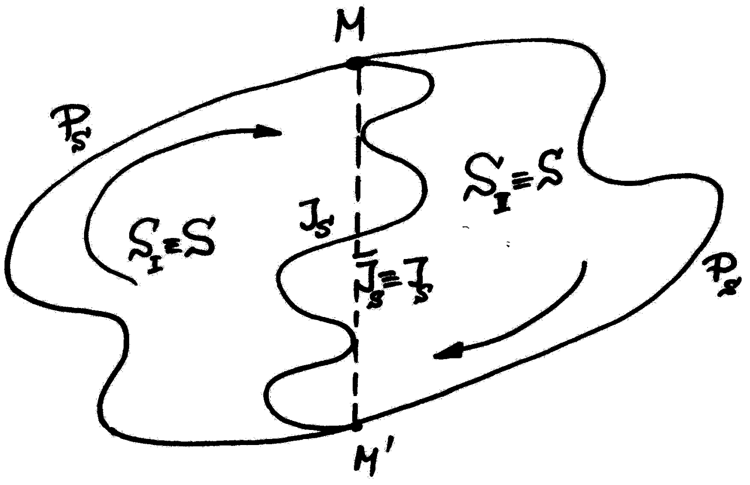


Fig. 7. 'Self-docking' over the same boundary portion of the two identical shapes S and S. The self docking boundary portion $\mathbf{J}$ is completely hidden inside the composite shape, whose boundary is of the form $\mathbf{P_s P_s}$, where $\mathbf{P_s}$ is the free portion of the boundary of $S : \mathbf{P_s J_s}$.

$s \in [0, L]$. Without loss of generality let us take $s_A = 0$. Then the boundary of S will comprise the intervals

$$\mathbf{J}_{[s_A=0,s_B]}\ \mathbf{P}_{[s_B \bar{s}_A]}\ \bar{\mathbf{J}}_{[\bar{s}_A \bar{s}_B]}\ \mathbf{Q}_{[\bar{s}_B,L]} \text{ in cyclic order.}$$

Using this motivation we will have that the two identical shapes : $S_1 : \mathbf{JP\bar{J}Q}$ and $S_2 : \mathbf{JP\bar{J}Q}$ when docked so as to have $\mathbf{J}$ matched to $\bar{\mathbf{J}}$ will result in a combined shape with boundary described by the following syntax

$$(S_{\text{combined}}) : \mathbf{P\bar{J}QQJP} \sim \mathbf{\bar{J}QQJPP} \sim \mathbf{QJPP\bar{J}Q}$$

where $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{J}$, $\bar{\mathbf{J}}$ are the $k(s)$ portions that describe the original (component piece) S. Hence if a shape can be represented as the docking of two identical jigsaw-puzzle pieces its boundary is either of the form:

$$(S_{\text{combined}}) : \mathbf{PP} \text{ if } S : \mathbf{JP} \equiv \mathbf{\bar{J}P} \quad \text{(in this case } \mathbf{J} \equiv \bar{\mathbf{J}}\text{)}$$

or of the form:

$$(S_{\text{combined}}) : \mathbf{P\bar{J}QQJP} \text{ if } S : \mathbf{JP\bar{J}Q} \quad \text{(in this case } \mathbf{J} \neq \bar{\mathbf{J}}\text{).}$$

4 Finding Crazy Cuts

The structure of the invariant signature representing the combined shape yields an efficient algorithm for finding the crazy cut of a planar shape if such a cut

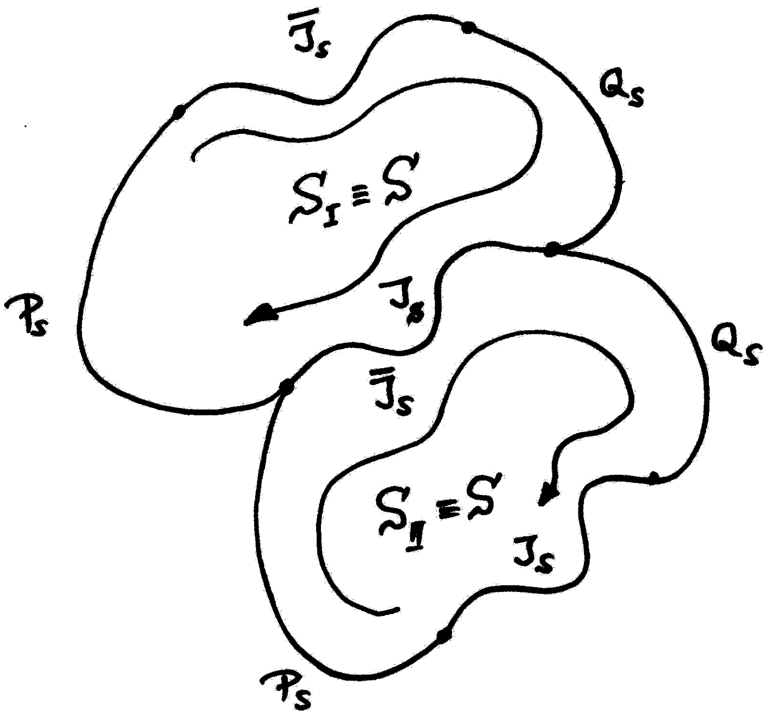


Fig. 8. 'Self-docking' over disjoint boundary portions of the two identical shapes $S_I \equiv S_{II} \equiv S$. The 'self-docking' portions $\mathbf{J}$ and $\bar{\mathbf{J}}$ are visible in the composite shape whose boundary has the form $\mathbf{P_s \bar{J}_s Q_s Q_s J_s P_s}$ where the boundary of S is $\mathbf{P_s \bar{J}_s Q_s J_s}$.

exists, or for determining that such a cut is not possible. Indeed when we are given a $k(s)$ or any Euclidean invariant signature representation of the composite boundary we have to determine whether it has, from some starting point the structure $\mathbf{PP}$ or the structure $\mathbf{P\bar{J}QQJP}$. In fact, exhaustive search for a given string of length L, which would test all starting points and all possible internal distributions of lengths of $\mathbf{P}, \bar{\mathbf{J}}, \mathbf{Q}, \mathbf{J}$ would be feasible and quite efficient if the signature was discrete. All starting point possibilities will involve L runs of checking whereas $l(\mathbf{P})$ and $l(\mathbf{J}) \equiv l(\bar{\mathbf{J}})$ are two additional parameters that are needed to be set. (Recall that $l(\mathbf{P}) + l(\mathbf{J}) + l(\bar{\mathbf{J}}) + l(\mathbf{Q}) = L$ hence these two parameters also determine $l(\mathbf{Q})$!). Hence we can test a string of length L for the structure $\mathbf{P\bar{J}QQJP}$ with an exhaustive search algorithm of $O(L^4)$ complexity, and this without any sophisticated string manipulation optimization.

Notice that the additional (fourth) $O(L)$ complexity is due to the need to verify the syntax for every choice. In the sequel we will argue that the latter can be included in the $O(L^3)$ search leading to an overall complexity of $O(L^3)$.

4.1 An Efficient Algorithm for Finding Crazy Cuts for Polygons

In this section we detail the crazy cut algorithm for a polygonal shape. A polyline description may start by specifying the coordinates of the first vertex and the direction of the first edge. If such initialization is omitted, we may, w.l.o.g. place the vertex in the origin, and orient the first edge in the positive $\hat{x}$ direction. Following the initialization we traverse the boundary of the shape in a clockwise manner, specifying L couples (l_i, φ_i) of edge lengths l_i and turn angles φ_i as see for example in Figure 3b.

Notice that since we know that the polyline describes a closed shape the length of the last edge and the turn angle of the two last segments are redundant. For completeness one may assume they are given, and we may verify that the data agrees with the closed curve assumption.

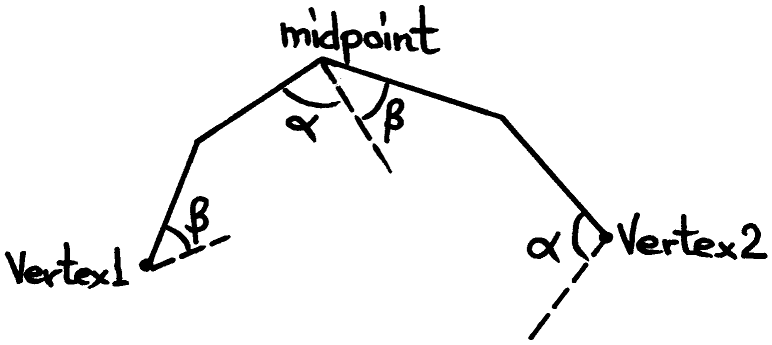
We start the algorithm with a choice of two vertices and the boundary polyline segment connecting the first vertex to the second vertex in a clockwise manner. There are $O(L^2)$ possible vertex choices.

For each choice of vertices we will check the possibility that the connecting boundary segment contains the two segments **PP** in the boundary syntax sequence $\mathbf{QJPP\bar{J}Q}$. To do that, we will cut the initial boundary segment into two segments of equal length, and compare them. They are qualified to be together the **PP** segment if both halves are similar segments. Traversed clockwise on both halves edges should be of the same length, and with the same turn angles. Additionally, the internal angles on both ends of the initial segment should sum up to the internal angle corresponding the midpoint of the segment, since the midpoint should be the location where the **J** segment of one half shape meets the $\mathbf{\bar{J}}$ segment of the other half (see Figure 9a).

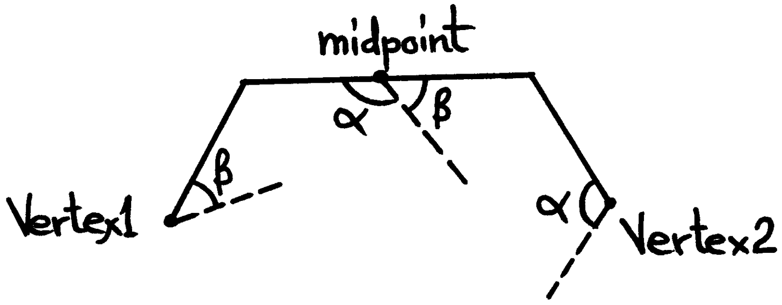
This is the time to review a couple of interesting special cases that may affect this part of the algorithm.

1. The boundary segment **PP** may be trivially short, that is, it may well be a single vertex. Indeed, a vertex of a shape may often be a pivot point around which one copy of a shape turns, whereby one side of the vertex matches the other side (see e.g. Figure 10). Therefore every vertex constitutes one successful virtual selection of the two vertices above.
2. The midpoint of the segment does not have to occur at a vertex. It can well be inside an edge of the full polyline which may be the adjoining point of two vertices complementing each other to 180^0, (see e.g. Figure 9b).
3. The end point of one of the segments can occur on an edge. It should be noted that in this case the other end (corresponding to the other end of the segment **P**) has to coincide with a vertex. Note also that in this case the interface between the two copies of **P** occurs on a vertex whose internal angle is the sum of the vertex angle and 180^0, see Figure 11a . The next two paragraphs will deal with this special case.

   To cover the cases described in the last item, the first part of the algorithm has to be repeated. This time again two vertices are selected, and the segment between them is assumed to be the first copy of **P**. To check this,



(a)



(b)

Fig. 9. **PP** segment candidates and angle conditions

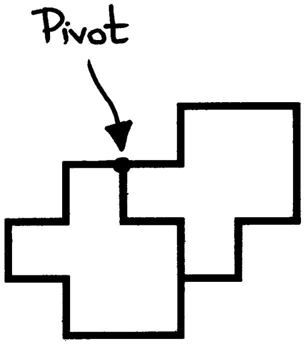
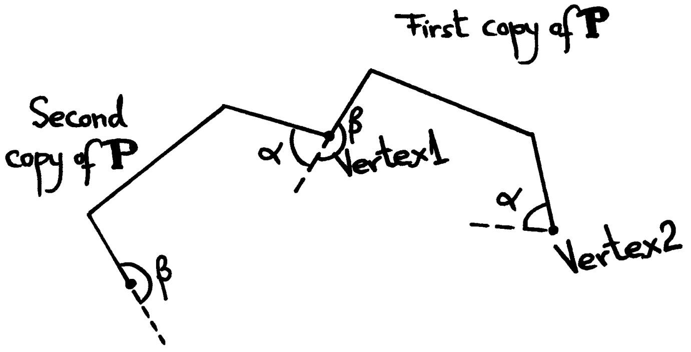


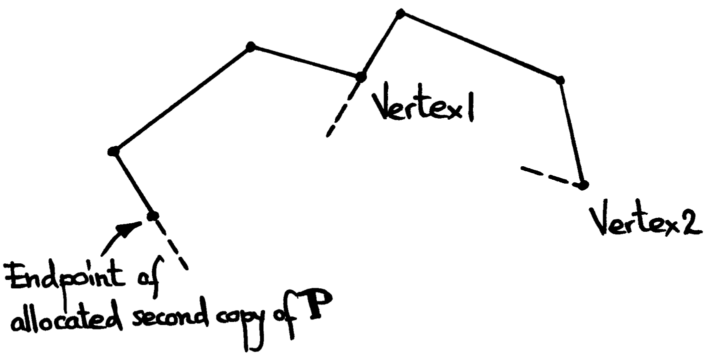
Fig. 10. A pivot vertex constituting a trivial **PP** segment

a second copy is allocated by allocating the length of the first segment on its counterclockwise side (see Figure 11b). The algorithm described above is repeated, with one change the counterclockwise end of the second copy has an 180^0 vertex (if the second copy ends in a vertex one can skip this part). Naturally one has to repeat the algorithm above for every initial selection of two vertices by allocating the length of the initial boundary segment on the clockwise side of the first selection.

Having completed the first part of the algorithm where a boundary segment has qualified to be the **PP** part of the required boundary syntax $\mathbf{QJPP\bar{J}Q}$, we can continue checking the rest of the syntax. Following the boundary on both ends of the **PP** segment we start assembling the **J** and the $\bar{\mathbf{J}}$ segments. We match the length of the first edge on both ends, and make sure the next turn angles are negatives of each other. We continue until one of the conditions is broken. If the angle condition was met and one of the edge segments is shorter, we stop the $\mathbf{J\bar{J}}$ search in the vertex of the short edge, and the middle of the longer edge. If the



(a)



(b)

Fig. 11. The case where one end of **P** is inside an edge

edge length condition has been met, but not the turn angle condition, we stop at both vertices. Notice that this stage cannot fail (we start with a successful vertex condition and may well stop on the first edge).

The boundary segments we managed to traverse are the candidates for the **J** and $\bar{\mathbf{J}}$ segments. The remaining boundary segment should now correspond to the **QQ** segment. The latter is verified by cutting it at the midpoint, and checking exactly as we did the **PP** segment (including the internal angle condition, see Figure 9).

It should be noted that a shape may well pass all the syntax conditions, but not have a crazy cut. To verify the validity of the crazy cut one has to make sure the boundary segment made of $\mathbf{QJP}\bar{\mathbf{J}}$ constitutes a valid closed shape. To do that we have to concatenate all the boundary segments (including the turn angles in the ends of the boundary segments) and test whether the the resulting boundary is a simply closed contour.

To summarize the complexity of the algorithm: The first selection of points is $O(L^2)$. For each selection the traversal of all the other conditions: Checking the **PP** segment, the **J** and $\bar{\mathbf{J}}$ segments and the **QQ** segment amounts to an $O(L)$ processing time. Hence the total algorithm complexity is $O(L^3)$. A successful completion of the syntax search is so rare (in our implementations we found only one false alarm) that the additional $O(L \log L)$ complexity of the final verification does not increase the total complexity.

4.2 Algorithm for Crazy Cuts for Pixelized Shapes

Pixelized shapes are shapes made out of a collection of square pixels. Formally, they are a special case of polylines discussed above. For simplicity we can optionally modify their description such that each edge is exactly one unit length long, and turn angles are either $+90^0 - 90^0$ or 0^0 the latter being the main difference from the standard polyline description. All above depicted syntax checking algorithms remain essentially the same. Obviously, here edge length comparison becomes trivially simple, and there are no edge length mismatch cases such as

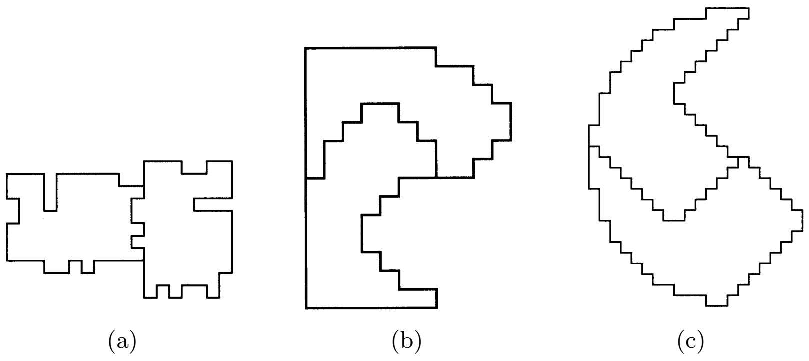

(a) (b) (c)

Fig. 12. Some interesting pixelized crazy cuts

in Figure 11. Additionally, if we require that both the full shape and each of the cuts are pixelized, we inevitably miss some crazy cuts (where the cuts are not pixelized). Figure 12 depicts some interesting pixelized crazy cuts that were combined and then detected via our algorithms.

5 Concluding Remarks

We have seen that analysis of the self-docking problem for planar shapes readily leads to a very nice characterization of shapes that can be split into two identical shapes and to efficient ways to solve crazy cut puzzles. We would have been elated to be the first to provide a mathematical discussion on an algorithm for crazy-cut problems, however a thorough search of the web revealed a paper of Kimmo Eriksson dealing with 'crazy-cut' in 1996 [1]. His approach, subsequently criticized and elaborated upon by G. Rote and his collaborators [2], [3], [4], relies on checking whether two parallel tracings, one along the border of the shape (master) and a corresponding curve (slave) that 'follows the master' tracing in a Euclidean invariant way yield a solution to the problem, when initiated at two arbitrarily selected border points. Although both Eriksson and Rote eventually obtain efficient algorithms to solve crazy cut puzzles, their approach is considerably complicated by the fact that the very simple syntax made obvious by analyzing self-docking was lacking in their work. It is very nice however to realize that both approaches eventually yield solutions to the geometric problem at hand via string analysis, the string being an Euclidean-invariant-signature-based description of the borders of the object. This way of encoding shape is very important and basic in the shape analysis field, where automatic curve matching [5], computerized jigsaw puzzle solutions [6], shape docking [7], invariant shape processing [8], and skew-symmetry detection issues [9] were dealt with via such an approach.

Note that from the work on viewpoint-invariant planar shape analysis via variations of projective, affine and similarity invariant boundary signatures, see e.g. [8,9], we see that invoking generalized invariant signatures we could readily solve crazy cut puzzles even for shapes distorted by such viewing transformations.

Acknowledgement

We are grateful to Tomer Shaked for challenging us with several hard crazy-cut puzzles, thereby initiating the work reported in this paper.

References

1. Eriksson, K.: Splitting a polygon into two congruent pieces. The American Mathematical Monthly 103(5), 393–400 (1996)
2. Rote, G.: Some thoughts about decomposition of a polygon into two congruent pieces (1997) (unpublished draft), http://page.mi.fu-berlin.de/rote/Papers/postscript/Decomposition+of+a++polytope+into+two+congruent+pieces.ps

3. El-Khechen, D., Fevens, T., Iacono, J., Rote, G.: Partitioning a polygon into two congruent pieces. In: Kyoto International Conference on Computational Geometry and Graph Theory, KyotoCGGT2007, Kyoto, Japan, June 11-15 (2007)
4. El-Khechen, D., Fevens, T., Iacono, J., Rote, G.: Partitioning a polygon into two mirror congruent pieces. In: 20th Canadian Conference on Computational Geometry, CCCG 2008, Montréal, Québec, August 13-15 (2008)
5. Wolfson, H.J.: On curve matching. IEEE Transactions on PAMI 21(5), 483–489 (1990)
6. Wolfson, H., Schonberg, E., Kalvin, A., Lamdan, Y.: Solving jigsaw puzzles using computer vision. Ann. Oper. Res. 12, 51–64 (1988)
7. Imiya, A., Kudo, S.: Docking of polygons using boundary descriptor. In: Petkov, N., Westenberg, M.A. (eds.) CAIP 2003. LNCS, vol. 2756, pp. 25–32. Springer, Heidelberg (2003)
8. Bruckstein, A.M., Shaked, D.: On projective invariant smoothing and curve evolutions. Journal of Mathematical Imaging and Vision 7, 225–240 (1997)
9. Bruckstein, A.M., Shaked, D.: Skew symmetry detection via invariant signatures. Pattern Recognition 31(2), 181–192 (1998)

Piecewise Rational Manifold Surfaces with Sharp Features

G. Della Vecchia and B. Jüttler

Johannes Kepler University, Institute of Applied Geometry,
Linz, Austria
giovanni.dellavecchia@jku.at, bert.juettler@jku.at

Abstract. We present a construction of a piecewise rational free-form surface of arbitrary topological genus which may contain sharp features: creases, corners or cusps. The surface is automatically generated from a given closed triangular mesh. Some of the edges are tagged as sharp ones, defining the features on the surface. The surface is $\mathcal{C}^s$ smooth, for an arbitrary value of s, except for the sharp features defined by the user. Our method is based on the manifold construction and follows the blending approach.

Keywords: Manifold surface, sharp features, smooth piecewise rational free-form surface, arbitrary topological genus, geometric continuity.

1 Introduction

Several approaches for free-form surface modeling exist. The main approaches include geometrically continuous spline surfaces, subdivision surfaces and manifold-based constructions.

The constructions of geometrically continuous spline surfaces generate collections of piecewise polynomial or rational patches, which are joined together with various degrees of smoothness, e.g., [13,14,15,17,18,20,21]. The available techniques include the use of singular parameterizations and of multisided generalizations of Bézier surfaces.

Subdivision surfaces have been developed into a very valuable tool for free-form surface modeling, see e.g. [19,22]. Their theory - in particular the shape analysis around extraordinary vertices - has made substantial progress during the last years. However, these surfaces do not possess closed-form parametric representation in the vicinity of extraordinary points and it is difficult to achieve higher orders of smoothness.

Another class of methods is based on the notion of a differentiable manifold from differential geometry. This notion has been developed into a construction for smooth surfaces of arbitrary topological genus which are covered by a collection of local parameterizations. The first paper in this direction was authored by Grimm and Hughes [6], and this interesting approach was developed further in a number of publications [3,4,7,8,16,24]. In particular [7,8] present a construction of an affine structure for manifold spline surfaces. This construction requires only one chart for the evaluation at one point, but it has the disadvantage that in general holes in the mesh are required to deal with models of arbitrary topology.

In some cases, the possibility to define and to describe sharp features on the surface is required in order to obtain adequate results. Therefore, the problem of modeling piecewise smooth free-form surfaces which possess a number of sharp features is of some interest.

So far, most of the available techniques for solving this task are either based on subdivision surfaces or they work directly with triangular meshes, see e.g. [5,12]. The two papers [2,11] describe methods to define the path of feature curves at arbitrary locations on the surface. Various other approaches, mostly for surfaces represented by triangular meshes, have been proposed [1,10,23].

In this paper we present a manifold-based free-form surface construction which can generate sharp features. More precisely, given a closed triangular mesh of arbitrary topological genus, we generate a free-form surface. Some of edges of the mesh can be tagged as "sharp" edges, defining the feature curves on the free-form surface. The surface can achieve any desired order of smoothness, except of course along the feature curves, where it is just continuous. The construction is based on an extension of our recent paper [4], where we constructed a manifold surface by blending together circular charts.

The method presented in this paper preserves the manifold structure (i.e. the parameterized atlas associated with the mesh) and the surface is evaluated following the blending approach. The definition of sharp features in our construction is possible because of the high flexibility which is available in the definition of the transition functions, which are generated by the method of subchart parameterization. This may be an advantage compared to other manifold constructions which are based on more uniform transition functions.

The remainder of the this paper consists of four parts. Section 2 summarizes the method described in [4], which is the starting point for the construction presented in this paper. Section 3 extends the previous results to manifold surfaces with sharp features. The fourth section presents several examples. Finally we conclude this paper.

2 A Construction of Rational Manifold Surfaces

In order to make this paper self-contained we recall a construction of a rational spline manifold which was presented in [4]. This construction proceeds in three steps. We (1) define charts and subcharts, (2) generate subchart parameterizations, transition functions and the domain, and (3) obtain the manifold surface as a blending-based embedding of the domain.

2.1 Charts and Subcharts

We consider an oriented triangular mesh M in $\mathbb{R}^3$ with m_V vertices, where the ith vertex possesses the valency $v(i)$.

Let $[n(i,j)]_{j=0}^{v(i)-1}$ be the list of neighbours of the ith vertex in counterclockwise order with respect to the orientation of M. The second argument of n will be used modulo $v(i)$, i.e., $n(i,j) = n(i, j + k\,v(i))$ for all $k \in \mathbb{Z}$. Let

$$V = \{i : \ i = 1, \ldots, m_V\}, \tag{1}$$
$$E = \{\{i, n(i,r)\} : \ i \in V, \ r = 1, \ldots, v(i)\}, \text{ and} \tag{2}$$
$$F = \{\,\{i, n(i,r), n(i,r+1)\} : \ i = 1, \ldots, m_V, \ r = 1, \ldots, v(i)\}. \tag{3}$$

be the *sets* of *vertex, edge* and *face indices*. Note that we use set-valued edge and face indices, hence the order within these indices is not relevant.

For each vertex index $i \in V$, we define an associated chart $C^i \subset \mathbb{R}^2 \times \{i\}$,

$$C^i = \{(x, y, i) : \ x^2 + y^2 \leq 1\}, \tag{4}$$

which is essentially a circular unit disk, centered at the origin. The third coordinate i has been added in order to obtain mutually disjoint charts,

$$i \neq j \Rightarrow C^i \cap C^j = \emptyset. \tag{5}$$

The chart C^i is subdivided into edge and face subcharts and an innermost region, as follows.

– For each edge $\{i, n(i,r)\}$ of the mesh which starts or ends at the ith vertex, the chart C^i possesses an *edge subchart* $C^i_{n(i,r)}$.
– For each face $\{i, n(i,r), n(i,r+1)\}$ which shares the ith vertex, the chart C^i has a *face subchart* $C^i_{n(i,r),n(i,r+1)}$, where $r = 1, \ldots, v(i)$.
– The remaining or *innermost part* of C^i is

$$\hat{C}^i \ = \overline{C^i \setminus (\bigcup_{r=1,..,v(i)} C^i_{n(i,r)} \ \cup \bigcup_{r=1,..,v(i)} C^i_{n(i,r),\, n(i,r+1)})}. \tag{6}$$

The generic layout of charts and subcharts is shown in Figure 1, see also [4]. The subcharts are arranged in counterclockwise order along the boundaries of the charts.

The face subcharts are triangular regions with two straight and one circular boundary. They correspond to the overlap of the local parameterizations which are defined by three charts.

The edge subcharts are quadrangular regions with one circular, two straight and one free-form boundary (which is shared with the innermost part of the chart). The *overlapping region* of the chart C^i with another chart $C^{n(i,r)}$, where $(i, n(i,r))$ is an edge of the mesh, is the union of two face subcharts and one edge subchart,

$$O^i_{n(i,r)} = C^i_{n(i,r-1),n(i,r)} \cup C^i_{n(i,r)} \cup C^i_{n(i,r),n(i,r+1)}. \tag{7}$$

It corresponds to the overlap of the local parameterizations which are defined by two charts. This region has the shape of a biangle with two $\mathcal{C}^s$ smooth boundary curves meeting in two vertices, where s is the order of smoothness of the manifold surface.

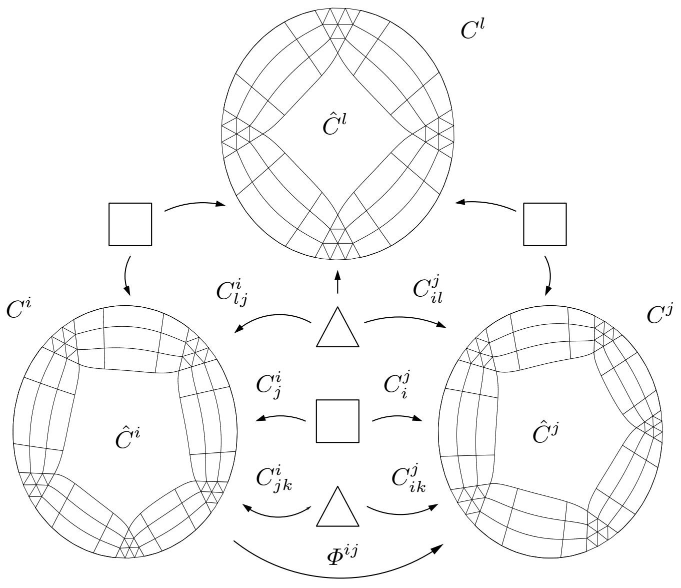


Fig. 1. Charts, subcharts and subchart parameterizations

2.2 Transition Functions and Domain

We define the transition functions with the help of *subchart parameterizations*, whose domains are the unit square $\square$ or the standard triangle $\triangle$,

$$\square = [0,1]^2 \text{ and } \triangle = \{(u,v,w) :\ u \geq 0,\ v \geq 0,\ w \geq 0,\ u+v+w=1\}. \quad (8)$$

More precisely, we define the following parameterizations.

– For each edge $e = \{i,j\} \in E$ of the triangular mesh we define two *edge subchart parameterizations*

$$\phi^i_j : \square \to C^i_j \text{ and } \phi^j_i : \square \to C^j_i. \quad (9)$$

– For each face $f = \{i,j,k\} \in F$ of the triangular mesh, we define three *face subchart parameterizations*,

$$\phi^i_{jk} : \triangle \to C^i_{jk},\ \ \phi^j_{ki} : \triangle \to C^j_{ki} \text{ and } \phi^k_{ij} : \triangle \to C^k_{ij}. \quad (10)$$

The order of the lower indices is not relevant, $\phi^i_{jk} = \phi^i_{kj}$.

These mappings are assumed to be $\mathcal{C}^s$ smooth, surjective, and orientation preserving. Moreover, they are assumed to be compatible in the following sense:

– For each vertical edge of the unit square, there is exactly one of the two edge subchart parameterizations (9) which maps it into the boundary of the chart.

– For each edge of the standard triangle, there is exactly one of the three face subchart parameterizations (10) which maps it into the boundary of the chart.

Now we define the *transition function* (or coordinate transformation) between any pair of charts C^i and C^j with $\{i,j\} \in E$. Let $\{i,j,k\}, \{i,j,l\} \in F$ such that

$$\exists\, r : k = n(i, r-1),\ j = n(i,r),\ l = n(i, r+1), \tag{11}$$

$$\exists\, s : l = n(j, s-1),\ i = n(j,s),\ k = n(j, s+1). \tag{12}$$

The transition function Φ^{ij} maps the overlapping region $O^i_j \subset C^i$ into the overlapping region $O^j_i \subset C^j$,

$$\Phi^{ij} \,:\, O^i_j \to O^j_i \,:\, u \mapsto \begin{cases} (\phi^j_{li} \circ (\phi^i_{lj})^{-1})(u) & \text{if} \quad u \in C^i_{lj} \\ (\phi^j_{i} \circ (\phi^i_{j})^{-1})(u) & \text{if} \quad u \in C^i_{j} \\ (\phi^j_{ik} \circ (\phi^i_{jk})^{-1})(u) & \text{if} \quad u \in C^i_{jk} \end{cases} . \tag{13}$$

These transition functions obey the cocyle condition,

$$(\Phi^{jk} \circ \Phi^{ij})(u) = \Phi^{ik}(u) \quad \text{if} \quad u \in C^i_{jk}, \tag{14}$$

since the overlapping region of three charts is parameterized with respect to the common domain $\triangle$. However, the transition functions are not automatically guaranteed to be $\mathcal{C}^s$ smooth, where s is the required degree of smoothness of the surface. This is needed in order to obtain a manifold surface which possesses this smoothness. The smoothness of the transition functions has to be ensured by the construction of the subchart parameterizations.

For each chart C^i, associated with the vertex i, we choose the face subchart parameterizations ϕ^i_{jk} as planar rational Bézier triangles of degree two. The layout of these Bézier triangles along the boundary of the chart is automatically determined by projecting the neighboring vertices into the tangent plane of the vertex i.

The edge subchart parameterizations ϕ^i_j are then constructed as rational tensor product patches of degree $(4, 4s+2)$. Using a construction which is based on Möbius transformations and blending we are able to achieve $\mathcal{C}^s$ smoothness of the transition functions.

See [4] for more details about the parameterization of face and edge subcharts.

The transition functions Φ^{ij} define an equivalence relation on the union of the charts $\bigcup_{i \in V} C^i$. More precisely, two points $u \in C^i$ and $v \in C^j$ are considered as equivalent ($\sim$), if the transition function Φ^{ij} maps u into v,

$$u \sim v \iff \exists\, \{i,j\} \in E :\ \Phi^{ij}(u) = v. \tag{15}$$

The *domain* of the manifold surface with charts C^i and transition functions Φ^{ij} is then obtained by forming the equivalence classes of this relation,

$$\Omega = (\bigcup_{i \in V} C^i) / \sim . \tag{16}$$

The elements of Ω are *sets* containing one, two or three points of different charts.

The face $f = \{i, j, k\} \in F$ of the mesh M corresponds to a subset of Ω which consists of sets with three elements,

$$\Omega^{ijk} = \{\{\phi^i_{jk}(t), \phi^j_{ik}(t), \phi^k_{ij}(t)\} : \ t \in \triangle\} \subset \Omega. \tag{17}$$

The edge $e = \{i, j\} \in E$ of the mesh M corresponds to a subset of Ω which consists of sets with two elements,

$$\Omega^{ij} = \{\{\phi^i_j(t), \phi^j_i(t)\} : \ t \in \square\} \subset \Omega. \tag{18}$$

Finally, the ith vertex of the mesh corresponds to a subset which consists of sets with only one element,

$$\Omega^i = \{\{u\} : \ u \in \hat{C}^i\} \subset \Omega \tag{19}$$

where $\hat{C}^i$ is the innermost part of the chart C^i.

2.3 Manifold Surface by Blending

The manifold spline surface is generated as an embedding of the domain. For each chart C^i we define a *geometry function* $\mathbf{g}^i$ and an *influence function* β^i.

– The geometry function

$$\mathbf{g}^i : \ C^i \to \mathbb{R}^3 : \ (x, y, i) \mapsto \mathbf{g}^i(x, y) \tag{20}$$

is a vector-valued bivariate quadratic polynomial, which is automatically generated by approximating the neighbours of the ith vertex of the given mesh.

– The influence function β^i is a suitable power of the circle equation,

$$\beta^i : \ C^i \to \mathbb{R} : \ (x, y, i) \mapsto (1 - x^2 - y^2)^{s+1} \tag{21}$$

where s is the desired order of smoothness of the surface. This defines a bubble function whose first s derivatives vanish along the boundary of C^i.

Finally we define the embedding of the domain Ω by blending together the contributions of all charts,

$$\mathbf{p} : \Omega \to \mathbb{R}^3 : u \mapsto \frac{\displaystyle\sum_{u \cap C^i \neq \emptyset, \ i \in V} \beta^i(u \cap C^i)\, \mathbf{g}^i(u \cap C^i)}{\displaystyle\sum_{u \cap C^i \neq \emptyset, \ i \in V} \beta^i(u \cap C^i)} \tag{22}$$

Depending on the cardinality of u, the point $\mathbf{p}(u)$ is obtained by blending together three, two or one geometry functions.

For all $u \in \Omega^{ijk}$, which correspond to the face $\{i, j, k\}$ of the given mesh M, the set of corresponding points on the manifold surface can be parameterized as a rational triangular patch with the domain $\triangle$. For all $u \in \Omega^{ij}$, which correspond to the edge $\{i, j\}$ of M, the set of corresponding points on the manifold surface can be parameterized as a rational quadrangular patch with domain $\square$. Finally, the innermost parts of the charts can be covered by triangular or quadrangular patches, which again correspond to rational patches on the manifold surface. Consequently we obtain a C^s smooth manifold surface, which can be represented as a collection of rational quadrangular and triangular surface patches.

Remark 1. Since the degree of the quadrangular and triangular surface patches is relatively high, it is not recommended to represent them in closed form as Bézier patches. Nevertheless, the surfaces are expressed in explicit form and it is simple to use tools such as automatic differentiation.

3 Extension to Sharp Features

We extend the presented framework in order to model objects with sharp features. Sharp features can be classified as *darts*, *creases* and *corners* (see [9,12]). Smooth curves along which the surface presents tangent discontinuity are *creases*. *Corners* are points where three or more creases meet; a *dart* is an interior point of the surface where a crease starts or ends. We adapt our construction to all three types.

3.1 Sharp Edges and *k*-Vertices

For any given mesh M, the construction presented in the previous section produces a C^s smooth surface (provided that no singularities occur) from a given mesh. In certain situations, however, the given mesh is not suitable for a smooth surface generation, and the method generates inadequate results. For instance, this is the case if large dihedral angles between the faces of the triangles are present.

We assume that some of the edges of the mesh are tagged as *sharp edges*. For instance, one may simply choose all edges where the angle between the incident faces exceeds a certain threshold.

Consider the ith vertex with valency $v(i)$. Depending on the number k of sharp edges which share this vertex, this vertex is classified as a *k–vertex* of the mesh.

- $k = 0$: A 0–vertex corresponds to a smooth region on the surface.
- $k = 1$: A 1–vertex corresponds to a dart on the surface, i.e., to a point where a crease starts or ends.
- $k = 2$: A 2–vertex corresponds to a segment of a crease on the surface
- $k \geq 3$: Such a vertex corresponds to a corner of the surface.

3.2 k–Charts and Their Subcharts

If the ith vertex is a k–vertex, then the chart associated with it will be called a k–chart. We describe how to adapt the constructions of the subcharts and of the subchart parameterizations for $k \neq 0$. The case $k = 0$ is dealt with as in [4].

The construction of the k–charts and of its subcharts consists of three steps (cf. Fig. 2).

- *Step 1: Face subcharts and face subchart parameterizations.* The face subcharts $C^i_{n(i,r),n(i,r+1)}$ are constructed using the method described in [4]. We choose the layout of the face subcharts based on the geometry of the mesh in the vicinity of the ith vertex. More precisely, we consider the projection of its neighborhood into the estimated tangent plane and use it to choose certain geometric parameters controlling the layout of edge and face subcharts. The face subcharts are then parameterized by quadratic rational Bézier triangles with one circular and two straight line boundaries.
- *Step 2: Central point $\mathbf{c}^i$ and edge lines.* We choose a central point $\mathbf{c}^i \in C^i$, where the edge lines will meet. For each sharp edge $\{i, n(i,r)\}$ emanating from the ith vertex, we choose an edge line $E^i_{n(i,r)}$ which connects $\mathbf{c}^i$ with a

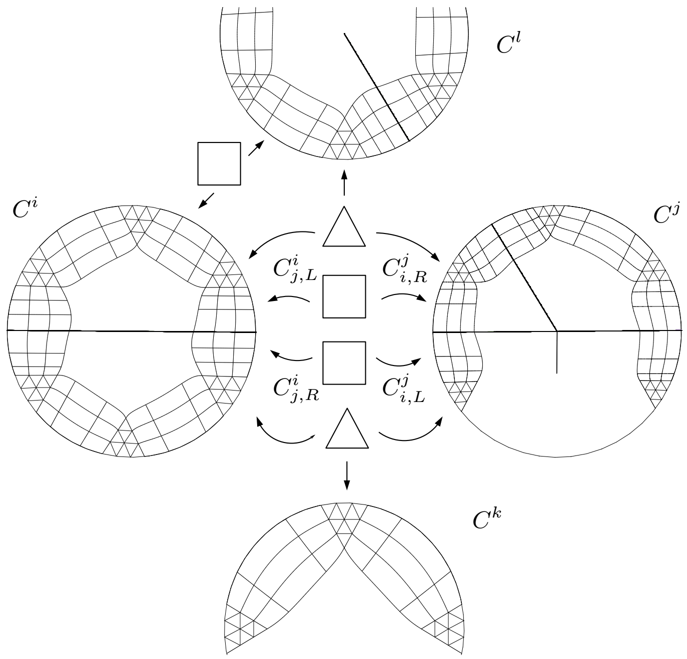


Fig. 2. Definition of k-charts and subcharts for different values of k. The chart C^i (center, left) is a 2-chart.

point on the boundary ∂C^i of the chart, where this point is located between the face subcharts $C^i_{n(i,r-1),n(i,r)}$ and $C^i_{n(i,r),n(i,r+1)}$.

In our implementation, the central point is simply the center of the circle, and the edge lines are straight line segments. An exception is made for $k = 2$. In that case, we get only two edge lines, which are supposed to meet smoothly at the central point. In that case we choose both edge lines as a straight line segment connecting two points on the boundary ∂C^i, and the central point as its midpoint.

– *Step 3: Edge semi-subcharts.* For each sharp edge $\{i, n(i,r)\}$, the edge subchart $C^i_{n(i,r)}$ is subdivided into a *left* and a *right semi–subchart* $C^i_{n(i,r),L}$ and $C^i_{n(i,r),R}$ with associated parameterizations. The left (resp. the right) edge semi–subchart shares one boundary arc with $C^i_{n(i,r-1),n(i,r)}$ (resp. with $C^i_{n(i,r),n(i,r+1)}$) and one with the innermost part $\hat{C}^i$. The remaining two boundary arcs are segments of the edge line $E^i_{n(i,r)}$ and of the chart boundary ∂C^i. The boundary arcs of the two semi-subcharts cover approximately one third of the edge line E^i_j.

The remaining edge subcharts (which correspond to non–sharp edges) are generated as in the case $k = 0$, see [4].

The parameterization of the two semi-subcharts is described in the next section.

3.3 Parameterization of Semi-subcharts

The edge semi-subcharts are parameterized by mappings whose domain is the unit square $\square = [0,1]^2$. These two mappings will be called left and right edge semi-subchart parameterization

$$\phi^i_{j,L} : \square \to C^i_{j,L} \quad \text{and} \quad \phi^i_{j,R} : \square \to C^i_{j,R}. \tag{23}$$

The transition functions between two charts are defined as in the smooth case, see (13). These transition functions need to be C^s smooth everywhere except along the edge lines, where s is the desired smoothness of the manifold surface. This is obtained by ensuring that the two edge semi-subcharts satisfy the following conditions.

– They have a $\mathcal{C}^s$ smooth joint with the neighboring face subcharts which are reparameterized as tensor-product patches.
– They have a $\mathcal{C}^0$ joint along the edge line E^i_j.
– Their outer boundary is contained in the boundary ∂C^i.

The construction of the edge semi-subchart parameterizations is obtained with the help of Möbius transformations, similar to the case of edge subchart parameterizations (cf. [4]). We describe in more detail the construction of a right edge semi–subchart, see also Fig. 3. The other case is dealt with analogously.

Let μ be a Möbius transformation that maps the unit circle into the real axis, which is computed in the same way as in [4]; let ρ be the reparameterization

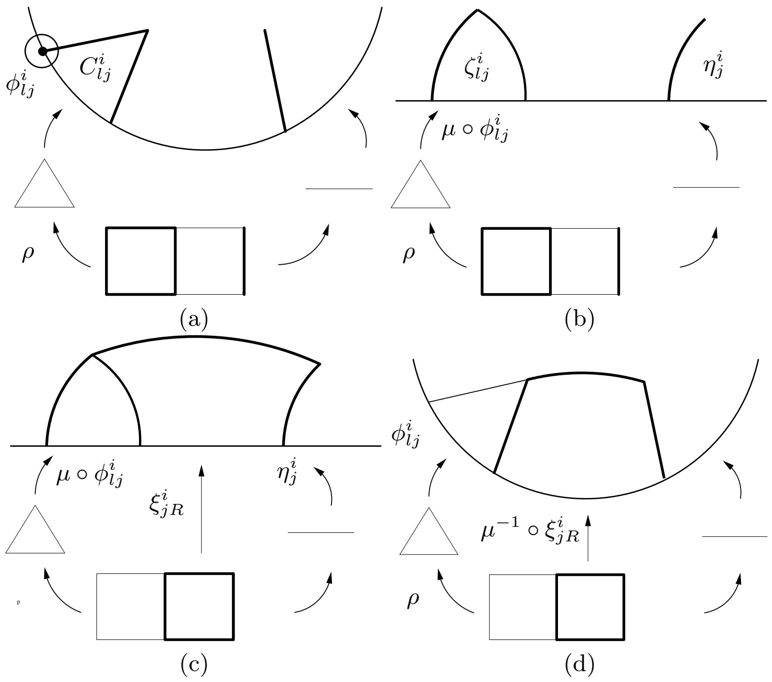


Fig. 3. Parameterization of a right edge semi-subchart. (a) The neighbouring face subchart is represented as a degenerate quadrangular patch, (b) applying the Möbius transformation, (c) construction of a tensor-product patch, (d) the inverse Möbius transformation gives the desired parameterization.

$$\rho : \square \to \triangle : (r, s) \mapsto (r, (1-r)s, (1-r)(1-s)) \tag{24}$$

that represents a triangular patch as a degenerate tensor product patch, with the edge $r = 1$ collapsing into a singular point.

1. The face subchart parameterization ϕ_{lj}^i is a quadratic Bézier triangle. By composing it with ρ we obtain a degenerate tensor-product patch of degree $(1, 2)$ (see [4]). Finally it is composed with μ and this gives the rational tensor-product patch

$$\zeta_{lj}^i = \mu \circ \phi_{lj}^i \circ \rho \tag{25}$$

of degree $(2, 4)$ which parameterizes the image $\mu(C_{lj}^i)$. We consider a linear parameterization of the segment of the edge line E_j^i, where the two semi-subcharts are to be joined. By composing this parameterization with μ we obtain the curve η_j^i.
2. We create a tensor-product patch ξ_{jR}^i which possesses a $\mathcal{C}^s$ smooth joint with ζ_{lj}^i and $\mathcal{C}^0$ joint with η_j^i. This patch can be chosen as a rational tensor–product patch of degree $(2, s+1)$.

3. We apply the inverse Möbius transformation in order to get the desired edge semi-subchart parameterization

$$\phi^i_{j,R} = \mu^{-1} \circ \xi^i_{j,R}. \tag{26}$$

This gives a rational tensor product patch of degree $(4, 2s + 2)$.

3.4 Feature Lines and Geometry Functions

Finally we have to generate geometry functions which take the edge lines into account. Let the chart C^i be a k-chart with k edge lines E^i_j. If $k = 0$, then the geometry function is computed as in the smooth case. Otherwise we assume that – for each edge line – a feature curve $f^i_j : E^i_j \to \mathbb{R}^3$ is given. All feature curves share the common point $f^i_j(\mathbf{c}^i)$. Moreover, if $k = 2$, then the two feature curves have a $\mathcal{C}^s$ smooth joint.

The feature curves can either be specified by the user, or they can be automatically generated by approximating the sharp edges of the mesh by smooth curves.

The geometry function $\mathbf{g}^i : C^i \to \mathbb{R}^3$ of the k-chart is now chosen as a continuous piecewise polynomial function with k pieces which respects the feature curves, i.e.,

$$\mathbf{g}^i\big|_{E^i_j} = f^i_j \tag{27}$$

holds for all sharp edges $\{i, j\}$. It is C^s smooth, except for the edge lines.

For example, each geometry function $\mathbf{g}^i$ can be defined as a piecewise function consisting of k Bézier triangles, corresponding to the k feature lines f^i_j. These Bézier triangles are defined such that the control points along the boundaries,

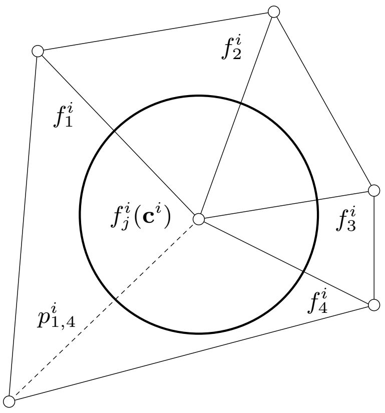


Fig. 4. Example of geometry function in the planar case as a piecewise function consisting of Bézier triangles. Since the angle between f^i_1 and f^i_4 is greater than π, the phantom edge $p^i_{1,4}$ is introduced, defining two Bézier triangles joined with $\mathcal{C}^d$ continuity.

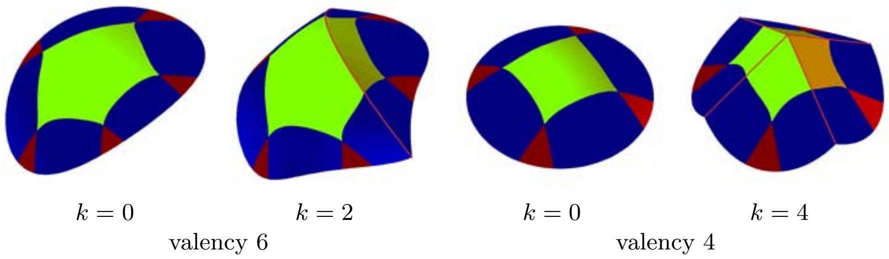


Fig. 5. Geometry functions for different valencies and for different values of k. The different colors visualize the different subcharts.

i.e. the feature lines, coincide. In case the angle between two feature lines f^i_j and f^i_k exceeds π, one can introduce a phantom edge p^i_{jk} between these feature lines and define the piece of $\mathbf{g}^i$ between f^i_j and f^i_k as two Bézier triangles joined with a certain $\mathcal{C}^d$ continuity along p^i_{jk}. Figure 4 shows an example of a piecewise defined geometry function in the planar case.

In our implementation, if $k \geq 3$, then the feature curves are segments and the geometry are piecewise linear. If $k = 2$, then we choose cubic feature curves and piecewise cubic geometry functions. In all other cases we choose the geometry functions as in [4].

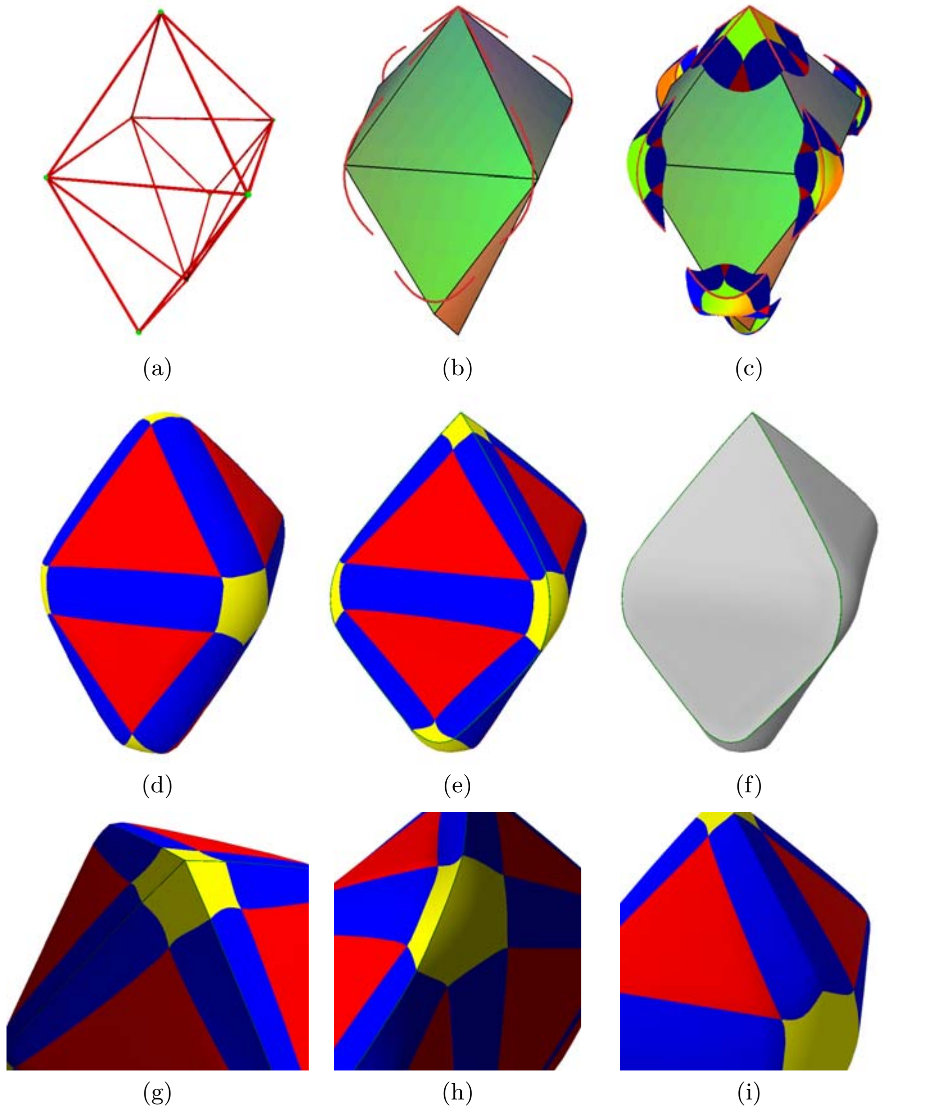
Figure 5 shows the geometry functions for k–charts with different valencies.

4 Examples

We present three examples for surfaces which have been generated via the proposed method. All surfaces can be represented as a collection of rational surface patches which form C^2 smooth manifold surfaces, except for the sharp features. The construction was implemented in Maple, and the surfaces were visualized using PovRay, where each quadrangular resp. triangular surface patch was rendered using 50 resp. 25 triangles.

Example 1 (see Fig. 6). The triangular mesh consists of 8 vertices and 12 faces. We defined two loops of edge curves which meet in the top vertex. Consequently, the mesh defines six 2-charts, one 4–chart and one 0–chart (the bottom one). The figure shows the results of the construction with and without sharp features. In addition, we also show the result which is obtained by defining a 1–chart, which leads to a dart feature on the surface.

Example 2 (see Fig. 7). This surface has been generated from a star–shaped polyhedron, consisting of 18 vertices and 32 faces. We create a crease feature by defining a closed curve of four sharp edges, which leads to four 2–charts and 14 0–charts.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 6. Example 1: Manifold surface generated from a mesh with 8 vertices and two loops of sharp edges. The mesh (a), feature curves (b), geometry functions (c), a smooth model obtained with smooth geometry functions (d), the model with sharp edges (e,f), and details of the features (g,h) are shown. The last plot (i) shows the effect of a 1-chart which generates a dart feature on the surface. The different colors in (d,e,g,h) visualize the contributions of the different subcharts.

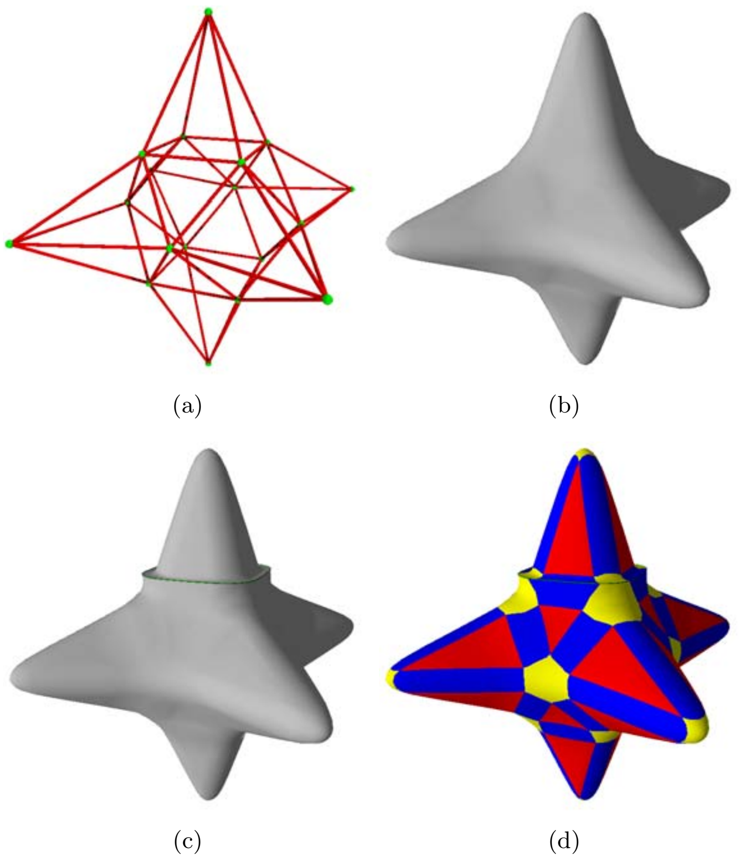


Fig. 7. Example 2: A manifold spline surface (b) obtained from a star–shaped polyhedron (a) and a surface which was obtained defining a loop of sharp edges (c,d)

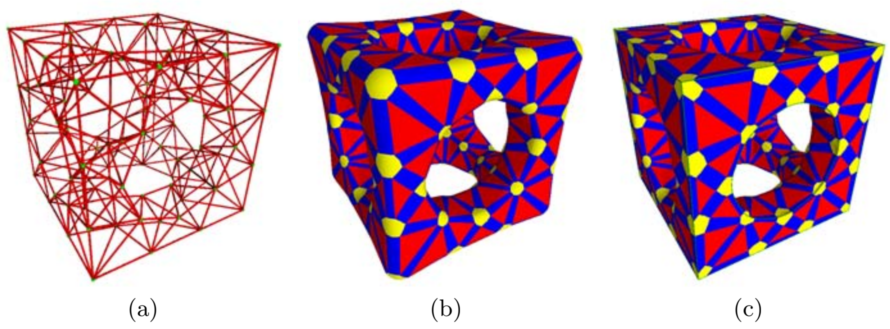


Fig. 8. Example 3: A smooth manifold surface (b) obtained from a mesh of a hollow cube (a) and a surface with sharp features along the outer edges and around one of the circles (c)

Example 3 (see Fig. 8). The final example has been obtained from a mesh describing a hollow cube with 112 vertices and 240 faces. The sharp features correspond to the edges of the cube, which define 12 creases and 8 corners, and to the circles in the faces. Here we modified only one of these circles.

5 Conclusions

We presented a new construction of piecewise rational free-form surfaces with the possibility of defining sharp features on it, based on the manifold construction already presented in [4]. Starting from a triangular mesh, where some edges are tagged as sharp edges, the algorithm generates automatically a surface, which can achieve any order of smoothness, except for the sharp features.

The definition of sharp features on the surface does not modify the manifold representation and the transition functions between overlapping charts, which keep the $\mathcal{C}^s$ smoothness everywhere except along the feature lines. The surface can be represented as a collection of quadrangular and triangular rational patches.

Future research is required concerning the optimal choice of the geometry functions, in order to obtain a better visual fairness of the manifold surfaces. Also, the analysis of manifold constructions which generate spline surfaces of lower degree is of potential interest.

Acknowledgment. This research has been supported by the Austrian Science Fund (FWF) in the FSP framework S092 'Industrial geometry', subproject 2.

References

1. Attene, M., Falcidieno, B., Rossignac, J., Spagnuolo, M.: Sharpen and bend: recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. IEEE Transactions on Visualization and Computer Graphics 11(2), 181–192 (2005)
2. Biermann, H., Martin, I., Zorin, D., Bernardini, F.: Sharp features on multiresolution subdivision surfaces. In: Proc. Pacific Graphics, pp. 140–149 (2001)
3. Cotrina-Navau, J., Pla-Garcia, N.: Modeling surfaces from meshes of arbitrary topology. Computer Aided Geometric Design 17(7), 643–671 (2000)
4. Della Vecchia, G., Jüttler, B., Kim, M.-S.: A construction of rational manifold surfaces of arbitrary topology and smoothness from triangular meshes. Computer Aided Geometric Design 25(9), 801–815 (2008)
5. DeRose, T., Kass, M., Truong, T.: Subdivision surfaces in character animation. In: Proc. SIGGRAPH, pp. 85–94. ACM, New York (1998)
6. Grimm, C., Hughes, J.: Modeling surfaces of arbitrary topology using manifolds. In: Proc. Siggraph, pp. 359–368. ACM Press, New York (1995)
7. Gu, X., He, Y., Jin, M., Luo, F., Qin, H., Yau, S.-T.: Manifold splines with single extraordinary point. In: Proc. Solid and Physical modeling, pp. 61–72. ACM Press, New York (2007)
8. Gu, X., He, Y., Qin, H.: Manifold splines. In: Proc. Solid and Physical Modeling, pp. 27–38. ACM Press, New York (2005)

9. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W.: Piecewise smooth surface reconstruction. In: Proc. SIGGRAPH, pp. 295–302. ACM, New York (1994)
10. Hubeli, A., Gross, M.: Multiresolution feature extraction from unstructured meshes. In: IEEE Visualization (2001)
11. Khodakovsky, A., Schröder, P.: Fine level feature editing for subdivision surfaces. In: Proc. Shape Modelling Appl., pp. 203–211. ACM, New York (1999)
12. Ling, R., Wang, W., Yan, D.: Fitting sharp features with Loop subdivision surfaces. Comput. Graph. Forum 27(5), 1383–1391 (2008)
13. Loop, C., DeRose, T.D.: A multisided generalization of Bézier surfaces. ACM Trans. Graph. 8(3), 204–234 (1989)
14. Loop, C., DeRose, T.D.: Generalized B-spline surfaces of arbitrary topology. In: Proc. SIGGRAPH, pp. 347–356. ACM Press, New York (1990)
15. Loop, C.: Smooth spline surfaces over irregular meshes. In: Proc. SIGGRAPH, pp. 303–310. ACM Press, New York (1994)
16. Cotrina Navau, J., Pla Garcia, N., Vigo Anglada, M.: A generic approach to free form surface generation. In: Proc. Solid Modeling and Applications, pp. 35–44. ACM Press, New York (2002)
17. Peters, J.: C^2 free–form surfaces of degree $(3, 5)$. Computer Aided Geometric Design 19, 113–126 (2002)
18. Peters, J.: Geometric continuity. In: Farin, G., Hoschek, J., Kim, M.-S. (eds.) Handbook of Computer Aided Geometric Design. Elsevier, Amsterdam (2002)
19. Peters, J., Reif, U.: Subdivision surfaces. Springer, Heidelberg (2008)
20. Prautzsch, H.: Freeform splines. Computer Aided Geometric Design 14(3), 201–206 (1997)
21. Reif, U.: TURBS - topologically unrestricted rational B-splines. Constructive Approximation 14, 57–77 (1998)
22. Sabin, M.A., Cashman, T.J., Augsdorfer, U.H., Dodgson, N.A.: Bounded curvature subdivision without eigenanalysis. In: Martin, R., Sabin, M.A., Winkler, J.R. (eds.) Mathematics of Surfaces 2007. LNCS, vol. 4647, pp. 391–411. Springer, Heidelberg (2007)
23. Yang, H., Jüttler, B.: Evolution of T-spline level sets for meshing non-uniformly sampled and incomplete data. The Visual Computer 24, 435–448 (2008)
24. Ying, L., Zorin, D.: A simple manifold-based construction of surfaces of arbitrary smoothness. Proc. Siggraph, ACM Transactions on Graphics 23(3), 271–275 (2004)

Deriving Box-Spline Subdivision Schemes

N.A. Dodgson[1], U.H. Augsdörfer[1], T.J. Cashman[1], and M.A. Sabin[2]

[1] The Computer Laboratory, University of Cambridge, UK
[2] Numerical Geometry Ltd., UK
{nad10,uha20,tc270,mas33}@cl.cam.ac.uk

Abstract. We describe and demonstrate an arrow notation for deriving box-spline subdivision schemes. We compare it with the z-transform, matrix, and mask convolution methods of deriving the same. We show how the arrow method provides a useful graphical alternative to the three numerical methods. We demonstrate the properties that can be derived easily using the arrow method: mask, stencils, continuity in regular regions, safe extrusion directions. We derive all of the symmetric quadrilateral binary box-spline subdivision schemes with up to eight arrows and all of the symmetric triangular binary box-spline subdivision schemes with up to six arrows. We explain how the arrow notation can be extended to handle ternary schemes. We introduce two new binary dual quadrilateral box-spline schemes and one new $\sqrt{2}$ box-spline scheme. With appropriate extensions to handle extraordinary cases, these could each form the basis for a new subdivision scheme.

1 Introduction

For several years, the Cambridge subdivision research team have used an arrow notation that allows easy derivation of the mask (Section 3.1), stencils (Section 3.2), continuity (Section 3.3), and safe extrusion directions (Section 3.4) of all box-spline subdivision schemes. It also permits enumeration of all possible box-spline schemes, which has allowed us to generate three new schemes which, to the best of our knowledge, have not yet been investigated.

The arrow notation is equivalent to other mechanisms for specifying box-spline schemes but has the advantage that it is a graphical, rather than numerical, notation, allowing easy visualisation of what is going on.

The notation has arrows of appropriate lengths pointing in the principal directions of the scheme. For binary schemes, in z-transform space [1], each arrow corresponds to a factor of $(1 + z)/2$ in the appropriate direction; for an n-ary scheme, to a factor of $(1 - z^n)/(n(1 - z))$.

We explain the arrow notation and the properties that can be derived from it for univariate (Sect. 2) and bivariate (Sect. 3) binary subdivision schemes. We use the simplest four-direction scheme (Sect. 3.6) as an example to demonstrate the four ways of deriving a scheme's mask: arrows, z-transform, matrix, and mask convolution. We enumerate all possible binary quadrilateral schemes with up to eight arrows (Sect. 3). We then consider the extension of the method to longer arrows representing factors of $1 + z^2$ (Sect. 4), to triangular meshes (Sect. 5), and to ternary schemes (Sect. 6). We conclude with suggestions for further work (Sec. 7).

2 Univariate Binary Schemes

In one-dimension we would represent the cubic box spline as four arrows:

$$\rightarrow\rightarrow\rightarrow\rightarrow$$

which corresponds to $2((1 + z)/2)^4$. This leads to the Laurent polynomial $(1 + 4z + 6z^2 + 4z^3 + z^4)/8$ which is itself the z-transform of the subdivision mask $[1, 4, 6, 4, 1]/8$ which has the two stencils $[1, 6, 1]/8$ and $[4, 4]/8$. For the purposes of this paper, we ignore the constant factor (in this case, one eighth) when it gets in the way of clear exposition, as it is trivial to derive from the fact that each stencil must sum to one.

Graphically, the arrow notation allows us to derive the mask directly by finding a number of distinct combinations (N.B., *not* permutations) of arrows which get us from the origin to each possible point on the number line. Label each arrow individually:

$$\xrightarrow{a}\xrightarrow{b}\xrightarrow{c}\xrightarrow{d}$$

There is one way to get to the origin (use no arrows), four to get to the first position (use any one of the arrows: $\{a, b, c, d\}$), six to get to the second position: $\{ab, ac, ad, bc, bd, cd\}$, four to the third: $\{abc, abd, acd, bcd\}$, and one to the fourth: $\{abcd\}$. This is simple combinatorics and it parallels exactly the derivation of the co-efficients on the polynomial product in the z-transform. The true usefulness of the graphical notation does not become apparent until we consider bivariate schemes.

Continuity can also be determined from the graphical notation. Again, this is only truly useful when we consider bivariate schemes. Each arrow represents an integration step. Each integration represents an increase in continuity by one. You may prefer to think of this as each arrow representing a multiplication

Arrows	z-transform	Mask	Continuity
$\rightarrow\rightarrow$	$2\left(\frac{(1+z)}{2}\right)^2$	$\frac{1}{2}[1, 2, 1]$	$C0$
$\rightarrow\rightarrow\rightarrow$	$2\left(\frac{(1+z)}{2}\right)^3$	$\frac{1}{4}[1, 3, 3, 1]$	$C1$
$\rightarrow\rightarrow\rightarrow\rightarrow$	$2\left(\frac{(1+z)}{2}\right)^4$	$\frac{1}{8}[1, 4, 6, 4, 1]$	$C2$
$\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow$	$2\left(\frac{(1+z)}{2}\right)^5$	$\frac{1}{16}[1, 5, 10, 10, 5, 1]$	$C3$

Fig. 1. The linear, quadratic, cubic and quartic binary univariate box-spline subdivision schemes. It is straightforward to extend this to higher powers of $(1 + z)/2$.

by a factor of $(1 + z)/2$, or a single smoothing step [2] in a refine-and-smooth formulation. In terms of the limit basis functions of the scheme, if there are no arrows, then we have an impulse function. One arrow integrates this to a step function, which is a function containing a discontinuity. A second arrow will integrate this to produce a $C0$ function. From here, each extra arrow adds one to the continuity. Thus, in the univariate case, continuity is two fewer than the number of arrows.

Figure 1 lists the first four univariate box-spline subdivision schemes.

3 Bivariate Binary Quadrilateral Schemes

The tensor product schemes are straightforward to calculate and to represent in arrow notation (Fig. 2). While it is possible to have a different number of arrows in the two primary principal directions, it is generally desirable to have the same number in each, because to do otherwise leads to an asymmetry in the subdivision scheme, which would effectively prevent the generalisation of the box spline into a subdivision scheme with extraordinary vertices.

3.1 Deriving the Mask

The mask of a subdivision scheme shows the contribution of a single original vertex to each new, subdivided vertex. To find the mask of a scheme, we need to find all ways to get from the origin to each point in the grid. For the tensor product schemes, this is simply the tensor product of the univariate case, as the two principal directions are orthogonal.

Arrows	z-transform	Mask	Continuity
$\begin{array}{l}\uparrow\\ \uparrow\!\rightarrow\rightarrow\end{array}$	$4\left(\frac{1+z_1}{2}\right)^2\left(\frac{1+z_2}{2}\right)^2$	$\frac{1}{4}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$	$C0$
$\begin{array}{l}\uparrow\\ \uparrow\\ \uparrow\!\rightarrow\rightarrow\rightarrow\end{array}$	$4\left(\frac{1+z_1}{2}\right)^3\left(\frac{1+z_2}{2}\right)^3$	$\frac{1}{16}\begin{bmatrix}1&3&3&1\\3&9&9&3\\3&9&9&3\\1&3&3&1\end{bmatrix}$	$C1$
$\begin{array}{l}\uparrow\\ \uparrow\\ \uparrow\\ \uparrow\!\rightarrow\rightarrow\rightarrow\rightarrow\end{array}$	$4\left(\frac{1+z_1}{2}\right)^4\left(\frac{1+z_2}{2}\right)^4$	$\frac{1}{16}\begin{bmatrix}1&4&6&4&1\\4&16&24&16&4\\6&24&36&24&6\\4&16&24&16&4\\1&4&6&4&1\end{bmatrix}$	$C2$

Fig. 2. The linear, quadratic and cubic tensor product bivariate box-spline subdivision schemes

The process is straightforward. Given the set of arrows for the scheme, find where each possible *combination* of arrows takes us, and then count how many combinations end up in each particular location. Take the cubic tensor product bivariate box-spline scheme. If we label each arrow individually, then the process is easy to follow:

$$\begin{tikzcd}
{} & & & & \\
{} \arrow[u, "h"'] & & & & \\
{} \arrow[u, "g"'] & & & & \\
{} \arrow[u, "f"'] & & & & \\
{} \arrow[u, "e"'] \arrow[r, "a"] & {} \arrow[r, "b"] & {} \arrow[r, "c"] & {} \arrow[r, "d"] & {}
\end{tikzcd}$$

There is one way to get from the origin to itself: use no arrows. There are 4 ways to get to the next position across: $\{a, b, c, d\}$. There are 16 ways to get to the position above that: $\{ae, af, ag, ah, be, bf, bg, bh, ce, cf, cg, ch, de, df, dg, dh\}$, and so on.

For schemes with non-orthogonal arrows, the situation is rather more interesting, and the arrows prove more useful. A detailed non-orthogonal example is given in Sect. 3.6 and Fig. 4.

3.2 Deriving Stencils

The stencils of a subdivision scheme show how to make a new, subdivided vertex from the surrounding original vertices. There are several stencils for any given scheme, each corresponding to a particular type of new vertex.

From the mask, it is straightforward to derive the stencils. For any binary bivariate scheme, there are four stencils, each derived as every second value on every second row. This is illustrated in Fig. 3. For a ternary scheme there would be nine stencils, each derived as every third value on every third row. Other arities have similar rules. Strictly, the mask should be mirror-imaged about its centre before extracting the stencils, but all masks in this paper are mirror-symmetric so this is not necessary.

3.3 Continuity

Calculating the continuity needs some explanation. We need to know what continuity to expect across any edge in the final mesh. Arrows which point *along* an edge cannot contribute to continuity *across* the edge. Therefore, to calculate continuity, find the direction with the maximum number of arrows. Discard those arrows and count the number of remaining arrows. Continuity is two fewer than this number, for the reasons given in Sect. 2. All of the tensor product schemes have the same continuity as their univariate counterparts. Note that we must consider the edges with minimum continuity and so we cannot claim any higher continuity for the scheme as a whole even if there are other directions where fewer arrows would be discarded.

If we extend to trivariate subdivision, for example for Finite Element Meshing, then a similar argument holds. The continuity across boundaries can be determined by selecting the plane that contains the maximum number of arrows, discarding these arrows, counting the remaining arrows, and subtracting two.

$$\begin{bmatrix} \boxed{1} & 4 & \boxed{6} & 4 & \boxed{1} \\ 4 & 16 & 24 & 16 & 4 \\ \boxed{6} & 24 & \boxed{36} & 24 & \boxed{6} \\ 4 & 16 & 24 & 16 & 4 \\ \boxed{1} & 4 & \boxed{6} & 4 & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} 1 & \boxed{4} & 6 & \boxed{4} & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & \boxed{24} & 36 & \boxed{24} & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & \boxed{4} & 6 & \boxed{4} & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ \boxed{4} & 16 & \boxed{24} & 16 & \boxed{4} \\ 6 & 24 & 36 & 24 & 6 \\ \boxed{4} & 16 & \boxed{24} & 16 & \boxed{4} \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & \boxed{16} & 24 & \boxed{16} & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & \boxed{16} & 24 & \boxed{16} & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Fig. 3. Deriving the four stencils from the cubic box-spline mask. Top left: vertex, top right: horizontal edge, bottom left: vertical edge, bottom right: face centre. Each should be divided by a factor of 64.

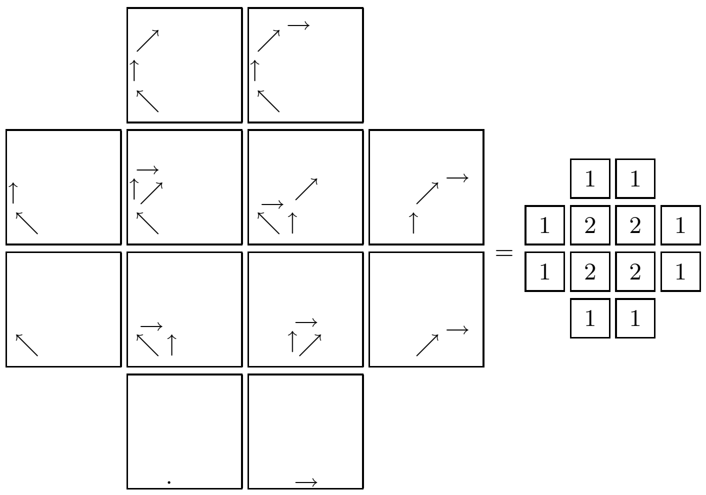


Fig. 4. Using the graphical arrow notation to derive the mask. At left we see all possible paths from the origin to each of the twelve reachable points. At right is a count of the number of paths, which is the mask of the scheme.

3.4 Safe Extrusion Directions

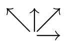
Lateral artifacts occur in the limiting surface if the original data is extruded in a direction for which the z-transform of the mask does not have a $(1+z)$ factor [3]. The arrow notation quickly allows one to see which are the safe directions: they are the ones in which there is an arrow. For the tensor product schemes, there are only two safe directions. Schemes with diagonal terms (Sect. 3.5) have four safe directions. Triangular schemes (Sect. 5) have three or six safe directions.

3.5 Diagonal Terms

In addition to horizontal and vertical arrows, quadrilateral schemes can have arrows on the 45° diagonals. These correspond to $(1 + z_1 z_2)$ and $(1 + z_2/z_1)$. We consider the horizontal and vertical arrows to be the primary principal directions, with the 45° arrows being the secondary principal directions. It is generally desirable to have the same number in each of the two primary principal directions, as in the tensor product schemes, because to do otherwise leads to an asymmetry in the subdivision scheme. Similarly, it is desirable to have the same number of arrows in each of the two secondary principal directions. However, it is not necessary to have the same number of arrows in the primary and secondary directions. For example, the tensor product schemes have no arrows in the secondary principal directions.

3.6 Four-Arrow, Four-Direction Scheme

The simplest box-spline subdivision scheme that uses diagonal terms is:



To find the mask of this scheme, we need to find all ways to get from the origin to each point in the grid (Fig. 4). Alternatively, we can get the same answer from the z-transform by expanding $(1 + z_1)(1 + z_2)(1 + z_1 z_2)(1 + z_2/z_1)$:

$$\begin{array}{cccc} & z_1^0 z_2^3 + & z_1^1 z_2^3 + & \\ z_1^{-1} z_2^2 + & 2z_1^0 z_2^2 + & 2z_1^1 z_2^2 + & z_1^2 z_2^2 + \\ z_1^{-1} z_2^1 + & 2z_1^0 z_2^1 + & 2z_1^1 z_2^1 + & z_1^2 z_2^1 + \\ & z_1^0 z_2^0 + & z_1^1 z_2^0 & \end{array}$$

Arranging the terms horizontally by increasing exponent on z_1 and vertically by increasing exponent on z_2 produces an array where the coefficients on the terms are the coefficients of the mask.

A third alternative is to use mask convolution. The four simple masks, each of which represents a $(1+z)$ term or an arrow in a principal direction, are convolved to produce the final mask.

$$\begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \begin{bmatrix}0&1&1&0\\1&2&2&1\\1&2&2&1\\0&1&1&0\end{bmatrix}$$

The final alternative is to use Peters and Shuie's matrix of directions [4]:

$$\mathcal{A}_{\text{simplest}} = \begin{bmatrix} 1 & 0\\ 0 & 1\\ 1 & 1\\ -1 & 1\end{bmatrix}.$$

$$\begin{bmatrix} \boxed{0} & 1 & \boxed{1} & 0 \\ 1 & 2 & 2 & 1 \\ \boxed{1} & 2 & \boxed{2} & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & \boxed{1} & 1 & \boxed{0} \\ 1 & 2 & 2 & 1 \\ 1 & \boxed{2} & 2 & \boxed{1} \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 0 \\ \boxed{1} & 2 & \boxed{2} & 1 \\ 1 & 2 & 2 & 1 \\ \boxed{0} & 1 & \boxed{1} & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & \boxed{2} & 2 & \boxed{1} \\ 1 & 2 & 2 & 1 \\ 0 & \boxed{1} & 1 & \boxed{0} \end{bmatrix}$$

Fig. 5. The four stencils derived from the mask. All four are rotational variants of one another.

Each direction corresponds to one of the arrows in the arrow notation, to one of the terms in the Laurent polynomial (z-transform), and to one of the masks in the mask convolution method.

The four stencils of the scheme can be derived by taking every second row from every second column, in the four possible ways this can be done (Fig. 5).

One of the interesting things about this scheme is that it can be factorised into a $\sqrt{2}$ scheme. One step of that scheme being $\nwarrow\nearrow$ combined with the rotation of 45°, the next step being $\uparrow\!\rightarrow$ with a further rotation of 45°, which realigns the subdivided mesh's primary directions with the original mesh's primary directions. This is the "simplest" subdivision scheme described by Peters and Reif [5].

The mask of $\nwarrow\nearrow$ is $\begin{bmatrix} & 1 & \\ 1 & & 1 \\ & 1 & \end{bmatrix}$. The two stencils are $\begin{bmatrix} 1 & 1 \end{bmatrix}$ for horizontal edges and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ for vertical edges. The mask of $\uparrow\!\rightarrow$ is $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. This is exactly the other simple mask rotated by 45° and contracted by a factor of $\sqrt{2}$. Convolving the two produces the mask of the binary scheme:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

To derive the continuity of the scheme, consider the direction with the greatest number of arrows. There is one arrow in each of the principal directions, whichever you choose. This leaves three arrows and the continuity is two fewer than that. Therefore the simplest scheme has $C1$-continuity in regular regions. It has four safe extrusion directions: the four principal directions.

In general, a binary box-spline scheme can be factored into a $\sqrt{2}$ scheme if the arrows can be split into two sets, one of which maps onto the other by a rotation of 45° and a dilation of $\sqrt{2}$.

3.7 Eight-Arrow, Four-Direction Scheme

In a similar way, we can evaluate the scheme with two arrows in each of the four principal directions. This is equivalent to two steps of the 4–8 scheme described by Velho [6].

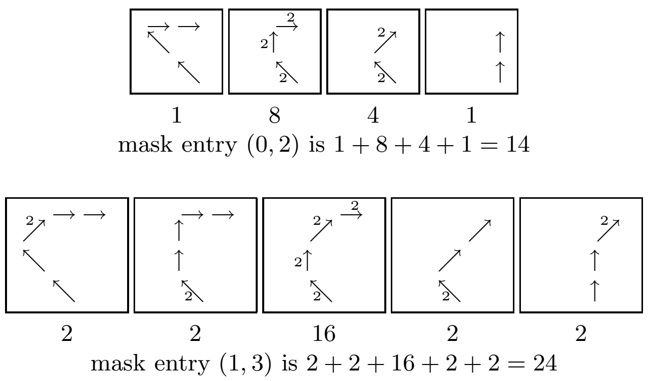
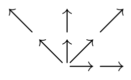

1 8 4 1
mask entry $(0, 2)$ is $1 + 8 + 4 + 1 = 14$
2 2 16 2 2
mask entry $(1, 3)$ is $2 + 2 + 16 + 2 + 2 = 24$

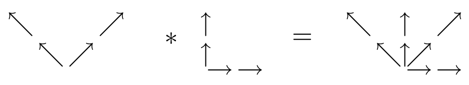
Fig. 6. Two examples of determining the mask entries using the arrow notation, for the eight-arrow, four-direction scheme. Arrows carry an annotation “2” when there are two possible arrows available. The number of possible combinations is shown under each diagram.



This scheme has continuity $C4$. This is determined by finding the direction with the greatest number of arrows (any one of the four principal directions), counting the number of arrows not in this direction (six) and subtracting two.

The entries in the mask can be determined in any of the ways described above. As an example, consider using the arrows to determine the entries. Figure 6 shows the derivation of two of the entries in the mask by this method. Clearly the more arrows there are, the more complex this procedure becomes and the less useful as a mechanism for deriving the mask entries.

Like the four-arrow, four-direction scheme, this binary scheme is factorisable into a $\sqrt{2}$ scheme:



The two $\sqrt{2}$ masks can be convolved to produce the mask of the binary scheme:

$$\begin{bmatrix} 0&0&1&0&0 \\ 0&2&0&2&0 \\ 1&0&4&0&1 \\ 0&2&0&2&0 \\ 0&0&1&0&0 \end{bmatrix} * \begin{bmatrix} 1&2&1 \\ 2&4&2 \\ 1&2&1 \end{bmatrix} = \begin{bmatrix} 0&0&1&2&1&0&0 \\ 0&2&6&8&6&2&0 \\ 1&6&14&18&14&6&1 \\ 2&8&18&24&18&8&2 \\ 1&6&14&18&14&6&1 \\ 0&2&6&8&6&2&0 \\ 0&0&1&2&1&0&0 \end{bmatrix}$$

$$\begin{bmatrix} & & 1 & & \\ & \boxed{2} & & \boxed{2} & \\ 1 & & 4 & & 1 \\ & \boxed{2} & & \boxed{2} & \\ & & 1 & & \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \qquad \begin{bmatrix} & & \boxed{1} & & \\ & 2 & & 2 & \\ \boxed{1} & & \boxed{4} & & \boxed{1} \\ & 2 & & 2 & \\ & & \boxed{1} & & \end{bmatrix} \rightarrow \begin{bmatrix} & 1 & \\ 1 & 4 & 1 \\ & 1 & \end{bmatrix}$$

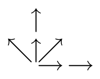
Fig. 7. The stencils of Velho's 4–8 scheme [6] derived from its mask. There is one stencil for introducing new vertices at face centres (left) and one stencil for moving old vertices (right). Note that values in the mask and stencils must be divided by eight to ensure that the values in each stencil sum to one.

You can easily extract Velho's 4–8 stencils from the $\sqrt{2}$ mask (Fig. 7). However, you need to note that there are only two stencils in a $\sqrt{2}$ scheme, compared with four in a binary scheme. Clearly, you could extract four stencils from the binary mask, and thus directly create a binary scheme. This would, however, produce stencils which are large. Large stencils make it more difficult to create mechanisms for the efficient handling of extraordinary vertices, edges and creases in the mesh.

3.8 Six-Arrow Schemes

There are two schemes which each have six arrows with at least one in each of the four principal directions, and which each produce smaller stencils than the binary version of Velho's 4–8 scheme.

The first of the two six-arrow schemes is the quadrilateral part of Peters and Shiue's 4–3 scheme [4]. In arrow notation it is:



From this we can see that the scheme is $C2$ in regular regions. It has the mask:

$$\begin{bmatrix} & 1 & 2 & 1 & \\ 1 & 4 & 6 & 4 & 1 \\ 2 & 6 & 8 & 6 & 2 \\ 1 & 4 & 6 & 4 & 1 \\ & 1 & 2 & 1 & \end{bmatrix}$$

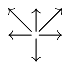
and the four stencils:

$$\begin{bmatrix} & 2 & \\ 2 & 8 & 2 \\ & 2 & \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 6 & 6 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 6 & 1 \\ 1 & 6 & 1 \end{bmatrix} \qquad \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$$

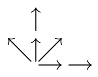
As illustrated in Sect. 3.6, the arrow notation is a straightforward graphical representation of the matrix of directions as used by Peters and Shuie [4]:

$$\mathcal{A}_{\triangle} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 1 \\ -1 & 1 \end{bmatrix}.$$

This corresponds to the arrow notation:

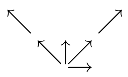


A shift of origin makes no difference to the resulting mask, so this is equivalent to:



We prefer the latter version, where all of the arrows which lie on the same line point in the same direction, as we believe that this makes the notation clearer.

The other six-arrow box-spline is:



which is clearly also $C2$ in regular regions. This has the mask:

$$\begin{bmatrix} & & 1 & 1 & & \\ & 2 & 3 & 3 & 2 & \\ 1 & 3 & 6 & 6 & 3 & 1 \\ 1 & 3 & 6 & 6 & 3 & 1 \\ & 2 & 3 & 3 & 2 & \\ & & 1 & 1 & & \end{bmatrix}$$

and its four stencils are all rotational variants of:

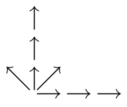
$$\begin{bmatrix} & 1 & \\ 1 & 6 & 3 \\ & 3 & 2 \end{bmatrix}$$

This is a dual binary quadrilateral subdivision scheme, making it most closely related to the quadratic tensor product box spline ($C1$, Doo-Sabin [7,8]) and the quartic tensor product box spline ($C3$) [9,10]. This second six-arrow box spline has never been extended to handle extraordinary cases, edges, or creases. We expect that this could be done reasonably easily, given the simplicity of the stencil.

Note that neither of the six-arrow schemes can be factorised into a $\sqrt{2}$ scheme, because neither meets the criteria described at the end of Sect. 3.6.

3.9 More Arrows — Larger Stencils

It is clearly possible to add more arrows. For example, adding two more arrows to either Peters and Shiue's 4–3 scheme or the Doo-Sabin scheme produces an eight-arrow box spline:



This is $C3$. Its mask is:

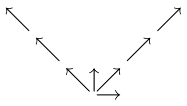
$$\begin{bmatrix} & 1 & 3 & 3 & 1 & \\ 1 & 6 & 13 & 13 & 6 & 1 \\ 3 & 13 & 24 & 24 & 13 & 3 \\ 3 & 13 & 24 & 24 & 13 & 3 \\ 1 & 6 & 13 & 13 & 6 & 1 \\ & 1 & 3 & 3 & 1 & \end{bmatrix}$$

and its four stencils are all rotational variants of:

$$\begin{bmatrix} & 3 & 1 \\ 3 & 24 & 13 \\ 1 & 13 & 6 \end{bmatrix}$$

This is yet another dual binary quadrilateral subdivision scheme. This eight-arrow box spline has never been extended to handle extraordinary cases, edges, or creases.

The final binary quadrilateral scheme with eight arrows is:



This is $C3$ and has a large mask:

$$\begin{bmatrix} & & & 1 & 1 & & & \\ & & 3 & 4 & 4 & 3 & & \\ & 3 & 6 & 12 & 12 & 6 & 3 & \\ 1 & 4 & 12 & 18 & 18 & 12 & 4 & 1 \\ 1 & 4 & 12 & 18 & 18 & 12 & 4 & 1 \\ & 3 & 6 & 12 & 12 & 6 & 3 & \\ & & 3 & 4 & 4 & 3 & & \\ & & & 1 & 1 & & & \end{bmatrix}$$

and a large stencil:

$$\begin{bmatrix} & & 1 & \\ & 6 & 12 & 3 \\ 1 & 12 & 18 & 4 \\ & 3 & 4 & \end{bmatrix}$$

$$\begin{bmatrix} & \\ 128 & 64 \\ 64 & \end{bmatrix} \quad \begin{bmatrix} & \\ 144 & 48 \\ 48 & 16 \end{bmatrix} \quad \begin{bmatrix} & 16 & \\ 16 & 96 & 48 \\ & 48 & 32 \end{bmatrix} \quad \begin{bmatrix} & 12 & 4 \\ 12 & 96 & 52 \\ 4 & 52 & 24 \end{bmatrix} \quad \begin{bmatrix} 1 & 10 & 5 \\ 10 & 100 & 50 \\ 5 & 50 & 25 \end{bmatrix}$$

Fig. 8. The stencils of the five dual quadrilateral binary box-spline schemes with masks of up to 3×3. All stencils have the common denominator of 256 to allow easy comparison. From left to right: simplest ($C0$) [5], Doo-Sabin ($C1$) [7,8], six-arrow ($C2$), eight-arrow ($C3$), quartic tensor product (ten-arrow, $C3$) [9,10].

This is another binary dual scheme. In this case, the stencil is so large as to make it difficult to generalise to extraordinary cases.

Consider all of the dual schemes with stencils up to size 3×3. We know, simply by enumerating all possible combinations of arrows, that there are only five of them (ignoring the trivial case which has just two arrows and which does not produce a limit surface). If we use a common weighting factor of 1/256, then the stencils are as shown in Fig. 8. The largest is the biquartic box-spline, for which extraordinary cases were considered by Qu [9], Zorin and Schröder [10]. It is not clear what, if any, advantage would be gained from the three larger stencils, compared with the two smaller ones. It would be interesting to have all five implemented, and compared against one another, and for mechanisms to be developed to handle extraordinary cases, edges and creases for the six-arrow and eight-arrow schemes.

4 Longer Arrows

It is possible to create schemes with squared or higher terms of z in their z-transform. In the arrow notation these are represented by longer arrows. For example $(1 + z^2)$ is represented by an arrow of twice the length of that representing $(1 + z)$.

The most interesting place where such a binary scheme arises is when we consider a $\sqrt{2}$ scheme with the arrow symbol:

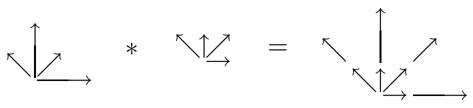
$$\nwarrow\uparrow\nearrow\rightarrow$$

This is the same symbol as for the binary simplest scheme (Sect. 3.6). However, it can also be implemented as a $\sqrt{2}$ scheme in its own right. This is done by deriving two, rather than four, stencils from the mask (Fig. 4). The stencils are read off from the mask at 45°. Rotating them by 45°, the two stencils are:

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}$$

for new vertices at the nominal centres of the vertical and horizontal edges respectively. This is straightforward to implement and it should be relatively straightforward to generalise to the extraordinary cases, edges and creases.

Taking the convolution of two $\sqrt{2}$ steps, where the second step is rotated by $45°$ and dilated by $\sqrt{2}$, gives a binary scheme with arrow symbol:



and with z-transform $(1+z_1)(1+z_1^2)(1+z_2)(1+z_2^2)(1+z_1z_2)^2(1+z_2/z_1)^2$. Note the positions of the exponents both inside and outside the parentheses.

The mask of this binary scheme is:

$$\begin{bmatrix} & & 1 & 1 & 1 & 1 & & \\ & 2 & 3 & 5 & 5 & 3 & 2 & \\ 1 & 3 & 8 & 10 & 10 & 8 & 3 & 1 \\ 1 & 5 & 10 & 14 & 14 & 10 & 5 & 1 \\ 1 & 5 & 10 & 14 & 14 & 10 & 5 & 1 \\ 1 & 3 & 8 & 10 & 10 & 8 & 3 & 1 \\ & 2 & 3 & 5 & 5 & 3 & 2 & \\ & & 1 & 1 & 1 & 1 & & \end{bmatrix}$$

Note the four entries of “1” along each of the horizontal and vertical edges. This is the most obvious indication that those double length arrows are doing something different to that observed when only single length arrows are used.

The stencils of the binary scheme are the four rotations of:

$$\begin{bmatrix} & 3 & 5 & 2 \\ 1 & 10 & 14 & 5 \\ 1 & 8 & 10 & 3 \\ & 1 & 1 & \end{bmatrix}$$

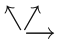
This is a big stencil and, thus, one would not expect to implement it as a binary scheme nor try to extend the binary scheme to extraordinary cases. Instead, as with 4–8, any such extension would be implemented for the $\sqrt{2}$ scheme.

While this scheme is certainly valid there is an interesting question, when determining the continuity, as to how those double length arrows contribute towards continuity. If they contribute as for single length arrows, then the scheme is $C4$ in regular regions. If not, it is almost certainly at least $C2$. Analysis of these double length arrows is best done in the univariate case in a similar way to that done by Dyn [11] and Hassan [12]. Regardless of whether it is $C4$ or $C2$ in regular regions, the scheme would benefit from further investigation and extension to extraordinary cases.

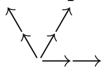
We note, in passing, that it would also be possible to have “knight’s move” arrows. It is unclear what the implications of these would be. Would they provide extra safe extrusion directions? Would they contribute towards higher continuity?

5 Triangular Schemes

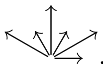
Triangular schemes have six principal directions, three primary and three secondary, but are otherwise handled in much the same way as for quadrilaterals. Two of the primary directions are $(1 + z_1)$ and $(1 + z_2)$. It is a matter of convention whether the third primary direction should be treated as $(1 + z_1 z_2)$ or $(1 + z_2/z_1)$, depending on whether you prefer the positive z_1 and z_2 axes to be separated by 60° or 120°. Note that $(1 + z_2/z_1)$ can be shifted to $(z_1 + z_2)$ if you prefer only non-negative powers of z. The shift has no effect on the resulting mask.

The simplest box-spline triangular scheme, linear interpolation, is $C0$, has the arrow symbol  and the mask:

$$\begin{array}{ccccc} & 1 & & 1 & \\ 1 & & 2 & & 1 \\ & 1 & & 1 & \end{array}$$

The next simplest, Loop subdivision [13], is $C2$ in regular regions, has the arrow symbol  and the mask:

$$\begin{array}{ccccccccc} & & 1 & & 2 & & 1 & & \\ & 2 & & 6 & & 6 & & 2 & \\ 1 & & 6 & & 10 & & 6 & & 1 \\ & 2 & & 6 & & 6 & & 2 & \\ & & 1 & & 2 & & 1 & & \end{array}$$

We can introduce secondary principal directions with the six-arrow symbol . This has the binary mask:

$$\begin{array}{ccccccccccc} & & & & 1 & & 1 & & & & \\ & 1 & & 2 & & 2 & & 2 & & 1 & \\ 1 & & 2 & & 4 & & 4 & & 2 & & 1 \\ & 2 & & 4 & & 4 & & 4 & & 2 & \\ 1 & & 2 & & 4 & & 4 & & 2 & & 1 \\ & 1 & & 2 & & 2 & & 2 & & 1 & \\ & & & & 1 & & 1 & & & & \end{array}$$

with four stencils, one for the vertex and three rotational variants for the three edges:

$$\begin{array}{ccccc} & 2 & & 2 & \\ 2 & & 4 & & 2 \\ & 2 & & 2 & \end{array} \qquad\qquad \begin{array}{ccccc} 1 & & 2 & & 1 \\ & 4 & & 4 & \\ 1 & & 2 & & 1 \end{array}$$

However, a binary scheme *cannot* be factorised into two $\sqrt{2}$ steps, because there is no way to construct a triangular $\sqrt{2}$ subdivision scheme [14,15]. Indeed, the longer arrows are $\sqrt{3}$ the length of the shorter arrows, rather than $\sqrt{2}$. To get a factorisable version, you must construct a ternary scheme, which can be factorised into the convolution of two $\sqrt{3}$ steps.

6 Ternary and Higher Arities

The arrow notation extends to box-spline schemes of higher arities. For example, the ternary arrow $\rightarrow$, which looks identical to the binary arrow, corresponds to $(1+z+z^2)/3$. This means that there are three possible ways in which the arrow can be used: no translation, a translation of one unit, and a translation of two units. Compare this with the binary arrow, which corresponds to $(1+z)/2$, where we can interpret it as either translation of one unit or no translation, which is equivalent to either using the arrow or not using it. So long as we remember that the geometric interpretation of the ternary arrow is somewhat different, we can proceed as for the binary case. In general, the n-ary arrow corresponds to $(1-z^n)/(n(1-z))$ which is $(1+z+\cdots+z^{n-1})/n$.

6.1 Ternary Univariate Schemes

As for the binary univariate schemes, we can list the possible ternary univariate box-spline schemes (Fig. 9). Remember that each of these schemes has three stencils, obtained by taking every third element from the mask. For example, the cubic ternary scheme has the three stencils $\frac{1}{27}[1,16,10]$, $\frac{1}{27}[4,19,4]$, and $\frac{1}{27}[10,16,1]$.

Arrows	z-transform	Mask	Continuity
$\rightarrow\rightarrow$	$3\left(\frac{(1+z+z^2)}{3}\right)^2$	$\frac{1}{3}[1,2,3,2,1]$	$C0$
$\rightarrow\rightarrow\rightarrow$	$3\left(\frac{(1+z+z^2)}{3}\right)^3$	$\frac{1}{9}[1,3,6,7,6,3,1]$	$C1$
$\rightarrow\rightarrow\rightarrow\rightarrow$	$3\left(\frac{(1+z+z^2)}{3}\right)^4$	$\frac{1}{27}[1,4,10,16,19,16,10,4,1]$	$C2$

Fig. 9. The linear, quadratic, and cubic ternary univariate box-spline subdivision schemes

6.2 Ternary Bivariate Schemes

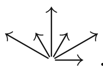
All of the derivations which work for the binary schemes also work for the ternary schemes. For example, we can easily derive the ternary mask for the arrow symbol $\overset{\nwarrow\;\nearrow}{\nwarrow\!\nearrow}\!\rightarrow\rightarrow$. This produces the ternary version of Loop subdivision [16] which has the mask:

```
            1    2    3    2    1
          2    6   10   10    6    2
        3   10   20   24   20   10    3
      2   10   24   36   36   24   10    2
    1   6   20   36   45   36   20    6    1
      2   10   24   36   36   24   10    2
        3   10   20   24   20   10    3
          2    6   10   10    6    2
            1    2    3    2    1
```

Selecting every third entry on every third line generates the stencils shown below, along with the five rotations of the second stencil and the one rotation of the third stencil:

```
  6   6             2   10            3   24   3
6   45   6        1   36   20           24   24
  6   6             2   10                3
```

These stencils are for new vertices at the vertex, a third of the way along an edge, and face centre respectively.

We can also generate the ternary version of the six-arrow, six-direction box spline, with arrow symbol . This has the mask:

```
                      1    1    1
                 1    2    3    3    2    1
            1    2    4    6    7    6    4    2    1
         1    3    6    9   11   11    9    6    3    1
      1    3    7   11   15   16   15   11    7    3    1
         2    6   11   16   19   19   16   11    6    2
      1    4    9   15   19   21   19   15    9    4    1
         2    6   11   16   19   19   16   11    6    2
      1    3    7   11   15   16   15   11    7    3    1
         1    3    6    9   11   11    9    6    3    1
            1    2    4    6    7    6    4    2    1
                 1    2    3    3    2    1
                      1    1    1
```

This *can* be factorised into two $\sqrt{3}$ steps, which are much simpler to evaluate.

```
            1
      1           1
1           2           1                1  1  1
      2           2                    1  2  2  1
1           3           1       *    1  2  3  2  1
      2           2                    1  2  2  1
1           2           1                1  1  1
      1           1
            1
```

The stencils which can be derived from these $\sqrt{3}$ masks are:

$$\begin{array}{ccccc} & 1 & & 1 & \\ 1 & & 3 & & 1 \\ & 1 & & 1 & \end{array} \qquad\qquad \begin{array}{ccccc} 1 & & 2 & & 1 \\ & 2 & & 2 & \\ & & 1 & & \end{array}$$

for new vertices at the vertex and face centre respectively. Note that this is *not* the Kobbelt $\sqrt{3}$ scheme [17] and that the Kobbelt scheme is not a box-spline scheme.

7 Summary

The four different methods of deriving subdivision masks all have their benefits. The arrow notation is useful in that it is a graphical, rather than strictly mathematical, representation and in that it allows us to read off the continuity of the box-spline scheme directly. It also allows us to enumerate all possible schemes easily by forming all possible combinations of arrows. This paper has shown all possible symmetric binary quadrilateral schemes which have up to eight arrows, and all possible symmetric binary triangular schemes which have up to six arrows.

There are interesting small projects that could be tackled, arising from this work, each involving investigating the generalisation of a box-spline scheme or schemes to the extraordinary cases, edges and creases. They are:

1. an investigation of the family of dual quadrilateral binary box-spline schemes illustrated in Fig. 8;
2. an investigation of the $\sqrt{2}$ scheme described in Sect. 4, including consideration of the effect of arrows which are longer than the shortest primary and secondary arrows; and
3. an investigation of the $\sqrt{3}$ scheme described at the end of Sect. 6.

References

1. Dyn, N.: Analysis of convergence and smoothness by the formalism of Laurent polynomials. In: Iske, A., Quak, E., Floater, M.S. (eds.) Tutorials on Multiresolution in Geometric Modelling, pp. 51–68. Springer, Heidelberg
2. Sabin, M.A., Augsdörfer, U.H., Dodgson, N.A.: Artifacts in box-spline surfaces. In: Martin, R., Bez, H., Sabin, M. (eds.) IMA 2005. LNCS, vol. 3604, pp. 350–363. Springer, Heidelberg (2005)
3. Sabin, M., Barthe, L.: Artifacts in recursive subdivision schemes. In: Cohen, A., Merrien, J.L., Schumaker, L.L. (eds.) Curve and Surface Fitting: Saint-Malo 2002, pp. 353–362. Nashboro Press (2003)
4. Peters, J., Shiue, L.J.: Combining 4- and 3-direction subdivision. ACM Trans. Graph. 23(4), 980–1003 (2004)
5. Peters, J., Reif, U.: The simplest subdivision scheme for smoothing polyhedra. ACM Trans. Graph. 16(4), 420–431 (1997)

6. Velho, L.: Quasi 4-8 subdivision. Computer Aided Geometric Design 18(4), 345–357 (2001)
7. Doo, D.: A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In: Proceedings on Interactive Techniques in Computer Aided Design, pp. 157–165 (1978) (Janurary 2009), http://www.idi.ntnu.no/~fredrior/files/Doo%201978%20Subdivision%20algorithm.pdf
8. Doo, D., Sabin, M.A.: Behaviour of recursive division surfaces near extraordinary points. Computer-Aided Design 10(6), 356–360 (1978)
9. Qu, R.: Recursive subdivision algorithms for curve and surface design. PhD thesis, Brunel University (1990)
10. Zorin, D., Schrder, P.: A unified framework for primal/dual quadrilateral subdivision schemes. Computer Aided Geometric Design 18(5), 429–454 (2001)
11. Dyn, N.: Subdivision schemes in computer-aided geometric design. In: Light, W. (ed.) Advances in Numerical Analysis, vol. 2, pp. 36–104. Clarendon Press (1992)
12. Hassan, M.F., Ivrissimtzis, I.P., Dodgson, N.A., Sabin, M.A.: An interpolating 4-point C2 ternary stationary subdivision scheme. Computer Aided Geometric Design 19(1), 1–18 (2002)
13. Loop, C.T.: Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics (1987)
14. Ivrissimtzis, I.P., Dodgson, N.A., Sabin, M.A.: A generative classification of mesh refinement rules with lattice transformations. Computer Aided Geometric Design 21(1), 99–109 (2004)
15. Dodgson, N.A.: An heuristic analysis of the classification of bivariate subdivision schemes. In: Martin, R.R., Bez, H.E., Sabin, M.A. (eds.) IMA 2005. LNCS, vol. 3604, pp. 161–183. Springer, Heidelberg (2005)
16. Loop, C.: Smooth ternary subdivision of triangle meshes. In: Cohen, A., Merrien, J.L., Schumaker, L.L. (eds.) Curve and Surface Fitting: Saint-Malo 2002, pp. 295–302. Nashboro Press (2003)
17. Kobbelt, L.: $\sqrt{3}$ subdivision. In: Proc. ACM SIGGRAPH, pp. 103–112 (2000)

Geometric Characterizations of Graphs Using Heat Kernel Embeddings

Hewayda El-Ghawalby[1,2] and Edwin R. Hancock[1,⋆]

[1] Department of Computer Science, University of York, UK
[2] Faculty of Engineering, Suez Canal University, Egypt
{howaida,erh}@cs.york.ac.uk

Abstract. In this paper, we investigate the heat kernel embedding as a route to computing geometric characterisations of graphs. The reason for turning to the heat kernel is that it encapsulates information concerning the distribution of path lengths and hence node affinities on the graph. The heat kernel of the graph is found by exponentiating the Laplacian eigensystem over time. The matrix of embedding co-ordinates for the nodes of the graph is obtained by performing a Young-Householder decomposition on the heat kernel. Once the embedding of its nodes is to hand we proceed to characterise a graph in a geometric manner. To obtain this characterisation, we focus on the edges of the graph under the embedding. Here we use the difference between geodesic and Euclidean distances between nodes to associate a sectional curvature with edges. Once the section curvatures are to hand then the Gauss-Bonnet theorem allows us to compute Gaussian curvatures at nodes on the graph. We explore how the attributes furnished by this analysis can be used to match and cluster graphs.

Keywords: Graph spectra, kernel methods, graph embedding, differential geometry, graph clustering.

1 Introduction

Graphs are used pervasively in computer science as representations of data with a network or relational structure. However, the analysis of data in this form has proved an elusive problem. The reason for this is that most of the available pattern analysis tools are couched in terms of vectorial rather than graph representations. One way to circumvent this problem is to is to embed the nodes of a graph in a vector space and to study the properties of the point distribution that results from the embedding. This is a problem that arises in a number of areas including manifold learning theory and graph-drawing. In the mathematics literature, there is a considerable body of work aimed at understanding how graphs can be embedded on manifolds [14]. In the pattern analysis community,

⋆ The authors acknowledge the financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

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 [23]. 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 PCA that restricts the complexity of the input data using a nearest neighbor graph [17], and the Laplacian eigenmap 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) [3]. 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 [4] and has proved to be a versatile mathematical tool that can be put to many practical uses including routing [1], indexing [19], clustering [18] and graph-matching [24,15].

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 [26]. 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 eigensystem 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. Recently, Lebanon and Lafferty [12] have shown how the heat kernel to construct statistical manifolds that can be used for inference and learning tasks. Moreover, in related work we have explored how a number of different invariants that can be computed from the heat kernel can be used for graph clustering [25]. Colin de Verdiere has shown how to compute geodesic invariants from the Laplacian spectrum [6].

In fact, a graph can be viewed as residing on a manifold whose pattern of geodesic distances is characterised by the heat kernel. Differential invariants can be computed from the heat kernel, and these in turn are related to the Laplacian eigensystem. This field of study is sometimes referred to as spectral geometry [8,26]. 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 [9,28,2,20]. Here considerable insight can be achieved through the analysis of the heat kernel of the graph [9,2]. 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 [4] (or the sum of the Laplacian eigenvalues exponentiated with time) can be expanded as a rational polynomial in time, and the co-efficiants 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 aim in this paper is to investigate whether the heat kernel can be used to provide a geometric characterisation of graphs that can be used for the purposes of graph-clustering. This is of course a problem that can be addressed directly by using the spectral geometry of the combinatorial Laplacian. However, there are two major obstacles. First, the results delivered by spectral geometry are interesting, they apply under the assumption that the graph Laplacian converges to the corresponding continuous Laplace operator provided that the graph is sufficiently large. Second, the calculations involved are intricate and the resulting expressions are not very elegant. Hence, we adopt a more pragmatic approach in this paper where we aim to characterise the geometry of point distribution based on embeddings derived from the heat-kernel.

The method involves performing a Young-Householder decomposition of the heat-kernel to recover the matrix of embedding co-ordinates. In other words, we perform kernel principal components analysis on the heat kernel to map nodes of the graph to points in the vector space. We provide an analysis which shows how the eigenvalues and eigenvectors of the covariance matrix for the point distribution resulting from the kernel mapping are related to those of the Laplacian.

With the embeddings to hand, we develop a graph characterisation based on differential geometry. To do this we compute the sectional curvatures associated with the edges of the graph, making use of the fact that the sectional curvature is determined by the difference between the geodesic and Euclidean distances. We take this analysis one step further, and use the Gauss-Bonnet theorem to compute the Gaussian curvatures associated with triangular faces of the graph. We characterise graphs using sets of curvatures, defined either on the edges or the faces. We explore whether these characterisations can be used for the purposes of graph matching and graph clustering. To this end, we compute the similarities of the sets using robust variants of the Hausdorff distance. This allows us to compute the similarity of different graphs without knowing the correspondences between edges or faces.

The outline of this paper is as follows. In Section 2 we provide some background on the heat-kernel and its relationship with the Laplacian spectrum. Section 3 describes how the Young-Householder factorisation of the heat kernel leads to a co-ordinate embedding of the nodes of a graph. Section 4 shows how the Euclidean and geodesic distances between nodes under the embedding can be used to estimate sectional curvatures for the edges and Gaussian curvatures for the triangular faces, using two different embeddings. In Section 5 we experiment with the method on the houses database. Finally, Section 7 offers some conclusions and directions for future research.

2 Heat Kernels on Graphs

In this section, we give a brief introduction on the graph 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

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

We also construct the diagonal degree matrix D, whose elements are given by the degree of the nodes, i.e. $D(u,u) = deg(u) = \sum_{v \in V} A(u,v)$. 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,

$$L(u,v) = \begin{cases} deg(v) & \text{if} \quad u = v \\ -1 & \text{if} \quad u \quad \text{and} \quad v \quad \text{are} \quad \text{adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The normalized Laplacian $\hat{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ has elements

$$\hat{L}(u,v) = \begin{cases} 1 & \text{if} \quad u = v \quad \text{and} \quad d_v \neq 0 \\ -\frac{1}{\sqrt{deg(u)deg(v)}} & \text{if} \quad u \quad \text{and} \quad v \quad \text{are} \quad \text{adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The spectral decomposition of the normalized Laplacian matrix is $\hat{L} = \Phi \Lambda \Phi^T$, where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})(\lambda_1 < \lambda_2 < ... < \lambda_{|V|})$is the diagonal matrix with the ordered eigenvalues as elements and $\Phi = (\phi_1|\phi_2|....|\phi_{|V|})$ is the matrix with the ordered eigenvectors as columns. Since $\hat{L}$ is symmetric and positive semi-definite, the eigenvalues of the normalized Laplacian are all non-negative. The multiplicity of the zero eigenvalue is the number of isolated cliques in the graph. For a connected graph, the multiplicity of the zero eigenvalue is one. The eigenvector associated with the smallest non-zero eigenvector is referred to as the Fiedler-vector [4].

2.1 Heat Equation

We are interested in the heat equation associated with the Laplacian, and this is given by.

$$\frac{\partial h_t}{\partial t} = -\hat{L} h_t \tag{4}$$

where h_t is the heat kernel and t is time. The heat kernel is the fundamental solution of the heat equation. It can 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

$$h_t = e^{-t\hat{L}} \tag{5}$$

From [4] we can proceed to compute the heat kernel on a graph by exponentiating the Laplacian eigenspectrum, i.e.

$$h_t = \Phi \exp[-\Lambda t] \Phi^T = \sum_{i=1}^{|V|} \exp[-\lambda_i t] \phi_i \phi_i^T \tag{6}$$

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

$$h_t(u,v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t]\phi_i(u)\phi_i(v) \tag{7}$$

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 $h_t \simeq I - \exp[-\lambda_2 t]\phi_2\phi_2^T$, 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.

2.2 Geodesic Distance from the Heat Kernel

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 the MacLaurin expansion on the heat kernel to re-express it as a polynomial in t. The result of this expansion is

$$h_t = e^{-t}(I + tP + \frac{(tP)^2}{2!} + \frac{(tP)^3}{3!} + ...) = e^{-t}\sum_{k=0}^{\infty} P^k \frac{t^k}{k!} \tag{8}$$

For a connected graph, the matrix P has elements

$$P(u,v) = \begin{cases} 0 & \text{if} \quad u = v \\ \frac{1}{\sqrt{deg(u)deg(v)}} & \text{if} \quad u \neq v \quad \text{and} \quad (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

As a result, we have that

$$P^k(u,v) = \sum_{S_k} \prod_{i=1}^{k} \frac{1}{\sqrt{deg(u_i)deg(u_{i+1})}} \tag{10}$$

where the walk S_k is a sequence of vertices $u_0, ..., u_k$ of length k such that $(u_i, u_{i+1}) \in E$. 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

$$h_t(u,v) = \exp[-t]\sum_{k=0}^{|V|^2} P^k(u,v)\frac{t^k}{k!} \tag{11}$$

We can find a spectral expression for the matrix P^k using the eigendecomposition of the normalized Laplacian. Writing $P^k = (I - \hat{L})^k$ it follows that $P^k = \Phi(I - \Lambda)^k\Phi^T$. The element associated with the nodes u and v is

$$P^k(u,v) = \sum_{i=1}^{|V|}(1 - \lambda_i)^k \phi_i(u)\phi_i(v) \tag{12}$$

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. $d_G(u,v) = floor_k P_k(u,v)$.

3 Heat Kernel Embedding

In this section we first show how the heat kernel can be used to embed the nodes of a graph in a vector space using the Young-Householder decomposition. Second, we provide an analysis revealing the relationship between the eigenvalues and eigenvectors of the heat-kernel and those of the covariance matrix for the point distribution resulting from the embedding.

3.1 Co-ordinate Embedding

We use the heat kernel to map the nodes of the graph into a vector space. Let $Y = (y_1|...|y_u|...|Y_{|V|})$ be the $|V| \times |V|$ matrix with the vectors of co-ordinates as columns. The vector of co-ordinates y_u for the node index u is hence the u^{th} column of Y. The co-ordinate matrix is found by performing the Young-Householder decomposition $h_t = Y^T Y$ on the heat-kernel. Since $h_t = \Phi \exp[-\Lambda t]\Phi^T$, $Y = \exp[-\frac{1}{2}\Lambda t]\Phi^T$. Hence, the co-ordinate vector for the node indexed u is

$$y_u = (\exp[-\frac{1}{2}\lambda_1 t]\phi_1(u), \exp[-\frac{1}{2}\lambda_2 t]\phi_2(u), ..., \exp[-\frac{1}{2}\lambda_{|V|} t]\phi_{|V|}(|V|))^T \quad (13)$$

The kernel mapping $\mathcal{M} : V \to \mathcal{R}^{|V|}$, embeds each node on the graph in a vector space $\mathcal{R}^{|V|}$. The heat kernel $h_t = Y^T Y$ can also be viewed as a Gram matrix, i.e. its elements are scalar products of the embedding co-ordinates. Consequently, the kernel mapping of the nodes of the graph is an isometry. The squared Euclidean distance between nodes u and v is given by

$$d_E(u,v)^2 = (y_u - y_v)^T(y_u - y_v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t]\Big\{\phi_i(u) - \phi_i(v)\Big\}^2 \quad (14)$$

Figure 1 shows the steps to embed the graph into a manifold.

3.2 Point Distribution Statistics

One very simple way to characterize the embedded point-set is to study the properties of the covariance matrix of the point-set generated by the embedding methods. To construct the covariance matrix, we commence by computing the mean coordinate vector. The mean co-ordinate vector for the heat kernel embedding is

$$\hat{y} = \frac{1}{|V|} Y e = \frac{1}{|V|} \exp[-\frac{1}{2}\Lambda t]\Phi^T e \quad (15)$$

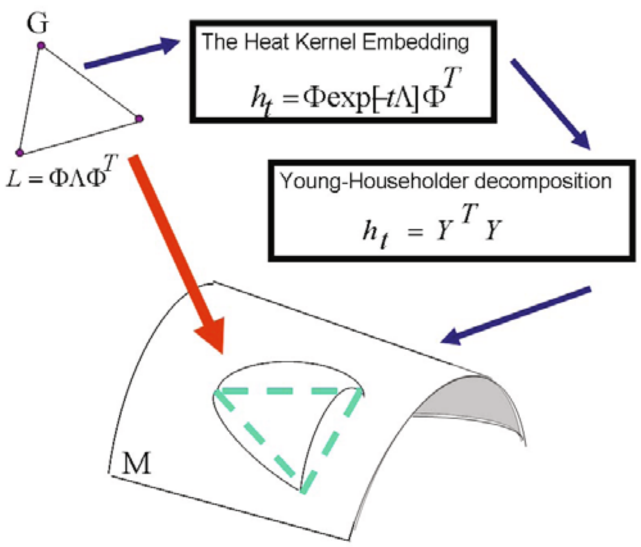

G
The Heat Kernel Embedding
$h_t = \Phi \exp[-t\Lambda]\Phi^T$
Young-Householder decomposition
$h_t = Y^T Y$
$L = \Phi\Lambda\Phi^T$
M

Fig. 1. Illustration of the geometric embedding of the graph into a manifold

where $e = (1, 1, ..., 1)^T$ is the all ones vector of length $|V|$. The matrix of centred co-ordinates is found by subtracting the mean position vector from each of the co-ordinate vectors and is given by

$$Y_C = Y - \frac{1}{|V|}Yee^T = \exp[-\frac{1}{2}\Lambda t]\Phi^T(I - \frac{1}{|V|}ee^T) = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T \quad (16)$$

where $M^T = (I - \frac{1}{|V|}ee^T)$. The covariance matrix for the embedded point-positions is

$$S_Y = Y_C Y_C^T = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T M\Phi \exp[-\frac{1}{2}\Lambda t] \quad (17)$$

Hence, we can write

$$S_Y = \frac{1}{|V|}C^T C$$

where $C = M\Phi \exp[-\frac{1}{2}\Lambda t]$. To compute the eigenvectors of S_Y we first construct the matrix,

$$CC^T = M\Phi \exp[-\Lambda t]\Phi^T M^T = M h_t M^T$$

i.e. CC^T has eigenvalue matrix $\Lambda_h = \exp[-\Lambda t]$ and un-normalised eigenvector matrix $U = M\Phi$. As a result the matrix $C^T C$ has normalised eigenvector matrix $\hat{U} = C^T U \Lambda_h^{-\frac{1}{2}}$ and eigenvalue matrix Λ_h.

To see this note that

$$(C^T U \Lambda_h^{-\frac{1}{2}})\Lambda_h (C^T U \Lambda_h^{-\frac{1}{2}})^T = C^T U U^T C = C^T C \quad (18)$$

Hence $C^T C$ has eigenvector matrix $\Lambda_h = \exp[-\Lambda t]$ and normalised eigenvector matrix

$$\hat{U} = (M\Phi \exp[-\frac{1}{2}\Lambda t])^T M\Phi(\exp[-\Lambda t])^{-\frac{1}{2}} = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T M\Phi \exp[\frac{1}{2}\Lambda t] \tag{19}$$

Finally, it is interesting to note that the projection of the centred co-ordinates onto the eigenvectors of the covariance matrix S_Y is

$$Y_P = \hat{U}^T Y_C = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T = Y_C$$

4 Geometric Characterisation

In this section we develop our differential characterisation of graphs. We commence by showing how the geodesic and Euclidean distances estimated from the spectrum of the Laplacian and the heat kernel embedding can be used to associate a sectional curvature with the edges of a graph. Next, we turn our attention to geodesic triangles formed by the embedding of first order cycles, i.e. triangles, of the graph. From the turning angles of the geodesic triangles, we estimate Gaussian curvature.

4.1 Sectional Curvature

In this section we show how the Euclidean distance and geodesic distances computed for embedding can be used to compute the sectional curvature associated with edges of the graph. The sectional curvature is determined by the degree to which the geodesic bends away from the Euclidean chord. Hence for a torsionless geodesic on the manifold, the sectional curvature can be estimated easily if the Euclidean and geodesic distances are known. Suppose that the geodesic can be locally approximated by a circle. Let the geodesic distance between the pair of points u and v be $d_G(u,v)$ and the corresponding Euclidean distance be $d_E(u,v)$. Further let the radius of curvature of the approximating circle be $r_s(u,v)$ and suppose that the tangent vector to the manifold undergoes a change in direction of $2\theta_{u,v}$ as we move along a connecting circle between the two points. We show an illustration of the above in Figure 2.

In terms of the angle $\theta_{u,v}$, the geodesic distance, i.e. the distance traversed along the circular arc, is $d_G(u,v) = 2r_s(u,v)\theta_{u,v}$, and as a result we find that $\theta_{u,v} = d_G(u,v)/2r_s(u,v)$. The Euclidean distance, on the other hand, is given by $d_E(u,v) = 2r_s(u,v)\sin\theta_{u,v}$, and can be approximated using the MacLaurin series

$$d_E(u,v) = 2r_s(u,v)\{\theta_{u,v} - \frac{1}{6}\theta_{u,v}^3 + ...\} \tag{20}$$

Substituting for $\theta_{u,v}$ obtained from the geodesic distance, we have

$$d_E(u,v) = d_G(u,v) - \frac{d_G(u,v)^3}{24r_s^2(u,v)} \tag{21}$$

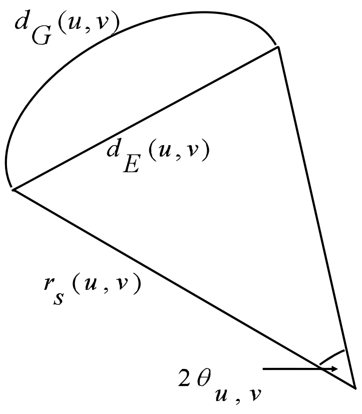


Fig. 2. Illustration of relationship between the geodesic distance, Euclidean distances and the sectional curvature

Solving the above equation for the radius of curvature, the sectional curvature of the geodesic connecting the nodes u and v is approximately

$$k_s(u,v) = \frac{1}{r_s(u,v)} = \frac{2\sqrt{6}(d_G(u,v) - d_E(u,v))^{\frac{1}{2}}}{d_G(u,v)^{\frac{3}{2}}} \tag{22}$$

4.2 The Gauss-Bonnet Theorem

The Gauss-Bonnet Theorem links the topology and geometry of a surface in an elegant and compact manner. Spivak [21] and Stillwell [22] give accounts of the early history of its development and application.

For a smooth compact oriented Riemannian 2-manifold M, let $\triangle_g$ be a triangle on M whose sides are geodesics, i.e. paths of shortest length on the manifold. Further, let α_1, α_2 and α_3 denote the interior angles of the triangle. According to Gauss's theorem, if the Gaussian curvature K (i.e. the product of the maximum and minimum curvatures at a point on the manifold) is integrated over $\triangle_g$, then

$$\int_{\triangle_g} K dM = \sum_{i=1}^{3} \alpha_i - \pi \tag{23}$$

where dM is the Riemannian volume element.

4.3 Gaussian Curvature

To estimate the Gaussian curvature from the above, we must determine the interior angles α_i of the geodesic triangle. To this end we assume that T is a triangulation of a smooth manifold M, $\triangle_g$ be a geodesic triangle on M with angles $\{\alpha_i\}_{i=1}^3$ and geodesic edge lengths $\{d_{gi}\}_{i=1}^3$. Further suppose that $\triangle_e$ be the corresponding Euclidean triangle with edge lengths $\{d_{ei}\}_{i=1}^3$ and interior angles $\{\varphi_i\}_{i=1}^3$. We assume that the geodesic index i is a great arc on a sphere with radius r_i, $i = 1, 2, 3$. By averaging over the constituent geodesic edges, we treat the geodesic triangles as residing on a hyper−sphere with radius $r = \frac{1}{3}\sum_{i=1}^3 r_i$.

To commence, we compute the area of the geodesic triangle. Here we'll make use of the geometry of the sphere, the area of the spherical triangle is given by

$$A_g = (\sum_{i=1}^{3} \alpha_i - \pi) r^2 \tag{24}$$

From (24) we can see that

$$\sum_{i=1}^{3} \alpha_i - \pi = \frac{A_g}{r^2} \tag{25}$$

Now, considering a small area element on the sphere given in spherical coordinates by $dA = R^2 \sin\theta d\theta d\varphi$, the integration of dA bounded by θ gives us another formula for computing the area of the geodesic triangle

$$A = \frac{1}{2r} d_e^2 \tag{26}$$

where d_e^2 is computed from the embedding using (14).

From (23), (25) and (26), we get the following formula for the Gaussian curvature residing over the geodesic triangle:

$$\int_{\triangle_g} K dM = \frac{1}{2r^3} d_e^2 \tag{27}$$

5 Graph Similarity

We represent the graphs using sets of curvatures defined either over the edges (i.e. sectional curvatures) or triangular faces (i.e. Gaussian curvatures) of the graphs under consideration. The sets of curvatures are unordered, i.e. we do not know the correspondences between edges or faces in different graphs, We hence require a set-based similarity measure to compare graphs in the absence of correspondences. One route is provided by the Hausdorff distance. However, this is known to be sensitive to noise, so we explore median and probabilistic variants of the Hausdorff distance.

5.1 Hausdorff Distance

The Hausdorff distance provides a means of computing the distance between sets of unordered observations when the correspondences between the individual items are unknown. In its most general setting, the Hausdorff distance is defined between compact sets in a metric space. Given two such sets, we consider, for each point in one set, the closest point in the second set. Hausdorff distance is the maximum over all these values. More formally, the classical Hausdorff distance(HD)[11] between two finite point sets A and B is given by $H(A,B) = \max(h(A,B), h(B,A))$ where the directed Hausdorff distance from A to B is defined to be

$$h(A,B) = \max_{a\in A} \min_{b\in B} \|a-b\| \tag{28}$$

and $\|.\|$ is some underlying norm on the points of A and B (e.g., the L2 or Euclidean norm). Dubuisson and Jain [7] proposed a robust modified Hausdorff distance (MHD) based on the average distance value instead of the maximum value, in this sense they defined the directed distance of the MHD as

$$h(A,B) = \frac{1}{N_A} \sum_{a\in A} \min_{b\in B} \|a-b\| \tag{29}$$

Using these ingredients we can describe how Hausdorff distances can be extended to graph-based representations. To commence let us consider two graphs $G_1 = (V_1, E_1, k_1)$ and $G_2 = (V_2, E_2, k_2)$, where V_1,V_2 are the sets of nodes, E_1,E_2 the sets of edges and k_1,k_2 the matrices whose elements are the curvature defined in the previous section. We can now write the distances between two graphs as follows:

1) The classical Hausdorff distance (HD) is

$$h_{HD}(G_1, G_2) = \max_{i\in V_1} \max_{j\in V_1} \min_{I\in V_2} \min_{J\in V_2} \|k_2(I,J) - k_1(i,j)\| \tag{30}$$

2) The modified Hausdorff distance (MHD) is

$$h_{MHD}(G_1, G_2) = \frac{1}{|V_1|} \sum_{i\in V_1} (\frac{1}{|V_1|} \sum_{i\in V_1} \min_{I\in V_2} \min_{J\in V_2} \|k_2(I,J) - k_1(i,j)\|) \tag{31}$$

5.2 A Probabilistic Similarity Measure (PSM)

One of the well documented problems with both the Hausdorff and modified Hausdorff distances, is lack of robustness. In order to overcome this problem, Huet and Hancock [10] have recently developed a probabilistic variant of the Hausdorff distance. This measures the similarity of the sets of attributes rather than using defined set based distance measures. For the graphs G_1 and G_2, the set of all nodes connected to the node $I \in G_2$ by an edge is defined as $C_I^2 = \{J|(I,J) \in E_2\}$, and the corresponding set of nodes connected to the node $i \in G_1$ by an edge is $C_i^1 = \{j|(i,j) \in E_2\}$. For the match of the graph G_2 onto G_1 Huet and Hancock's similarity measure is

$$S(G_1, G_2) = \frac{1}{|V_2| \times |V_1|} \sum_{i\in V_1} \max_{I\in V_2} \sum_{j\in C_i^1} \max_{J\in C_I^2} P((i,j) \to (I,J)|k^2_{(I,J)}, k^1_{(i,j)}) \tag{32}$$

In this formula the *a posteriori* probability $P((i,j) \rightarrow (I,J)|k^2_{(I,J)}, k^1_{(i,j)})$ represents the value for the match of the G_2 edge (I,J) onto the G_1 edge (i,j) provided by the corresponding pair of attribute structures $k^2_{(I,J)}$ and $k^1_{(i,j)}$.

The similarity measure commences by finding the maximum probability over the nodes in C^2_I then averaging the edge-compatibilities over the nodes in C^1_i. Similarly, we consider the maximum probability over the nodes in the graph G_2 followed by averaging over the nodes in G_1. It is worth mentioning that unlike the Hausdorff distance, this similarity measure does not satisfy the distance axioms. Moreover, while the Hausdorff distance is saliency-based (i.e. it measures the maximum distance between two sets of observations) our measure here returns the maximum similarity. back to the formula where we still need to compute the probability $P((i,j) \rightarrow (I,J)|k^2_{I,J}, k^1_{i,j})$ For that purpose we will use a robust weighting function

$$P((i,j) \rightarrow (I,J)|k^2_{(I,J)}, k^1_{(i,j)}) = \frac{\Gamma_\sigma(\|k^2_{(I,J)}, k^1_{(i,j)}\|)}{\sum_{(I,J)\in E_2} \Gamma_\sigma(\|k^2_{(I,J)}, k^1_{(i,j)}\|)}) \qquad (33)$$

where $\Gamma_\sigma(.)$ is a distance weighting function. There are several alternative robust weighting functions. Here we work with a Gaussian of the form $\Gamma_\sigma(\rho) = \exp(-\frac{\rho^2}{2\sigma^2})$.

5.3 Multidimensional Scaling

With the graph distances in hand, we require a means of visualizing the distribution of graphs. We choose to use the classical Multidimensional Scaling (MDS) method [5] to embed the data specified in the matrix in Euclidean space. Let H be the distance matrix with row r and column c entry H_{rc}. The first step of MDS is to calculate a matrix T whose element with row r and column c is given by $T_{rc} = -\frac{1}{2}[H^2_{rc} - \widehat{H}^2_{r.} - \widehat{H}^2_{.c} + \widehat{H}^2_{..}]$ where $\widehat{H}_{r.} = \frac{1}{N}\sum_{c=1}^N H_{rc}$ is the average value over the rth row in the distance matrix, $H_.c$ is the similarly defined average value over the cth column and $\widehat{H}_{..} = \frac{1}{N^2}\sum_{r=1}^N \sum_{c=1}^N H_{rc}$ is the average value over all rows and columns of the distance matrix. Then, we subject the matrix T to an eigenvector analysis to obtain a matrix of embedding coordinates X. If the rank of T is k; $k \leq N$, then we will have $k$non-zero eigenvalues. We arrange these k non-zero eigenvalues in descending order, i.e., $l_1 \geq l_2 \geq ... \geq l_k \geq 0$. The corresponding ordered eigenvectors are denoted by u_i where l_i is the ith eigenvalue. The embedding coordinate system for the graphs is $X = [\sqrt{l_1}u_1, \sqrt{l_2}u_2, ..., \sqrt{l_k}u_k]$ for the graph indexed i, the embedded vector of the coordinates is $x_i = (X_{i,1}, X_{i,2}, ..., X_{i,k})^T$.

6 Experiments

In this section we experiment with the curvature-based attributes extracted using the heat-kernel embedding, and explore whether they can be used for the purposes of graph-clustering. In our experiments the graphs extracted from images

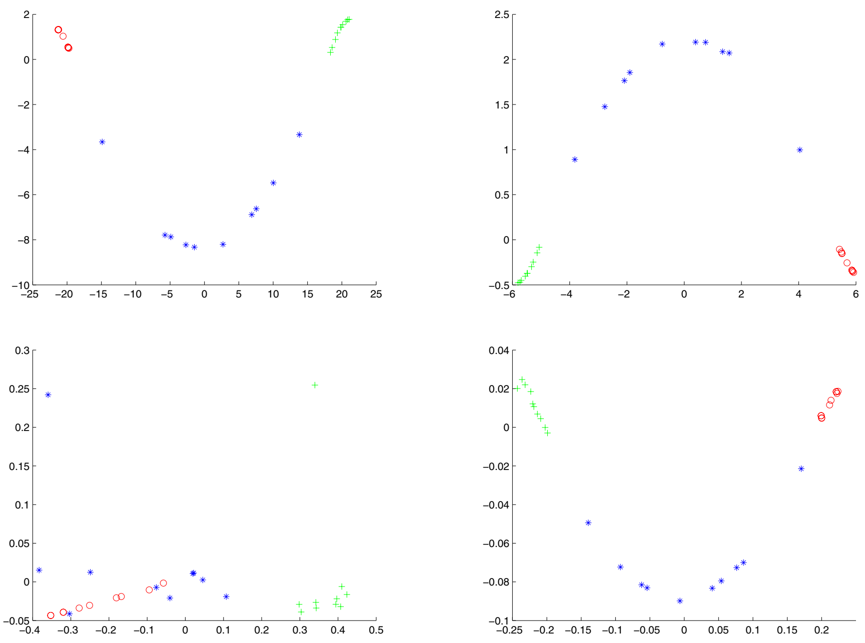


Fig. 3. MDS embedding obtained using MHD for house data represented by the sectional curvatures residing on the edges

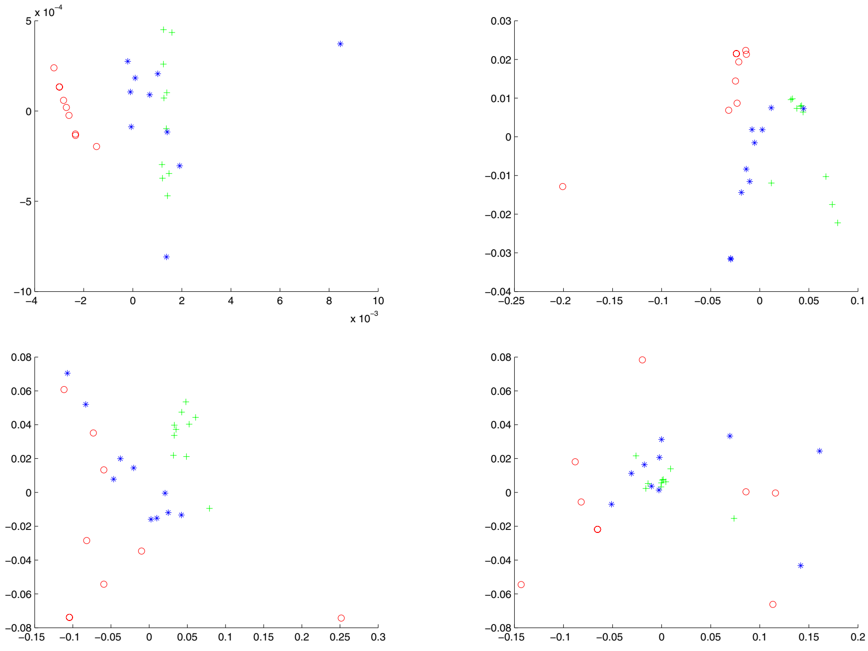


Fig. 4. MDS embedding obtained using MHD for the houses data represented by the Gaussian curvature associated with the geodesic triangles

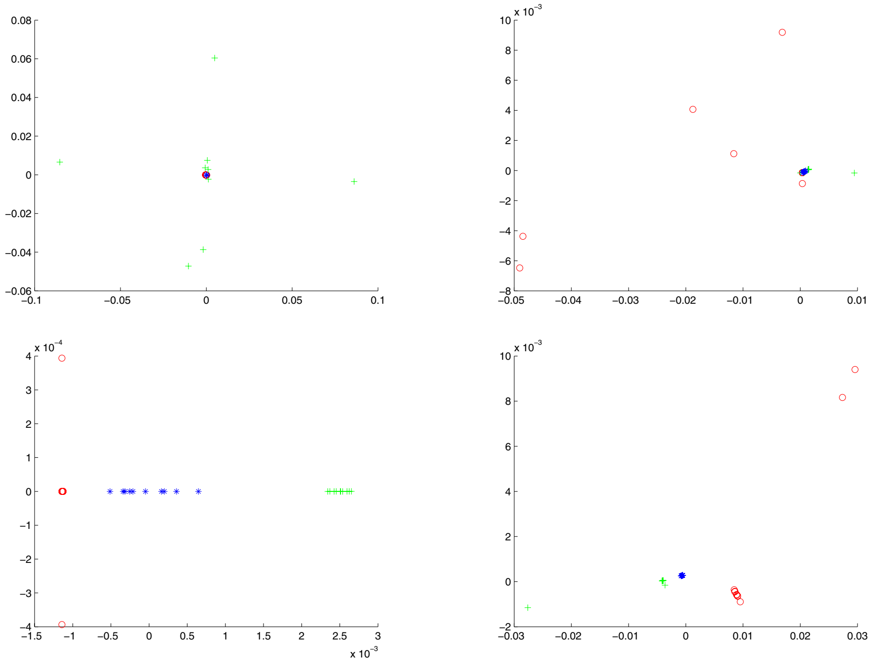


Fig. 5. MDS embedding obtained using the probabilistic similarity measure for the houses data set represented by the sectional curvature residing on the edges

of toy houses in the standard CMU, MOVI and chalet house image sequences [16]. These data sets contain different views of model houses from equally spaced viewing directions. From the house images, corner features are extracted, and Delaunay graphs representing the arrangement of feature points are constructed. Our data consists of ten graphs for each of the three houses. Each node in a Delaunay graph belongs to a first order cycle, and as a result the graph is a triangulation. To commence, we obtain the embedding for each of the thirty graphs following the steps outlined in Section 3.1. We compute the Euclidean distances between the nodes in each graph based on the heat kernel embedding obtained with the values of $t = 10.0, 1.0, 0.1$ and 0.01. We work with two representations of the graphs. The first is the sectional curvature associated with the edges, outlined in Section 4.1. The second is the Gaussian curvature on the triangles of the Delaunay triangulations extracted from the graphs, as outlined in Section 4.3. We use both the sectional and gaussian curvature as features for the purposes of gauging the similarity of graphs using both the modified Hausdorff distance and the probabilistic similarity measure. We subject the distance matrices to the Multidimensional Scaling (MDS) procedure to embed the graphs into a low dimensional space. Figures 3, 4, 5, 6 show the results obtained, where the subfigures are ordered from left to right and from top to bottom, using the heat kernel embedding with the values $t = 10.0, 1.0, 0.1$ and 0.01.

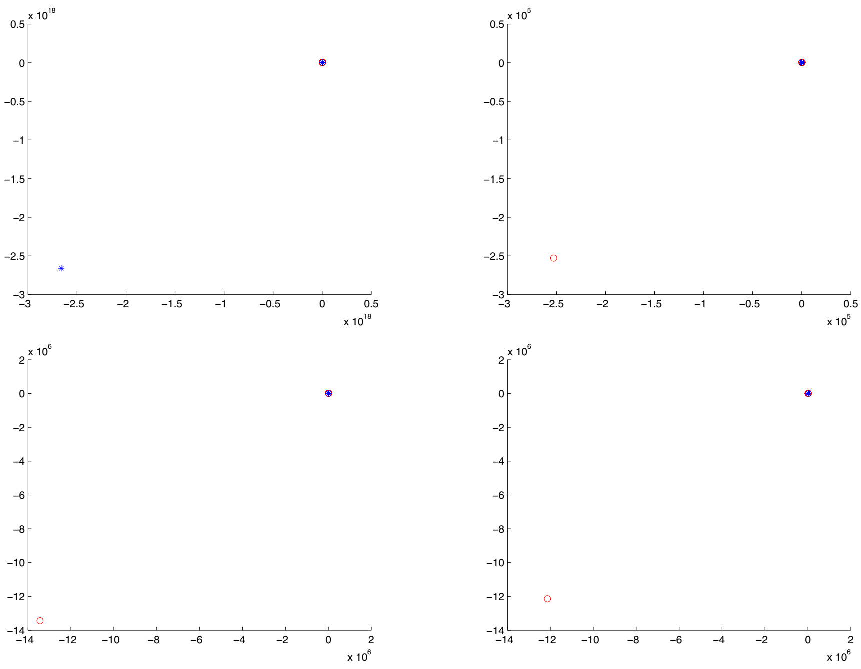


Fig. 6. MDS embedding obtained using the probabilistic similarity measure for the houses data set represented by the Gaussian curvature associated with the geodesic triangles

Table 1. A rand index vs. t

		t=10	t=1.0	t=0.1	t=0.01
MHD	Sectional curvature	0.1333	0.2333	0.1333	0.0333
MHD	Gaussian curvature	0.1667	0.0333	0.1333	0.4000
PSM	Sectional curvature	0.0000	0.0333	0.2667	0.3667
PSM	Gaussian curvature	0.3000	0.3000	0.3000	0.3000

To investigate the data in more detail Table 1 shows the rand index for the data as a function of t. This index is computed as follows: a) We commence by computing the mean for each cluster, b) We then compute the distance from each point to each mean. c) If the distance from the correct mean is smaller than those to remaining means, then the classification is correct, if not then classification is incorrect. d) The rand index is R= (♯ incorrect) / (♯ incorrect + ♯ correct).

A comparison shows that the curvature attributes associated with the edges give slightly better clusters than those obtained using the attributes derived from the triangles.

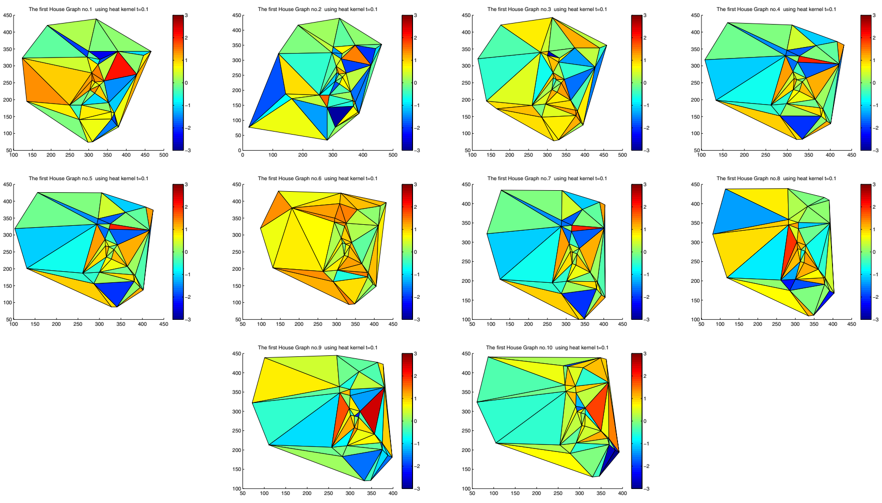


Fig. 7. The distribution of the Gaussian curvatures of the geodesic triangles for the ten graphs of the 1st house

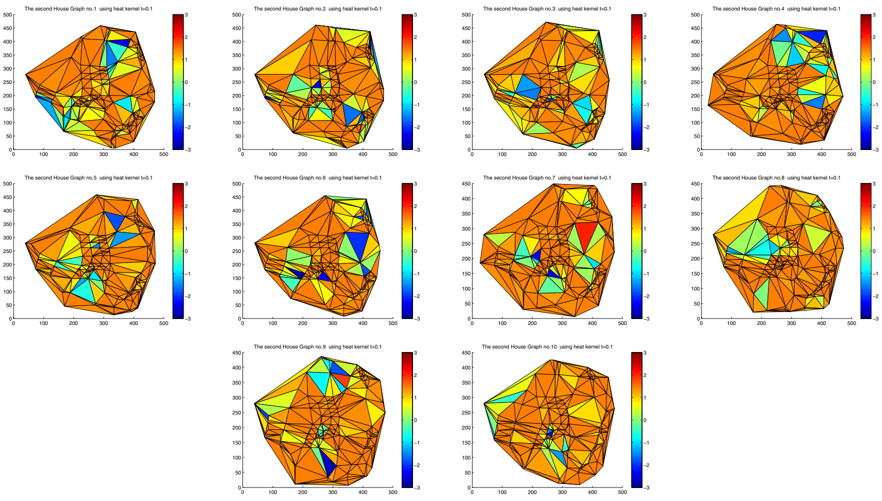


Fig. 8. The distribution of the Gaussian curvatures of the geodesic triangles for the ten graphs of the 2nd house

Finally, we investigate how the Gaussian curvatures of the geodesic triangles are distributed over the Delaunay graph. Figures 7, 8 and 9 show distribution for sample embeddings computed from the heat kernel at $t = 0.1$.

From the sequence it is clear that the Gaussian curvature distribution over the different views of each house is stable, moreover it moves smoothly from

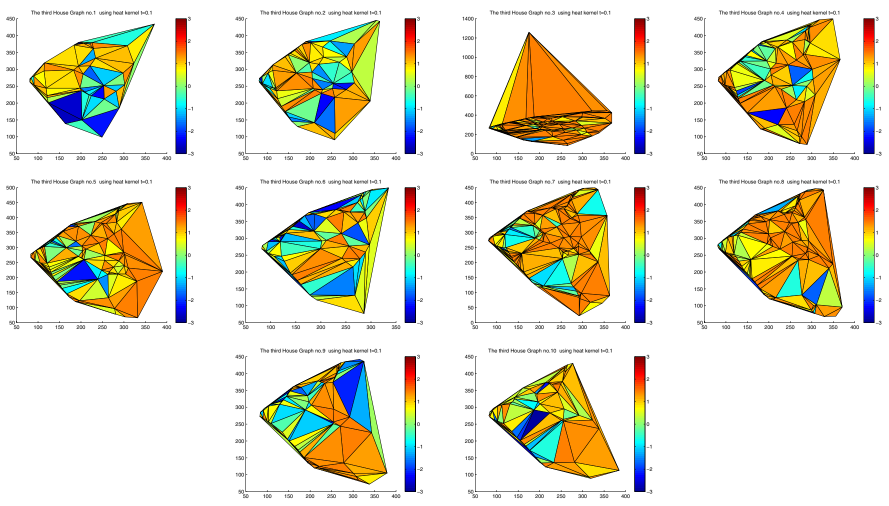


Fig. 9. The distribution of the Gaussian curvatures of the geodesic triangles for the ten graphs of the 3rd house

positive (elliptical) to negative (hyperbolic) regions. This suggests that the arrangement of triangles and their Gaussian curvatures could be used as the basis of a matching algorithm.

7 Conclusion

In this paper we have explored how to use the heat kernel to characterise graphs in a geometric manner. We commence by performing a Young-Householder decomposition on the heat kernel to recover the matrix of embedding co-ordinates. Once embedded we explore a number of alternative ways of characterising the graphs in a geometric manner. These include the sectional curvatures of edges estimated from the geodesic and Euclidean distances between nodes, and the Gaussian curvature of first order cycles. The conclusions of our experimental study is that both characterisations are effective as a means of characterising graphs for the purposes of clustering.

Our future plans revolve around developing ways of controlling noise in the representation. Here we aim to exploit graph-spectral regularisation [27] and curvature-based diffusion methods. We also aim to explore whether the pattern of sectional and Gaussian curvatures can be used to construct pattern spaces for graphs.

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 [13]. A second line of investigation would be the Euclidean distances or sectional curvature associated with the edges as attributes for the purposes of 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. Atkins, J.E., Boman, E.G., Hendrickson, B.: A spectral algorithm for seriation and the consecutive ones problem. SIAM J. Comput. 28(1), 297–310 (1998)
2. Barlow, M.T.: Diffusions on fractals. Lecture Notes Math., vol. 1690, pp. 1–121. Springer, Heidelberg (1998)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, vol. 14 (2002)
4. Chung, F.R.K.: Spectral graph theory. CBMS 92 (1997)
5. Cox, T., Cox, M.: Multidimensional Scaling. Chapman-Hall, Boca Raton (1994)
6. de Verdi'ere, Y.C.: Spectres de graphes. Societe Mathematique De France (1998)
7. Dubuisson, M., Jain, A.: A modified Hausdorff distance for object matching, pp. 566–568 (1994)
8. Gilkey, P.B.: Invariance theory, heat equation, and the index theorem. Mathematics Lecture Series (1984)
9. Grigor'yan, A.: Heat kernels on manifolds, graphs and fractals. European Congress of Mathematics I, 393–406 (2001)
10. Heut, B., Hancock, E.R.: Relational object recognition from large structural libraries. Pattern Recognition 32, 1895–1915 (2002)
11. Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the Hausdorff distance. IEEE. Trans. Pattern Anal. Mach. Intell. 15, 850–863 (1993)
12. Lebanon, G., Lafferty, J.D.: Hyperplane margin classifiers on the multinomial manifold. In: ICML (2004)
13. Lindman, H., Caelli, T.: Constant curvature Riemannian scaling. Journal of Mathematical Psychology 17, 89–109 (1978)
14. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica 15, 215–245 (1995)
15. Luo, B., Hancock, E.R.: Structural graph matching using the EM algorithm and singular value decomposition. IEEE Trans. Pattern Anal. Mach. Intell. 23(10), 1120–1136 (2001)
16. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recogintion 36, 2213–2230 (2003)
17. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290 (5500), 2323–2326 (2000)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE PAMI 22, 888–905 (2000)
19. Shokoufandeh, A., Dickinson, S.J., Siddiqi, K., Zucker, S.W.: Indexing using a spectral encoding of topological structure. In: CVPR, pp. 2491–2497 (1999)
20. Smola, E.J., Kondor, R.: Kernels and regularization on graphs (2004)
21. Spivak, M.: A Comprehensive Introduction to Differential Geometry, 2nd edn., vol. 1-5. Publish or Parish, Houston (1979)
22. Stillwell, J.: Mathematics and its History. Springer, New York (1974)

23. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319 (2000)
24. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE Trans. Patt. Anal. Mach. Intell. 10, 695–703 (1988)
25. Xiao, B., Hancock, E.R.: Trace formula analysis of graphs. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 306–313. Springer, Heidelberg (2006)
26. Yau, S.T., Schoen, R.M.: Differential Geometry. Science Publication Co. (1988) (in Chinese)
27. Zhou, D., Schölkopf, B.: A regularization framework for learning from graph data. In: ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields, pp. 132–137 (2004)
28. Zhu, X., Kandola, J.S., Ghahramani, Z., Lafferty, J.D.: Nonparametric transforms of graph kernels for semi-supervised learning. In: NIPS (2004)

Compressive Algorithms—Adaptive Solutions of PDEs and Variational Problems

M. Fornasier

Johann Radon Institute for Computational and Applied Mathematics, Austria
massimo.fornasier@oeaw.ac.at

Abstract. This paper is concerned with an overview of the main concepts and a few significant applications of a class of adaptive iterative algorithms which allow for dimensionality reductions when used to solve large scale problems. We call this class of numerical methods *Compressive Algorithms*. The introduction of this paper presents an historical excursus on the developments of the main ideas behind compressive algorithms and stresses the common features of diverse applications. The first part of the paper is addressed to the optimal performances of such algorithms when compared with known benchmarks in the numerical solution of elliptic partial differential equations. In the second part we address the solution of inverse problems both with sparsity and compressibility constraints. We stress how compressive algorithms can stem from variational principles. We illustrate the main results and applications by a few significant numerical examples. We conclude by pointing out future developments.

1 Introduction

Compressive Algorithms are a novel approach to efficient adaptive computing and take advantage of the property of solutions of certain PDE's and variational problems to be characterized by few *degrees of freedom*, which are recovered by adaptive nonlinear iterations. The approach to efficient computing via compressive algorithms responds to the need of addressing very large scale problems by means of a dimensionality reduction which requires the solution of a combinatorial optimization. Compressive algorithms can perform optimal complexity solutions since they tend to use the minimal number of degrees of freedom, and are simple to implement. They are already successfully applied in several problems. Their numerical analysis is still very challenging.

Roughly speaking, compressive algorithms are generically formulated as *thresholded gradient iterations*

$$\begin{cases} u^{(n+1)} = H_{\gamma_n} \underbrace{\left(u^{(n)} + L_n^*(f - L_n u^{(n)})\right)}_{\text{gradient step}}, \quad n \in \mathbb{N} \\ L_n \approx L, \text{ for } n \to \infty \\ (\gamma_n)_n \text{ is a sequence of } \textit{shape} \text{ parameters }, \end{cases} \tag{1}$$

to solve exactly, for well-posed problems, or approximatively, for regularized ill-posed problems, the equation

$$Lu = f.$$

Here L and L_n are suitable linear operators acting on the underlying solution space (usually a suitable Hilbert space). They depend on the particular problem at hand. f is the datum of the problem. In several instances L_n is a suitable approximation to a preconditioned version of L. The nonlinear function H_γ acts as a thresholding operator, i.e., *penalizes small/promotes relevant features* of the solution. It depends on the problem, and on the "shape parameter" γ, which controls the thresholding type and level. Let us stress from now that the "relevant features" might not be merely, e.g., large *wavelet coefficients* (see, e.g. [16]), but they can be expressed in terms of more sophisticated representations of the solution. For example, we will consider solutions of degenerate PDE's with discontinuities along curves, and these will be the interesting features to be recovered during the solution process.

The main task in compressive algorithms is to predetermine or to adapt the shape parameters $(\gamma_n)_n$ and the approximations L_n in order to realize the best trade-off between rate of convergence to the solution and complexity.

Compressive algorithms have a long history of important successes. Their first formulations, starting with the concept of *thresholding*, can be traced back to the mid-nineties with the pioneering work of Chambolle, DeVore, Lee, and Lucier [16], Donoho and Johnstone [53,54], Rudin, Osher, and Fatemi [85]. Indeed, on the one side, the design of bases (e.g., curvelets, local Fourier bases, wavelets) for sparse representations of digital signals has led to extremely efficient compression methods, such as JPEG 2000 [78]. Applications in signal de-noising appear in [48,49]. On the other side, degenerate elliptic PDE's for anisotropic diffusion, as appearing in total variation minimization, were also successfully applied for noise removal in digital images, since they essentially perform a suitable thresholding of derivatives and promote few edges. Degenerate PDE's have revolutionized image processing with an enormous impact and consequences [3].

In the late-nineties, the attention moved from the compressibility of signals to the compressibility of functions that are only *implicitly* given as solutions of equations. A new generation of compressive algorithms was proposed by Cohen, Dahmen and DeVore in a sequence of fundamental papers [22,23,24] for the computation of compressed solutions of *elliptic* differential and integral equations, exploiting adaptive and greedy strategies. One of the innovations of their work is the *a priori* analysis of the optimality of such an adaptive scheme. Since solutions can be compressed, hence only few relevant degrees of freedom are sufficient to well-approximate it, one would like to have algorithms that approximate the solution performing a number of algebraic operations which are asymptotically proportional to the number of degrees of freedom of the best approximation. In other words, the complexity of the algorithm is $\mathcal{O}(N)$, where N is the minimal number of degrees of freedom for approximating the solution up to a given accuracy.

Although they were mostly interested in approximations of solutions by means of multiscale bases (e.g., wavelets), their work has recently influenced significantly also the approach to the analysis of adaptive finite element methods and the understanding of their optimal performances [5,90,91]. The latter optimality was previously evaluated only by *a posteriori* numerical tests [79, pag. 634].

The use of the compressibility in more general variational problems for signal recovery and solution of nonlinear equations is the most recent step of this concept's long career of "simplifying and understanding complexity", with an enormous potential in applications [24,25,44,50,77,87,88]. In particular, the observation that it is possible to reconstruct compressible signals from vastly incomplete information just seeking for the total variation or ℓ_1-minimal solutions [9,10,11,51] has led to a new line of research called *sparse recovery* or *compressed sensing*, with very fruitful mathematical and applied results. Again compressive algorithms play a fundamental role in this context [41,45,15,17,96,97].

This historical *excursus* motivates our understanding that these instances of compressing algorithms appearing in different contexts indeed belong to the same *family of numerical methods* with very similar underlying concepts and technical approaches.

We present a non-exhaustive overview of these methods and significant examples of relevant applications. We particularly stress the variational nature of these algorithms, which often can be derived as related to nonlinear *subgradient iterations* in a nonsmooth minimization process. Often such compressive algorithms stem also from multiple-minimization strategies: the initial variational problem is solved by the alternating minimization of an augmented functional with multiple variables. Sometimes this strategy is employed to perform the original minimization in an easier way by introducing useful auxiliary variables, whereas at other times we want to realize clever subspace corrections in order to accelerate the convergence and to reduce the dimensionality of the problem.

We would like to illustrate two kinds of results, without stressing too much the rigor of the presentation. We will include exhaustive references for the reader in search of further details. Not all the references are recalled in the text. We would like instead to convey the main principles, also by illustrating them with numerical examples.

Optimality. The first kind of result is addressed in Section 2 to the optimal performances of compressive algorithms. In particular, we consider the difficult task of realizing optimal approximations to solutions of elliptic operator equations discretized by redundant decompositions (e.g., wavelet frames).

Variational formulation of compressing algorithms. The second kind of result is addressed in Section 3 to the formulation of new compressive algorithms for the solution of variational problems involving sparsity constraints, in particular when the dimension of the problem becomes exaggeratedly large.

In order to support the understanding of the impact for applications, we provide a collection of a few numerical examples for the adaptive solution of PDE's and for image processing.

2 Optimal Adaptive Frame Solvers for Operator Equations

2.1 The State of the Art

As already mentioned in the short introductory overview, compressive algorithms were proposed in order to compute with optimal complexity approximations to solutions of well-posed elliptic operator equations. They are generally realized by iterative adaptive gradient steps, where the matrix-vector multiplications are performed taking advantage of the compressibility of the *stiffness matrices* L resulting by discretizations via suitable multiscale bases (e.g., wavelets), followed by thresholding operations, called *coarsening* in this context,

$$u^{(n+1)} = H_{\gamma_n}\left(u^{(n)} + \beta^{(n)}(f - L_n u^{(n)})\right), \quad \beta^{(n)} > 0, \quad n \in \mathbb{N}. \tag{2}$$

The role of the latter operations is to enforce the elimination of negligible quantities and to select the most relevant coordinates, eventually ensuring the right balance between number of degrees of freedom/complexity and rate of convergence. The coarsening corresponds to a *hard-thresholding* H_{γ_n} with threshold $\gamma_n \to 0$ for $n \to \infty$, see Figure 1 for an illustrative explanation of soft- and hard-thresholding. This means that progressively more and more details of the solution are retained. In practice, the thresholding determines a procedure $\mathbf{COARSE}[v, \varepsilon] = H_{\gamma_\varepsilon}(v)$, such that

$$\|\mathbf{COARSE}[v, \varepsilon] - v\| \le \varepsilon, \quad \varepsilon > 0$$

for $\varepsilon > 0$, and $\mathbf{COARSE}[v, \varepsilon]$ has asymptotically minimal support for $\varepsilon \to 0$. The compressed matrix-vector multiplications $L_n u^{(n)}$ are also realized with the help of a suitable adaptive procedure $\mathbf{APPLY}[v, \varepsilon]$, which determines from a finitely supported v, an asymptotically minimal support vector (for $\varepsilon \to 0$) z_ε such that

$$\|Lv - z_\varepsilon\| \le \varepsilon, \quad \varepsilon > 0.$$

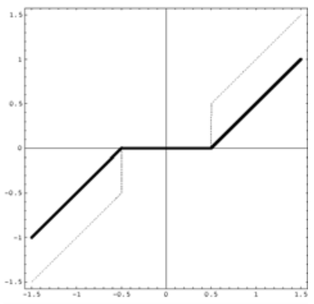


Fig. 1. We depict soft- and hard-thresholding curves, with solid and dashed lines respectively. Small absolute values are mapped to zero, while values exceeding in modulus the threshold γ are slightly diminished and just kept respectively.

In order to promote the minimal complexity in computing the coarsening, also the datum f is compressed by means of a suitable procedure $\mathbf{RHS}[f,\varepsilon]$ which returns an asymptotically minimal support vector (for $\varepsilon \to 0$) such that

$$\|\mathbf{RHS}[f,\varepsilon] - f\| \leq \varepsilon, \quad \varepsilon > 0.$$

The optimal complexity of such procedures is discussed, e.g., in [22,89,92], and they allow for an implementable version of (2):

```
$\mathbf{SOLVE}[\eta,\varepsilon] \to u$:
% Input should satisfy $\eta > 0$ large enough.
% Define the parameters $\alpha_{opt} := \frac{2}{\|L\|+\|L^\dagger\|^{-1}}$ and $\rho := \frac{\kappa(L)-1}{\kappa(L)+1}$.
% Let $\theta$ and $K$ be constants with $2\rho^K < \theta < 1/2$.

$u := 0$;
while $\eta > \varepsilon$ do
    for $j := 1$ to $K$ do
        $u := u + \alpha_{opt}\left(\mathbf{RHS}[f, \frac{\rho^j \eta}{2\alpha K}] - \mathbf{APPLY}[u, \frac{\rho^j \eta}{2\alpha K}]\right)$;
    endfor
    $\eta := 2\rho^K \eta/\theta$;
    $u := \mathbf{COARSE}[u, (1-\theta)\eta]$;
enddo
```

One can also modify this algorithm with a variable *descend parameter* α_η (instead of the prescribed α_{opt}), according to, e.g., a classical *steepest-descent* (SD) criterion, see [34] for more details. The optimality benchmark of **SOLVE** was mostly limited to best approximations with respect to wavelet bases. Unfortunately, the efficient applicability of these methods is spoiled by the crucial problem of constructing well-conditioned boundary adapted bases especially on domains with complicated geometry or manifolds [36,37,38]. The wavelet bases constructed so far exhibit relatively high condition numbers or limited smoothness. The patching used to construct global smooth wavelets by domain decomposition techniques appears complicated and, in most cases, makes the conditioning worse. The global smoothness of the basis, when implementing adaptive schemes in [22,23], is a necessary condition for getting compressibility (i.e., finitely banded approximations) of (infinite) discretization matrices L, especially for high order operators. This bottleneck has led to generalizations of the wavelet approach. These generalizations are based on *frames*, i.e., stable, redundant, non-orthogonal expansions [20,34,40,59,89], which are simpler to construct, hence potentially more attractive to practitioners.

2.2 The Frame Discretization

Frame construction is usually implemented by Overlapping Domain Decompositions (ODD) so no patching at the interfaces is needed to obtain global smoothness. Moreover, the use of frames, due to their intrinsic redundancy, tends to improve the conditioning (meant as the ratio between the largest and the smallest nonzero eigenvalue) of the corresponding discretization matrices and do not spoil their compressibility. Let us recall shortly the general setting.

Let Ω be a Lipschitz domain in $\mathbb{R}^n$, possibly with re-entrant corners. H is a Hilbert space of smooth functions with the following embeddings $H \subset L_2(\Omega) \subset H'$. It is also useful to introduce shortly the notations for sequence spaces. For a countable index set Λ, $0 < p < \infty$, we define the spaces $\ell_p(\Lambda) = \{c = (c_\lambda)_{\lambda \in \Lambda} : \sum_{\lambda \in \Lambda} |c_\lambda|^p < \infty\}$ endowed with the natural (quasi-)norm, and for $p = \infty$ we invoke the usual modification. The operator $\mathcal{L} : H \to H'$ is, e.g., linear, and elliptic. The Laplace operator acting on $H = H_0^1(\Omega)$ is an appropriate example:

$$\begin{aligned} -\triangle u &= f \quad \text{in} \quad \Omega, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{3}$$

The task to perform is to solve adaptively $\mathcal{L}u = f$, for $f \in H'$, when u has limited Sobolev smoothness, whereas high Besov regularity [27,31]. In particular, the assumption is that the solution $u^\star$ has *compressible expansions* $u^\star = \sum_\lambda u_\lambda^\star \psi_\lambda$ (i.e., with a small number of relatively large coefficients u_λ in absolute value) with respect to a frame $\Psi = \{\psi_\lambda\}$ (e.g., wavelets, shearlets) [20,34,40,59,89] constructed, e.g., on overlapping domain decompositions. In particular we choose a frame $\Psi = \{\psi_\lambda\}_{\lambda \in \Lambda}$ for H, i.e., $\|f\|_{H'} \asymp \|\langle f, \Psi\rangle\|_{\ell_2(\Lambda)}$, and $F^* : \ell_2(\Lambda) \to H : c \mapsto c^T\Psi := \sum_{\lambda \in \Lambda} c_\lambda \psi_\lambda$, $\tilde{F} : H \to \ell_2(\Lambda) : u \mapsto \langle u, \tilde{\Psi}\rangle$ are bounded operators, $\tilde{\Psi}$ is a dual frame. With the frame we discretize the problem as follows:

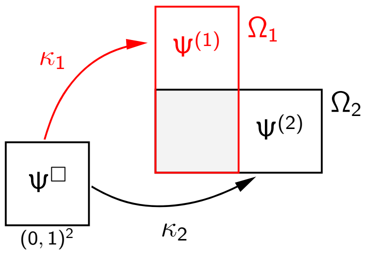
$$\mathcal{L}u^\star = f \Rightarrow Lu^\star = f,$$

$$\begin{cases} L := \langle \mathcal{L}\Psi, \Psi\rangle, \\ f := \langle f, \Psi\rangle. \end{cases}$$

For simplicity and with the hope not to create confusion in this informal discussion, we used here $u^\star$ and f both to indicate the function and the frame coefficients. The brackets $\langle \cdot, \cdot \rangle$ denote the duality between H' and H.

The construction of the frame by means of an overlapping domain decomposition can be schematically described as follows:

For $H = H_0^t(\Omega)$, we fix a smooth reference Riesz basis $\Psi^\Box \subset H_0^t(\Box)$, $\Box := (0,1)^n$, typically a wavelet basis with complementary Dirichlet boundary conditions. Given a suitable overlapping decomposition $\Omega = \sum_{i=1}^K \Omega_i$, and $\kappa_i : \Box \to \Omega_i$, C^m-diffeomorphisms, $m \geq t$, an appropriate lifting yields *aggregated frames* given by $\Psi = \bigcup_{i=1}^K \Psi_i$, see also [34,62,89].



Certainly, an overlapping domain decomposition generates regions of the domain where the side effect of the redundancy is that solutions are no longer uniquely representable by the global frame system. At first sight, it may seem that redundancy contradicts the minimality requirement on the amount of information being used to approximate the solution. Often accurate simulations already require processing a huge amount of data. How can one attempt such computations if the degrees of freedom are also made redundant? A figurative answer to this question is the so-called "dictionary example": The larger and richer is my dictionary the *shorter* are the phrases I compose. The use of the proper terminology avoids long circumlocutions for describing an object. Of course, the key point is the capability of choosing the right terminology. Back to mathematical terms, the combination of adaptivity (i.e., the capability of choosing the right terminology) and redundancy (i.e., the richness or non-uniqueness of representations) can give rise to compressed and accurate approximations.

2.3 Optimality of Adaptive Frame Approximations

Our main contribution in this setting was directed to the difficult estimate of the trade-off between compressibility and adaptivity effort. While for bases the best $N-$term approximation is the optimal benchmark, for frames this concept is not well-defined, due to the non-uniqueness of the expansion. Nevertheless, for frames constructed as a union of bases of the *same* nature (e.g., wavelet bases), as we do by using overlapping domain decompositions, we have to expect that the performances cannot be much better than using a unique global basis. Hence, in these cases, again the best $N-$term approximation with respect to a certain specific representation of the solution constructed via suitable partitions of unity does represent a good benchmark. The core of our work in this direction consisted in the proof of the optimality of compressive algorithms based on wavelet frame discretizations, despite the redundancy [33,32,34,89].

2.4 Numerical Experiments

In order to illustrate the mentioned theoretical results, we would like to present a few concrete numerical examples. For the discretization we use aggregated wavelet frames on suitable overlapping domain decompositions, as the union of local wavelet bases lifted to the subdomains. As such local bases we use piecewise linear and piecewise quadratic wavelets with complementary boundary conditions from [36], with *order of polynomial exactness* $d = 2$ and with $\tilde{d} = 2$ *vanishing moments.* In particular, we impose here homogenous boundary conditions on the primal wavelets and free boundary conditions on the duals. We test the algorithms on both 1D and 2D Poisson problems.

2.5 Poisson Equation on the Interval

We consider the variational formulation of the following problem of second order on the interval $\Omega = (0, 1)$, i.e., $n = 1$, with homogenous boundary conditions

$$-u'' = f \quad \text{on } \Omega, \quad u(0) = u(1) = 0. \tag{4}$$

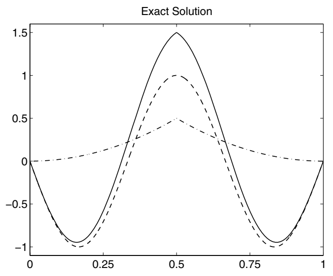


Fig. 2. Exact solution (solid line) for the one–dimensional example being the sum of the dashed and dash–dotted functions

The right-hand side f is given as the functional defined by $f(v) := 4v(\frac{1}{2}) + \int_0^1 g(x)v(x)dx$, where

$$g(x) = -9\pi^2 \sin(3\pi x) - 4.$$

The solution is consequently given by

$$u(x) = -\sin(3\pi x) + \begin{cases} 2x^2 & , \quad x \in [0, \frac{1}{2}) \\ 2(1-x)^2, & x \in [\frac{1}{2}, 1] \end{cases},$$

see Figure 2. As an overlapping domain decomposition we choose $\Omega = \Omega_1 \cup \Omega_2$, where $\Omega_1 = (0, 0.7)$ and $\Omega_2 = (0.3, 1)$. Associated to this decomposition we construct our aggregated wavelet frames just as the union of the local bases. It is shown in [33,32,89] that such a system is a (Gelfand) frame for $H_0^t(\Omega)$ and that it can provide a suitable characterization of Besov spaces in terms of wavelet coefficients.

On the one hand, the solution u is contained in $H_0^{s+1}(\Omega)$ only for $s < \frac{1}{2}$. This means that linear Galerkin methods can only converge with limited order. On the other hand, it can be shown that $u \in B_\tau^s(L_\tau(\Omega))$ for any $s > 0$, $1/\tau = s - 1/2$ [27,31], so that the wavelet frame coefficients u_λ associated with u define a sequence in ℓ_τ^w for any $s < \frac{d-t}{n}$ (see [47,89]). This ensures that the choice of wavelets with suitable order d can allow for any order of convergence in adaptive schemes like that presented in this paper, in the sense that the error is $O(N^{-s})$ where N is the number of unknowns. Due to our choice of piecewise linear wavelets with order $d = 2$, the optimal rate of convergence is expected to be $s = \frac{d-t}{n} = 1$. We show that the numerical experiments confirm this expected rate.

We tested the adaptive wavelet algorithm **SOLVE** with adaptive choice of the parameter α_η according to the steepest descent (SD) rule, and with parameter $\alpha_{opt} \approx 0.52$, $\theta = 2/7$, $K = 83$, and with initial $\eta = 64.8861$. The numerical results in Figure 3 illustrate the optimal computational complexity of the two implementations. In particular, we show that **SOLVE** with steepest-descent parameter rule outperforms a suboptimal choice of the damping parameter

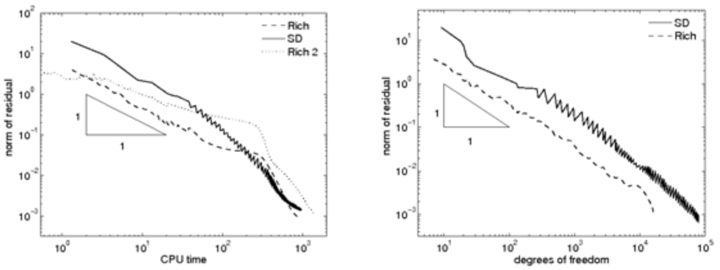


Fig. 3. *Left:* Convergence history of **SOLVE** with respect to CPU time. Tests for **SOLVE** with different damping parameters are shown. *Right:* Convergence history with respect to the support size of the iterands.

$\Omega_1 = (0, 0.7)$ and $\Omega_2 = (0.3, 1)$

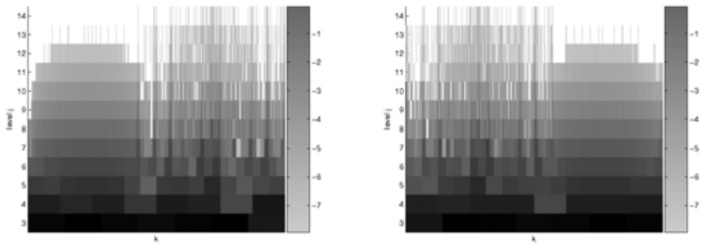


Fig. 4. Distribution of active wavelet frame elements in Ω_1 and Ω_2

($\alpha^* = 0.2 \leq \alpha_{opt} \approx 0.52$ in this specific test). In practice, the wrong guess of the damping parameter can even spoil convergence and/or optimality. Finally, Figure 4 illustrates the distribution of the active wavelet frame elements used by the steepest descent scheme, each of them corresponding to a colored rectangle. The two overlapping subintervals are shown separately. For both patches one observes that the adaptive scheme detects the singularity of the solution. The chosen frame elements arrange in a tree–like structure with large coefficients around the singularity, while on the smooth parts the coefficients are uniformly distributed, and along a fixed level they are of similar size here.

2.6 Poisson Equation on the *L*-Shaped Domain

We consider the model problem of the variational formulation of Poisson's equation in two spatial dimensions:

$$-\Delta u = f \text{ in } \Omega, \quad u_{|\partial\Omega} = 0. \tag{5}$$

The problem will be chosen in such a way that the application of *adaptive* algorithms pays off most, as is the case for domains with reentrant corners. Here,

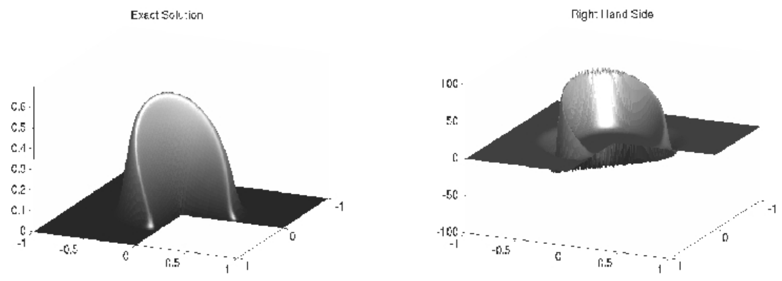


Fig. 5. Exact solution (left) and right–hand side for the two–dimensional Poisson equation in an L–shaped domain

the reentrant corners themselves lead to singular parts in the solutions, forcing them to have a limited Sobolev regularity, even for smooth right–hand sides f. We use

$$\mathcal{S}(r,\theta) := \zeta(r) r^{2/3} \sin\left(\frac{2}{3}\theta\right),$$

as the exact solution, which is shown together with the corresponding right–hand side in Figure 5. It is well-known that $\mathcal{S} \in H^s(\Omega)$ for $s < 5/3$ only, but it is contained in every Besov space $B^s_\tau(L_\tau(\Omega))$, where $s > 0$, $1/\tau = (s-1)/2 + 1/2$ (see [27]).

As has been previously noted, the convergence rate of a uniform refinement strategy is determined by the Sobolev regularity of the solution, while in the context of adaptive schemes it depends on the Besov regularity (cf. [28]). In particular, considering piecewise quadratic approximation, the best possible convergence rate in the $H^1(\Omega)$-norm for uniform refinement strategies is $\mathcal{O}(N^{-(\frac{5}{3}-1)/2})$, with N being the number of unknowns, whereas our adaptive frame scheme gives the optimal rate $\mathcal{O}(N^{-1})$.

For our numerical experiments, we use an aggregated wavelet frame. With $\Omega_1 = (-1,0) \times (-1,1)$, $\Omega_2 = (-1,1) \times (-1,0)$, and $\square = (0,1)^2$, let κ_i be affine bijections between $\square$ and Ω_i $(i = 1,2)$. For $\Psi^\square$ being a piecewise quadratic wavelet basis for $H^1_0(\square)$, where $d = 3$ and $\tilde{d} = 5$, we set $\Psi = \cup_{i=1}^2 \kappa_i(\Psi^\square)$.

In Figure 6 we show some of the approximations and the corresponding pointwise differences to the exact solution produced by our steepest descent scheme using piecewise quadratic frame elements. The numerical results in Figure 7 illustrate the optimal convergence of the scheme with different descent parameter rules. Figure 8 shows similar results by using improved bases [32] with $d = \tilde{d} = 3$.

2.7 Towards Domain Decomposition Methods

Figure 4 is very useful in order to highlight a fundamental behavior of **SOLVE**. In the overlapping region nearly the same wavelet coefficients related to both patches are simultaneously activated. This means that these algorithms are unable to eliminate the frame redundancy and to approximate the sparsest

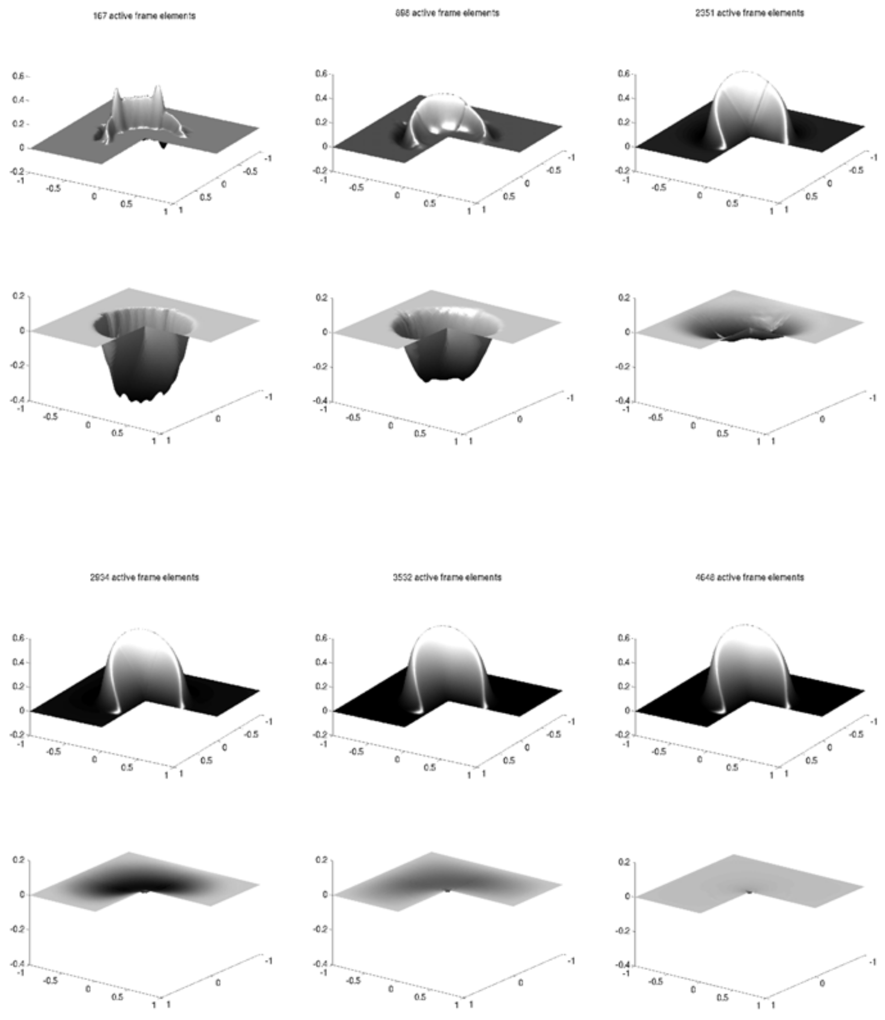


Fig. 6. Approximations and corresponding pointwise errors produced by the adaptive steepest descent algorithm, using piecewise quadratic frame elements. *Upper part:* Approximations with 167, 898, and 2351 frame elements. *Lower part:* Approximations with 2934, 3532, and 4648 frame elements.

representation of the solution. This problem can be solved by realizing a coarsening that does not just threshold coefficients but really promotes very sparse representations. However, such a procedure needs to implement a basis pursuit strategy which can be computationally expensive (see the next section). A more promising direction is an adaptive implementation of a Schwarz alternating method, as proposed, e.g., in [83,84],

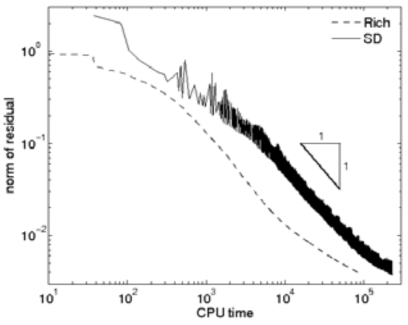


Fig. 7. Convergence histories of **SOLVE** and **CDD2SOLVE** with respect to CPU time using piecewise quadratic frame elements ($d = 3$, $\tilde{d} = 5$)

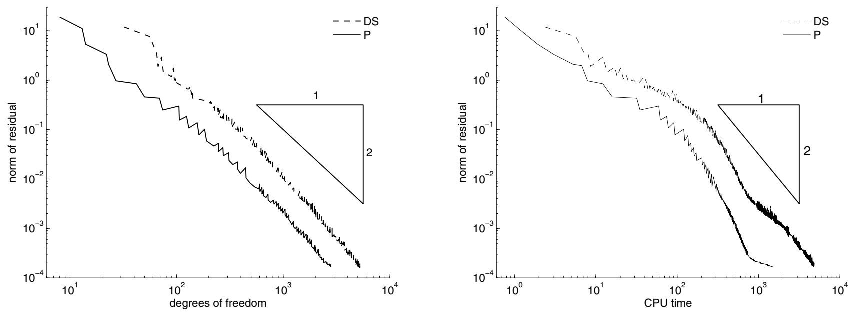


Fig. 8. Convergence histories of the adaptive steepest descent method with respect to the support size of the iterates *(left column)* or CPU time *(right column)* for $d = \tilde{d} = 3$. The algorithm has been tested with aggregated frames based on interval bases from [3] (solid line) and [36] (dashed line).

Algorithm 1. *Multiplicative Schwarz iteration*

```
for $k = 1, \ldots,$
  $u_0^{k-1} = u^{k-1}$
  for $i = 1, \ldots, n$
    $u_i^{k-1} = u_{i-1}^{k-1} + Q_i^T \tilde{L}_i^{-1} Q_i (f - L u_{i-1}^{k-1})$
  endfor
  $u^k = u_n^{k-1}$
endfor
```

Algorithm 2. *Additive Schwarz iteration*

```
for $k = 1, \ldots,$
  $u^k = u^{k-1} + \alpha \sum_{i=1}^{n} Q_i^T \tilde{L}_i^{-1} Q_i (f - L u^{k-1})$
endfor
```

where $0 < \alpha \leq 1$ is a suitable damping parameter and Q_i are suitable projections onto coordinate subspaces and $\tilde{L}_i^{-1}$ is an approximation of a local inverse of the operator L.

A recent exploration of this idea [93] confirms that an adaptive domain decomposition strategy successfully promotes sparser solutions, hence it is computationally far more advantageous, and it is also proven to be optimal. In particular, the wavelet coefficients activated in one patch contribute to the datum of the problem to be solved on the other patch, and are not going to re-activated again there. Presently, this is the best approach to adaptive numerical solution of elliptic PDEs by means of wavelet discretizations.

3 Sparse Recovery and Variational Problems

3.1 ℓ_1−Minimization and Iterative Soft-Thresholding Algorithms

The minimization of the functional

$$\mathcal{J}(u) := \|Lu - f\|^2_{\ell_2(\Lambda)} + 2\gamma \|u\|_{\ell_1(\Lambda)}, \quad \gamma > 0 \tag{6}$$

proved to be an extremely efficient alternative to the well-known Tikhonov regularization [60], whenever

$$Lu = f,$$

is an ill-posed problem on $\ell_2(\Lambda)$ and the solution u is expected to be a vector with a moderate number of nonzero entries. Indeed, the imposition of the ℓ_1−constraint does promote a sparse solution. The restriction to problems on $\ell_2(\Lambda)$ is just formal, this can always be achieved by discretization via a suitable frame for the underlying Hilbert space, as already shown in the previous section. The use of the ℓ_1−norm as a sparsity-promoting functional can be found first in reflection seismology and in deconvolution of seismic traces [21,86,95]. In the last decade more understanding of the deep motivations why ℓ_1−minimization tends to promote sparse recovery was developed. Rigorous results began to appear in the late-1980's, with Donoho and Stark [56] and Donoho and Logan [55]. Applications of ℓ_1−minimization in statistical estimation began in the mid-1990's with the introduction of LASSO, which stands for Least Absolute Shrinkage and Selection Operator, cf. [96] for details[1]. In signal processing, Basis Pursuit [19] was proposed in compression applications for extracting the sparsest signal representation from highly overcomplete frames. From these early steps the applications and understanding of compressive ℓ_1−minimization have continued to increase dramatically. It is now hard to trace all the relevant results and applications [8,12,9,51,52], and it is beyond the scope of this short summary[2]. In fact, ℓ_1-minimization has been so surprisingly effective in several applications, that

[1] `http://www-stat.stanford.edu/~tibs/lasso.html`

[2] The reader can also find a sufficiently comprehensive collection of the ongoing recent developments at the web-site `http://www.compressedsensing.com/`

Candès, Wakin, and Boyd call it the "modern least squares" in [13]. We thus clearly need efficient algorithms for the minimization of $\mathcal{J}$.

An iterative thresholding algorithm, realized by Richardson-Landweber steps followed by a *soft-thresholding* H_γ (Figure 1), was proposed for this task [26,41,45,87,96],

$$u^{(n+1)} = H_\gamma \left(u^{(n)} + L^*(f - Lu^{(n)}) \right), \quad n \in \mathbb{N}, \tag{7}$$

with a *fixed* threshold parameter $\gamma > 0$.

Unfortunately, despite its simplicity which makes it very attractive to users, this algorithm does not perform very well, as it has been recently verified systematically, e.g., in [76] where a comparison with several other alternative methods [4,43,61,75] is carefully provided. However, since the mentioned methods are formulated as a sequential algorithm, none of them is able to address in real-time, or at least in an acceptable computational time, extremely large problems, such as 4D imaging (spatial plus temporal dimensions) from functional magnetic-resonance in nuclear medical imaging, astronomical imaging or global terrestrial seismic tomography. For this reason, and parallel to the development of very successful approaches for well-posed problems [93], a "domain decomposition" algorithm was proposed in [63] together with its parallelization, and generalizations to general subspace corrections appear in [68].

We briefly illustrate below the general setting of the results in [43,63,68].

3.2 Accelerated Projected Gradient Methods

A concrete recipe to identify adaptively *good* shape parameters $\gamma = (\gamma_n)_n$ in iterative thresholding algorithms is by construction of a suitable convex set K and a projection map P_K such that $P_K(u) = H_\gamma(u)$ for an adaptive $\gamma := \gamma(K, u)$. In the case of the specific functional (6) the set K is an ℓ_1-ball, while H_{γ_n} is the soft-thresholding operator. This leads to the following accelerated projected gradient/steepest descent iteration, that turns out to be in several situations [76] much faster than (7):

$$\begin{cases} u^{(n+1)} = P_K(u^{(n)} + L_n^*(f - L_n u^{(n)})), \\ L_n \to L, \end{cases} \tag{8}$$

Note that, by definition of P_K, this iteration corresponds to a compressive algorithm where H_{γ_n} is adapted at each iteration.

Theorem 1. *The sequence* $\left(u^{(n)}\right)_{n\in\mathbb{N}}$ *as defined in* (8)*, where the operator approximation* $L_n^* L_n$ *is possibly chosen according to a suitable* steepest-descent criterion*, converges in norm to a minimizer of* $\mathcal{D}(u) = \|Lu - f\|^2_{\ell_2(\Lambda)}$ *on* K.

For the theoretical analysis and a discussions on the performance of this algorithm we refer the reader to [43]. Let us just include an example inspired by a real-life application in geoscience [77], in particular an application in seismic tomography based on earthquake data. The solution space consists of the wavelet

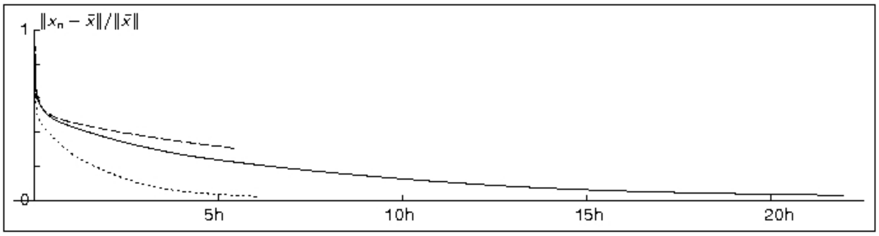


Fig. 9. The different convergence rates of the thresholded Landweber algorithm (solid line), the projected Landweber algorithm (dashed line), where no adaptive descent parameters are used, and the projected steepest descent algorithm (dotted line), for the third example. The projected steepest descent algorithm converges about four times faster than the thresholded Landweber iteration. The projected Landweber iteration does better at first (not visible in this plot), but loses with respect to iterative thresholding afterwards. The horizontal axis has time (in hours), the vertical axis displays the relative error.

coefficients of a 2D seismic velocity perturbation. There are 8192 degrees of freedom. In this particular case the number of data is 1848. Hence the matrix L has 1848 rows and 8192 columns. We apply the different methods to the same noisy data that are used in [77] and measure the time to convergence up to a specified relative error (see Figure 9). This example illustrates the slow convergence of the thresholded Landweber algorithm (7), and the improvements made by a projected steepest descent iteration (8).

3.3 Domain Decomposition Methods

A different strategy in order to deal with very large problems is based on suitable domain decomposition methods, as analyzed in [63]. By splitting disjointly the index set $\Lambda = \Lambda_1 \cup \Lambda_2$, alternating (sequential and parallel) algorithms can be formulated:

$$\begin{cases} u_{\Lambda_1}^{(n+1,\ell+1)} = H_\gamma(u_{\Lambda_1}^{(n+1,\ell)} + L_{n,\Lambda_1}^*((f - L_{n,\Lambda_2} u_{\Lambda_2}^{(n,M)}) - L_{n,\Lambda_1} u_{\Lambda_1}^{(n+1,\ell)})), \\ \ell = 0, \dots, L-1, \\ u_{\Lambda_2}^{(n+1,m+1)} = H_\gamma(u_{\Lambda_2}^{(n+1,m)} + L_{n,\Lambda_2}^*((f - L_{n,\Lambda_1} u_{\Lambda_1}^{(n+1,L)}) - L_{n,\Lambda_2} u_{\Lambda_2}^{(n+1,m)})), \\ m = 0, \dots, M-1. \\ u^{(n+1)} = u_{\Lambda_1}^{(n+1,L)} + u_{\Lambda_2}^{(n+1,M)}. \end{cases} \tag{9}$$

Here we denoted by u_{Λ_i} any vector supported on Λ_i, and by L_{n,Λ_i} the submatrix of L_n where only the columns corresponding to entries in Λ_i are considered. The latter algorithm is derived as an instance of the following alternating minimization: Pick an initial $u_{\Lambda_1}^{(0,L)} + u_{\Lambda_2}^{(0,M)} := u^{(0)} \in \ell_2(\Lambda)$, for example $u^{(0)} = 0$, and iterate

$$\begin{cases} u_{\Lambda_1}^{(n+1,L)} \approx \arg\min_{\operatorname{supp}(v_1)\subset\Lambda_1} \mathcal{J}(v_1 + u_{\Lambda_2}^{(n,M)}) \\ u_{\Lambda_2}^{(n+1,M)} \approx \arg\min_{\operatorname{supp}(v_2)\subset\Lambda_2} \mathcal{J}(u_{\Lambda_1}^{(n+1,L)} + v_2) \\ u^{(n+1)} := u_{\Lambda_1}^{(n+1,L)} + u_{\Lambda_2}^{(n+1,M)}. \end{cases}$$

Theorem 2. *The algorithm in* (9) *produces a sequence* $(u^{(n)})_{n\in\mathbb{N}}$ *in* $\ell_2(\Lambda)$ *whose strong accumulation points are minimizers of the functional* $\mathcal{J}$*. In particular, the set of strong accumulation points is non-empty.*

3.4 Subspace Correction Methods

Actually, the functional (6) is the prototype model of more general problems, that we shall now describe. Let $\mathcal{H}$ be a real separable Hilbert space. We are interested in the numerical minimization in $\mathcal{H}$ of the general form of functionals

$$\mathcal{J}(u) := \|Lu - f\|_{\mathcal{H}}^2 + 2\gamma\psi(u), \tag{10}$$

where $L \in \mathcal{L}(\mathcal{H})$ is a bounded linear operator, $f \in \mathcal{H}$ is a datum, and $\gamma > 0$ is a fixed constant. The function $\psi : \mathcal{H} \to \mathbb{R}_+ \cup \{+\infty\}$ is a semi-norm for a suitable subspace $\mathcal{H}^\psi$ of $\mathcal{H}$. An example of this setting is certainly (6). In particular, we investigate splittings into arbitrary orthogonal subspaces $\mathcal{H} = V_1 \oplus V_2$ for which we may have

$$\psi(\pi_{V_1}(u) + \pi_{V_2}(v)) \neq \psi(\pi_{V_1}(u)) + \psi(\pi_{V_2}(v)), \quad u, v \in \mathcal{H},$$

where π_{V_i} is the orthogonal projection onto V_i.

With this splitting we can address the minimization of $\mathcal{J}$ by suitable instances of the following alternating algorithm: Pick an initial $V_1 \oplus V_2 \ni u_1^{(0)} + u_2^{(0)} := u^{(0)} \in \mathcal{H}^\Psi$, for example $u^{(0)} = 0$, and iterate

$$\begin{cases} u_1^{(n+1)} \approx \arg\min_{v_1\in V_1} \mathcal{J}(v_1 + u_2^{(n)}) \\ u_2^{(n+1)} \approx \arg\min_{v_2\in V_2} \mathcal{J}(u_1^{(n+1)} + v_2) \\ u^{(n+1)} := u_1^{(n+1)} + u_2^{(n+1)}. \end{cases}$$

This algorithm is implemented by solving the subspace minimizations via an *oblique thresholding* iteration, which is defined by means of Lagrange multipliers. The attribute "*oblique*" emphasizes the presence of a fixed additional subspace that contributes to the computation of the thresholded solution.

In [68] we provide a detailed analysis of the convergence properties of this sequential algorithm and of its modification for parallel computation. Extensions to nonorthogonal decompositions $\mathcal{H} = V_1 + V_2$ are reported in [64][3].

We want to mention here a special application of the results above towards domain decomposition methods for total variation minimization and the solution of its associated degenerate elliptic PDE's. As we have already shown in the

[3] Matlab software and the numerical experiments are provided at `http://homepage.univie.ac.at/carola.schoenlieb/webpage_vdode/tv_dode_numerics.htm`

previous section dedicated to the adaptive solution of well-posed elliptic problems, domain decomposition methods were introduced as techniques for solving partial differential equations based on a decomposition of the spatial domain of the problem into several subdomains. The initial equation restricted to the subdomains defines a sequence of new local problems. The main goal is to solve the initial equation via the solution of the local problems. This procedure induces a dimension reduction which is the major reason for the success of such a method. Indeed, one of the principal motivations is the formulation of solvers which can be easily parallelized.

We can apply the theory and the algorithms described above to adapt domain decompositions to the minimization of functionals with total variation constraints. In this case the interesting solutions are usually discontinuous, e.g., along curves in 2D. These discontinuities may cross the interfaces of the domain decomposition patches. Hence, the crucial difficulty is the correct treatment of interfaces, with the preservation of crossing discontinuities and the correct matching where the solution is continuous instead. We consider the minimization of the functional $\mathcal{J}$ in the following different setting: Let $\Omega \subset \mathbb{R}^n$, for $n = 1, 2$, be a bounded open set with Lipschitz boundary. We are interested in the case when $\mathcal{H} = L^2(\Omega)$, $\mathcal{H}^\psi = BV(\Omega)$ and $\psi(u) = |Du|(\Omega)$, the variation of u. Then a nonoverlapping domain decomposition $\Omega = \Omega_1 \cup \Omega_2$ induces the space splitting into $V_i := \{u \in L^2(\Omega) : \operatorname{supp}(u) \subset \Omega_i\}$, $i = 1, 2$. For overlapping domain decomposition methods in this setting, we refer the reader to [64]. Hence, by means of the proposed alternating algorithm, we can address the minimization of the functional

$$\mathcal{J}(u) := \|Tu - g\|^2_{L^2(\Omega)} + 2\alpha |Du|(\Omega).$$

It is important to mention that several numerical strategies to perform efficiently total variation minimization have been proposed in the literature. We list a few of the relevant ones, ordered by their chronological appearance:

(i) the approach of Chambolle and Lions [17] by re-weighted least squares, see also [42] for generalizations and refinements in the context of compressed sensing;

(ii) variational approximation via local quadratic functionals as in the work of Vese et al. [97,3];

(iii) iterative thresholding algorithms based on projections onto convex sets as in the work of Chambolle [15] as well as in the work of Combettes-Wajs [26] and Daubechies et al. [45];

(iv) iterative minimization of the Bregman distance as in the work of Osher et al. [82];

(v) the approach proposed by Nesterov [81] and its modifications by Weiss et al. [98].

These approaches differ significantly, and it seems that the ones collected in the groups iv) and v) do show presently the best performances in practice. However, being formulated as sequential algorithms, none of the mentioned methods is able to address in real-time, or at least in an acceptable computational time,

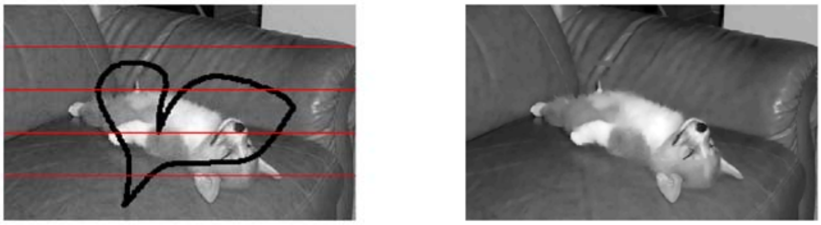


Fig. 10. An example of the solution of *total variation inpainting* (a nonlinear interpolation of an image with a missing part to be recovered) by means of the domain decomposition method [68]. The problem is split into 5 subdomains and it is correctly solved, providing a global minimization.

extremely large problems, and domain decomposition strategies are fundamental in such cases.

3.5 Joint Sparsity and Iterative *Firm-Thresholding* Algorithms

Actually, total variation and ℓ_1-constraints are of course not the only ways of promoting sparse solutions. In [66,67] we investigated more general forms of functionals that not only generate different types of compressive algorithms, but also deal with vector-valued solutions with components coupled by identical sparsity patterns.

Joint sparsity naturally occurs, for instance, in color images, where, e.g., the three color channels RGB can usually be well approximated by a jointly sparse wavelet or curvelet expansion since edges appear at the same locations throughout all channels. However, the range of applicability of this approach is not limited to color image restoration. Multimodal brain imaging, distributed networks and compressed sensing, and several other problems with coupled vector valued solutions are fields where one can expect fruitful applications. In this more general context we are given a Hilbert space $\mathcal{H}$, and we assume that L is a linear operator mapping $\ell_2(\Lambda)^M$ into $\mathcal{H}$ where $M \in \mathbb{N}$ indicates the number of vector components of the solution. We consider the functional

$$\begin{aligned} J(u,v) = J^{(q)}_{\theta,\rho,\omega}(u,v) \; &:= \\ &\|Lu-f\|^2_{\mathcal{H}} + \sum_{\lambda\in\Lambda} v_\lambda \|u_\lambda\|_q + \sum_{\lambda\in\Lambda} \omega_\lambda \|u_\lambda\|_2^2 + \sum_{\lambda\in\Lambda} \theta_\lambda(\rho_\lambda - v_\lambda)^2, \end{aligned} \tag{11}$$

where $\|\cdot\|_q$ denotes the usual ℓ_q-norm on $\mathbb{R}^M$, $q \in [1,\infty]$ and $\theta = (\theta_\lambda)$, $\omega = (\omega_\lambda)$, $\rho = (\rho_\lambda)$ are suitable sequences of positive parameters. The variable u is assumed to be in $\ell_2(\Lambda)^M$ and $v_\lambda \geq 0$ for all $\lambda \in \Lambda$. Observe, that u_λ is a vector in $\mathbb{R}^M$.

We are interested in the joint minimizer (u^*, v^*) of this functional, u^* is then considered the optimal solution. The variable v is an auxiliary variable that plays the role of an indicator of the sparsity pattern. As argued in [66] J promotes *joint* *sparsity*, i.e., $u^* = (u^{*(\ell)})_{\ell=1}^M$ can be expected to be jointly sparse, $\operatorname{supp}(u^{*(\ell)}) \subset \Lambda_0$, for all $\ell = 1, \dots, M$, and for a fixed $\Lambda_0 \subset \Lambda$, $\#\Lambda_0 < \infty$.

In [66] we proposed an iterative algorithm for computing the minimizer of $J(u,v)$. It consists of alternating a minimization with respect to u and v. More formally, for some initial choice $v^{(0)}$, for example $v^{(0)} = (\rho_\lambda)_{\lambda\in\Lambda}$, we define

$$\begin{aligned} u^{(n)} &:= \arg\min_{u\in\ell_2(\Lambda)^M} J(u, v^{(n-1)}), \\ v^{(n)} &:= \arg\min_{v\geq 0} J(u^{(n)}, v). \end{aligned} \tag{12}$$

Once again we see that an alternating minimization on several auxiliary variables plays a useful role in order to generate compressive algorithms. The minimizer $v^{(n)}$ of $J(u^{(n)}, v)$, for a fixed $u^{(n)}$, can be computed explicitly by the formula

$$v_\lambda^{(n)} = \begin{cases} \rho_\lambda - \frac{1}{2\theta_\lambda}\|u_\lambda^{(n)}\|_q, & \|u_\lambda^{(n)}\|_q < 2\theta_\lambda\rho_\lambda, \\ 0, & \text{otherwise}, \end{cases} \qquad \lambda \in \Lambda. \tag{13}$$

The minimization of $J(u, v^{(n-1)})$ with respect to u and fixed $v^{(n-1)}$ can be done by a thresholded Landweber iteration similar to the one analyzed in [41]. We showed in [66] that, for suitable choices of the parameters θ, ρ, ω, the algorithm (12) converges to the minimizer of the functional J and we show its linear convergence rate.

The functional $J = J_{\theta,\rho,\omega}^{(q)}$ depends on several parameters. In the effort of clarifying their role in shaping the optimal solution u^* we discovered an intriguing relationship between our new functional (11) and hard-thresholding, and more generally to the so-called *firm-thresholding*. We proved that different choices of the parameters $\theta = (\theta_\lambda), \omega = (\omega_\lambda)$ and $\rho = (\rho_\lambda)$ do generate an entire family of thresholding algorithms, which perform the corresponding minimization by the iteration

$$u^{(n)} = H_{\theta,\rho,\omega}^{(q)}(u^{(n-1)} + L^*(f - Lu^{(n-1)})), \quad n \in \mathbb{N}. \tag{14}$$

Actually, for the simplest case $M = 1$, i.e., in the scalar situation, the triple $\gamma = (\theta, \rho, \omega)$ very clearly defines a "shape parameter" since the form of the

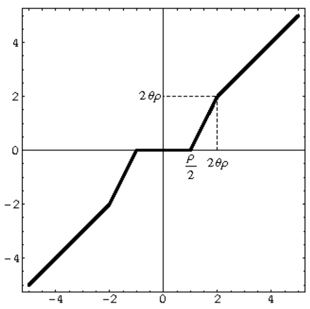


Fig. 11. For $M = 1$ and $\omega = 0$, the thresholding function $H_{\theta,\rho,\omega}^{(q)}$ acts componentwise applying the firm-thresholding curve here depicted

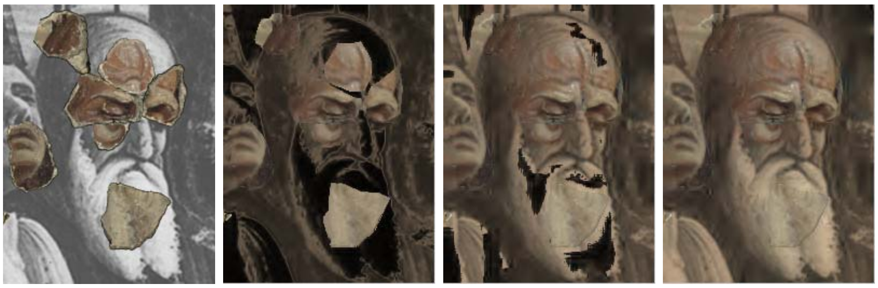


Fig. 12. We illustrate a recolorization (three iterates of the joint-sparsity promoting algorithm (14)) of an image from a few color fragments and gray level information of the missing parts [65]

corresponding thresholding function $H_\gamma := H^{(q)}_{\theta,\rho,\omega}$ (called *firm-thresholding*) changes accordingly, see Figure 11. In [67] we provided the proof of convergence of the latter algorithm to the minimizer u^*. Note that in this case, the auxiliary variable v does not play an explicit role anymore, although its presence implicitly contributes to the specific shape of the firm-thresholding function. By employing techniques of Γ-convergence and variational limits [39], we proved that the dependence of minimizers on the shape parameters $\theta = (\theta_\lambda), \omega = (\omega_\lambda)$ is continuous.

3.6 Compressive Algorithms Meet Free-Discontinuity Problems

As there exists a natural correspondence between total variation and ℓ_1-minimizations, there is also a corresponding *compressibility* in terms of derivatives to the one promoted by firm-thresholding algorithms. As we discovered recently, functionals of the type (11) are discrete approximations of functionals modelling free-discontinuity problems. The terminology "free-discontinuity problems" was introduced by De Giorgi in the late-1980's to indicate a class of variational problems that consist of the minimization of a functional, involving both volume and surface energies, depending on a closed set K, and a function u usually smooth outside of K: typically

– K is not fixed a priori and it is an unknown of the problem;
– K is not a boundary in general, but a free-surface inside the domain of the problem.

For a broad overview on free-discontinuity problems and their analysis, we refer to [2]. One of the best known examples of free-discontinuity problems is the one modelled by the Mumford-Shah functional [80] defined by

$$J(u,K) := \int_{\Omega\setminus K} \left[|\nabla u|^2 + \alpha(u-g)^2\right] dx + \beta\mathcal{H}_{n-1}(K\cap\Omega).$$

The set Ω is a bounded open subset of $\mathbb{R}^n$, $\alpha, \beta > 0$ are fixed parameters and $g \in L^\infty(\Omega)$. Here $\mathcal{H}_n$ denotes the n-dimensional Hausdorff measure. In this model, we seek for a function $u \in W^{1,2}(\Omega \setminus K)$ that approximates the datum g, the function u is smooth out of the discontinuity set K. In visual analysis g is a given noisy image that we want to approximate by u which is instead smooth except for a rectifiable set K, the set K is also used in order to *segment* the image into connected components.

In fact, the Mumford-Shah functional is the continuous version of a previous discrete formulation of the image segmentation problem proposed by Geman and Geman in [70]; see also the work of Blake and Zisserman in [6]. Let us recall this discrete approach. For simplicity let $n = 2$ (as for image processing problems), $\Omega = [0,1]^2$, and let $u_{i,j} = u(hi, hj)$, $(i,j) \in \mathbb{Z}^2$ be a discrete function defined on $\Omega_h := \Omega \cap h\mathbb{Z}^2$, for $h > 0$. Define $W_h(t) = \min\{t^2, \beta/h\}$ to be the truncated quadratic potential, and

$$\begin{aligned}\mathcal{J}_{\sqrt{\beta/h}}(u) := h^2 &\sum_{(hi,hj)\in\Omega_h} W_h\left(\frac{u_{i+1,j} - u_{i,j}}{h}\right) \\ + h^2 &\sum_{(hi,hj)\in\Omega_h} W_h\left(\frac{u_{i,j+1} - u_{i,j}}{h}\right) \\ + \alpha h^2 &\sum_{(hi,hj)\in\Omega_h} (u_{i,j} - g_{i,j})^2.\end{aligned}$$

Chambolle et al. [7,18,14] gave formal clarification as to how the discrete functional $\mathcal{J}_{\sqrt{\beta/h}}$ approximates the continuous functional J. It has been pointed out in [69] that, by discretization of the Mumford-Shah functional by means of suitable finite elements, and then by expressing the problem in terms of sole discrete derivatives, one can re-formulate the problem into a finite dimensional nonconvex and constrained optimization of the general type:

$$\begin{cases}\text{Minimize } \mathcal{J}_\gamma^p(u) = \left[\|Lu - f\|^2_{\ell_2^M} + \sum_{i=1}^N \min\{|u_i|^p, \gamma^p\}\right] \\ \text{subject to } \mathcal{Q}u = 0.\end{cases} \tag{15}$$

where $\mathcal{Q}$ is a suitable linear constraint, $\gamma > 0$, and $1 \leq p \leq 2$. In [69] we remarkably proved that these nonconvex functionals have always minimizers independently of the particular choice of L. Note that this result is not obvious since the functional is not convex, and for arbitrary choices of noninjective operators L, the problem does not seem *a priori* to be coercive. We also pointed out that the computation of these minimizers is an NP-hard problem [1].

Associated with the unconstrained minimization of $\mathcal{J}_r^p$ we designed an iterative thresholding algorithm

$$u^{(n+1)} = H_{(p,\gamma)}(u^{(n)} + L^*(f - Lu^{(n)})), \tag{16}$$

where $H_{(p,r)}$ is again a *thresholding function* which acts component-wise as depicted in Figure 13 for $p \in \{1, 3/2, 2\}$. This algorithm converges to fixed points $\bar{u}$ of the iteration (16).

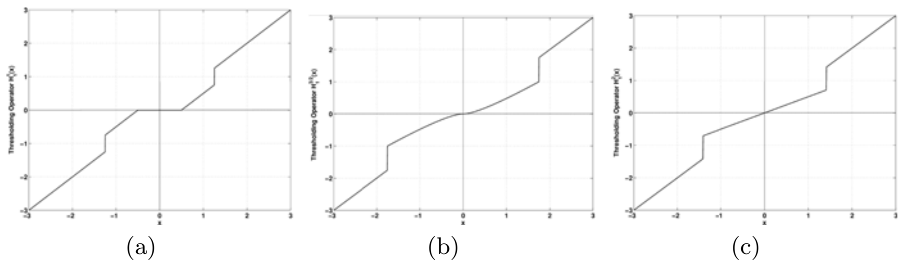


Fig. 13. The component-wise discontinuous thresholding functions $H_{(1,1)}$, $H_{(3/2,1)}$, and $H_{(2,1)}$, with parameters $p \in \{1, 3/2, 2\}$, respectively, and $\gamma = 1$

Theorem 3. *A fixed point $\bar{u}$ of the iteration* (16) *is a local minimizer of the functional $\mathcal{J}_\gamma^p$ defined in* (15).

More surprising is that global minimizers of $\mathcal{J}_\gamma^p$ are also fixed points, as shown in the following theorem.

Theorem 4. *Any global minimizer u^* of $\mathcal{J}_\gamma^p$ is a fixed point of the iteration* (13).

We reiterate that the computation of global minimizers is an NP-hard problem, and therefore the algorithm (16) is of particular importance because it furnishes a method to approximate local minimizer, but also it has the chance of finding a global one. We note that on a ball $B(\bar{u}, \varepsilon(\gamma))$ around an equilibrium point $\bar{u}$ of radius $\varepsilon(\gamma) > 0$ sufficiently small, the functional $\mathcal{J}_\gamma^p$ is convex; it is possible to show that $\mathcal{J}_\gamma^p$ is in fact *strictly* convex whenever $\bar{u} = u^*$ is a global minimizer. Hence a global minimizer is necessarily an isolated minimizer, whereas we cannot ensure the same property for local minimizers if L has a nontrivial null-space; in this case, local minimizers may form continuous sets. We conclude the following remark.

Corollary 1. *Minimizers of $\mathcal{J}_\gamma^p$ are isolated.*

4 Conclusion and Future Perspectives

We gave an overview of a few significant instances of iterative compressive algorithms which show their versatility in several and diverse applications. We ranged through optimal adaptive solution of PDEs, inverse problems with sparsity constraints, and free-discontinuity problems. We wanted to emphasize the capability of suitable implementations of compressive algorithms, e.g., via domain decomposition and subspace correction methods, of performing effective dimensionality reductions, also in problems of very large dimension. We stressed the variational nature of such algorithms that usually dispose of an associated energy functional which is minimized during the iterations, and provides additional robustness to the iterative process.

We expect that future developments of numerical methods, for instance, for nonlinear PDEs [94], will continue to be influenced by such approaches: new tools to cope with the "curse of dimensionality", further systematic developments of adaptivity in the presence of different scales, probabilistic algorithms, an increasing role for combinatorial aspects of the underlying algorithms, are few examples expected from future developments.

Acknowledgment

The author acknowledges the support of the FWF project no. Y-432-N15 START-award "Sparse Approximation and Optimization in High Dimensions".

References

1. Alexeev, B., Ward, R.: On the complexity of free-discontinuity and ℓ_0 regularization in image processing (preprint) (2009)
2. Ambrosio, L., Fusco, N., Pallara, D.: Functions of Bounded Variation and Free Discontinuity Problems. Oxford Mathematical Monographs, xviii. Clarendon Press, Oxford (2000)
3. Aubert, G., Kornprobst, P.: Mathematical Problems in Image Processing. In: Partial Differential Equations and the Calculus of Variation. Springer, Heidelberg (2002)
4. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. 2(1), 183–202 (2009)
5. Binev, P., Dahmen, W., DeVore, R.: Adaptive finite element methods with convergence rates. Numer. Math. 97(2), 219–268 (2004)
6. Blake, A., Zisserman, A.: Visual Reconstruction. MIT Press, Cambridge (1987)
7. Bourdin, B., Chambolle, A.: Implementation of an adaptive finite-element approximation of the Mumford-Shah functional. Numer. Math. 85(4), 609–646 (2000)
8. Candès, E.J.: Compressive sampling. In: International Congress of Mathematicians. Eur. Math. Soc., Zürich, vol. III, pp. 1433–1452 (2006)
9. Candès, E., Romberg, J., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Comm. Pure Appl. Math. 59(8), 1207–1223 (2006)
10. Candès, E., Tao, T.: Near optimal signal recovery from random projections: universal encoding strategies? IEEE Trans. Inform. Theory 52(12), 5406–5425 (2006)
11. Candès, E.J., Romberg, J., Tao, T.: Exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory 52(2), 489–509 (2006)
12. Candès, E.J., Tao, T.: Decoding by linear programming. IEEE Trans. Inform. Theory 51(12), 4203–4215 (2005)
13. Candès, E.J., Wakin, M., Boyd, S.: Enhancing sparsity by reweighted l1 minimization (Technical Report, California Institute of Technology)
14. Chambolle, A.: Image segmentation by variational methods: Mumford and Shah functional and the discrete approximations. SIAM J. Appl. Math. 55(3), 827–863 (1995)
15. Chambolle, A.: An algorithm for total variation minimization and applications. J. Math. Imaging Vision 20(1-2), 89–97 (2004)

16. Chambolle, A., DeVore, R.A., Lee, N.-Y., Lucier, B.J.: Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. IEEE Trans. Image Process. 7(3), 319–335 (1998)
17. Chambolle, A., Lions, P.-L.: Image recovery via total variation minimization and related problems. Numer. Math. 76(2), 167–188 (1997)
18. Chambolle, A., Dal Maso, G.: Discrete approximation of the Mumford-Shah functional in dimension two. M2AN Math. Model. Numer. Anal. 33(4), 651–672 (1999)
19. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. 1, 33–61 (1998) (Reprint)
20. Christensen, O.: An Introduction to Frames and Riesz Bases. Birkhäuser, Boston (2003)
21. Claerbout, J.F., Muir, F.: Robust modeling with erratic data. Geophysics 38(5), 826–844 (1973)
22. Cohen, A., Dahmen, W., DeVore, R.: Adaptive wavelet methods for elliptic operator equations — Convergence rates. Math. Comp. 70, 27–75 (2001)
23. Cohen, A., Dahmen, W., DeVore, R.: Adaptive wavelet methods II: Beyond the elliptic case. Found. Comput. Math. 2(3), 203–245 (2002)
24. Cohen, A., Dahmen, W., DeVore, R.: Adaptive wavelet schemes for nonlinear variational problems. SIAM J. Numer. Anal. 41(5), 1785–1823 (2003)
25. Cohen, A., Hoffmann, M., Reiss, M.: Adaptive wavelet Galerkin methods for linear inverse problems. SIAM J. Numer. Anal. 42(4), 1479–1501 (2004)
26. Combettes, P.L., Wajs, V.R.: Signal recovery by proximal forward-backward splitting. Multiscale Model. Simul. 4(4), 1168–1200 (2005)
27. Dahlke, S.: Besov regularity for elliptic boundary value problems on polygonal domains. Applied Mathematics Letters 12, 31–38 (1999)
28. Dahlke, S., Dahmen, W., DeVore, R.: Nonlinear approximation and adaptive techniques for solving elliptic operator equations. In: Dahmen, W., Kurdila, A., Oswald, P. (eds.) Multiscale Wavelet Methods for Partial Differential Equations, pp. 237–283. Academic Press, San Diego (1997)
29. Dahlke, S., Dahmen, W., Hochmuth, R., Schneider, R.: Stable multiscale bases and local error estimation for elliptic problems. Appl. Numer. Math. 23, 21–48 (1997)
30. Dahlke, S., Dahmen, W., Urban, K.: Adaptive wavelet methods for saddle point problems — Optimal convergence rates. SIAM J. Numer. Anal. 40(4), 1230–1262 (2002)
31. Dahlke, S., DeVore, R.: Besov regularity for elliptic boundary value problems. Commun. Partial Differ. Equations 22(1&2), 1–16 (1997)
32. Dahlke, S., Fornasier, M., Primbs, M., Raasch, T., Werner, M.: Nonlinear and adaptive frame approximation schemes for elliptic pdes: Theory and numerical experiments. To appear in Numerical Methods for Partial Differential Equations (2009)
33. Dahlke, S., Fornasier, M., Raasch, T.: Adaptive frame methods for elliptic operator equations. Adv. Comput. Math. 27(1), 27–63 (2007)
34. Dahlke, S., Fornasier, M., Raasch, T., Stevenson, R., Werner, M.: Adaptive frame methods for elliptic operator equations: The steepest descent approach. IMA J. Numer. Anal. 27(4), 717–740 (2007)
35. Dahlke, S., Hochmuth, R., Urban, K.: Adaptive wavelet methods for saddle point problems. M2AN Math. Model. Numer. Anal. 34, 1003–1022 (2000)
36. Dahmen, W., Schneider, R.: Wavelets with complementary boundary conditions — Function spaces on the cube. Result. Math. 34(3–4), 255–293 (1998)
37. Dahmen, W., Schneider, R.: Composite wavelet bases for operator equations. Math. Comp. 68, 1533–1567 (1999)

38. Dahmen, W., Schneider, R.: Wavelets on manifolds I. Construction and domain decomposition. SIAM J. Math. Anal. 31, 184–230 (1999)
39. Dal Maso, G.: An Introduction to Γ-Convergence. Birkhäuser, Boston (1993)
40. Daubechies, I.: Ten Lectures on Wavelets. SIAM, Philadelphia (1992)
41. Daubechies, I., Defrise, M., DeMol, C.: An iterative thresholding algorithm for linear inverse problems. Comm. Pure Appl. Math. 57(11), 1413–1457 (2004)
42. Daubechies, I., DeVore, R., Fornasier, M., Güntürk, S.: Iteratively re-weighted least squares minimization for sparse recovery. To appear in Commun. Pure Appl. Math. (2009); arXiv:0807.0575
43. Daubechies, I., Fornasier, M., Loris, I.: Accelerated projected gradient methods for linear inverse problems with sparsity constraints. J. Fourier Anal. Appl. 14(5-6), 764–792 (2008)
44. Daubechies, I., Teschke, G.: Variational image restoration by means of wavelets: Simultaneous decomposition, deblurring, and denoising. Appl. Comput. Harmon. Anal. 19(1), 1–16 (2005)
45. Daubechies, I., Teschke, G., Vese, L.: Iteratively solving linear inverse problems under general convex constraints. Inverse Problems and Imaging 1, 29–46 (2007)
46. Daudet, L., Torrésani, B.: Hybrid representations for audiophonic signal encoding. Signal Processing 82(11), 1595–1617 (2002)
47. DeVore, R.: Nonlinear approximation. Acta Numerica 7, 51–150 (1998)
48. Donoho, D.L.: Superresolution via sparsity constraints. SIAM J. Math. Anal. 23(5), 1309–1331 (1992)
49. Donoho, D.L.: De-noising by soft-thresholding. IEEE Trans. Inf. Theory 41(3), 613–627 (1995)
50. Donoho, D.L.: Nonlinear solution of linear inverse problems by wavelet-vaguelette decomposition. Appl. Comput. Harmon. Anal. 2(2), 101–126 (1995)
51. Donoho, D.L.: Compressed sensing. IEEE Trans. Inf. Theory 52(4), 1289–1306 (2006)
52. Donoho, D.L.: High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension. Discrete Comput. Geom. 35(4), 617–652 (2006)
53. Donoho, D.L., Johnstone, I.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika 81(3), 425–455 (1994)
54. Donoho, D.L., Johnstone, I.M., Kerkyacharian, G., Picard, D.: Wavelet shrinkage: asymptopia? J. Roy. Statist. Soc. Ser. B 57(2), 301–369 (1995)
55. Donoho, D.L., Logan, B.F.: Signal recovery and the large sieve. SIAM J. Appl. Math. 52(2), 557–591 (1992)
56. Donoho, D.L., Stark, P.B.: Uncertainty principles and signal recovery. SIAM J. Appl. Math. 49(3), 906–931 (1989)
57. Donoho, D.L., Tanner, J.: Sparse nonnegative solutions of underdetermined linear equations by linear programming. Proc. Nat. Acad. Sci. 102(27), 9446–9451 (2005)
58. Donoho, D.L., Tanner, J.: Counting faces of randomly-projected polytopes when the projection radically lowers dimension (preprint) (2006)
59. Duffin, R.J., Schaeffer, A.C.: A class of nonharmonic Fourier series. Trans. Amer. Math. Soc. 72, 341–366 (1952)
60. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. In: Mathematics and its Applications (Dordrecht), p. 375. Kluwer Academic Publishers, Dordrecht (1996)
61. Figueiredo, M., Nowak, R., Wright, S.J.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. IEEE J. Selected Topics in Signal Process 4(1), 586–597 (2007)

62. Fornasier, M.: Quasi-orthogonal decompositions of structured frames. J. Math. Anal. Appl. 289(1), 180–199 (2004)
63. Fornasier, M.: Domain decomposition methods for linear inverse problems with sparsity constraints. Inverse Problems 23, 2505–2526 (2007)
64. Fornasier, M., Langer, A., Schönlieb, C.B.: Domain decomposition methods for compressed sensing. In: Proceedings of SampTA 2009, Marseilles (2009)
65. Fornasier, M., Ramlau, R., Teschke, G.: The application of joint sparsity and total variation minimization algorithms to a real-life art restoration problem. To appear in Adv. Comput. Math. (2009)
66. Fornasier, M., Rauhut, H.: Recovery algorithms for vector valued data with joint sparsity constraints. SIAM J. Numer. Anal. 46(2), 577–613 (2008)
67. Fornasier, M., Rauhut, H.: Iterative thresholding algorithms. Appl. Comput. Harmon. Anal. 25(2), 187–208 (2008)
68. Fornasier, M., Schönlieb, C.B.: Subspace correction methods for total variation and ℓ_1-minimization. To appear in SIAM J. Numer. Anal. (2009)
69. Fornasier, M., Ward, R.: Iterative thresholding meets free-discontinuity problems, arXiv:0901.2605 (2009)
70. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell 6, 721–741 (1984)
71. Gilbert, A.C., Strauss, M., Tropp, J.A., Vershynin, R.: One sketch for all: Fast algorithms for compressed sensing (preprint) (2006)
72. Gilbert, A.C., Tropp, J.A.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans. Inform. Theory 53(12), 4655–4666 (2007)
73. Gribonval, R., Nielsen, M.: Highly sparse representations from dictionaries are unique and independent of the sparseness measure. Appl. Comput. Harmon. Anal. 22(3), 335–355 (2007)
74. Gröchenig, K.: Foundations of Time–Frequency Analysis. Birkhäuser, Basel (2000)
75. Kim, S.-J., Koh, K., Lustig, M., Boyd, S., Gorinevsky, D.: A method for large-scale ℓ_1-regularized least squares problems with applications in signal processing and statistics. IEEE Journal on Selected Topics in Signal Process (2007)
76. Loris, I.: On the performance of algorithms for the minimization of ℓ_1-penalized functionals. Inverse Problems 25(3), 035008 (2009)
77. Loris, I., Nolet, G., Daubechies, I., Dahlen, F.A.: Tomographic inversion using l_1-norm regularization of wavelet coefficients. Geophys. J. Int. 170, 359–370 (2007)
78. Mallat, S.: A Wavelet Tour of Signal Processing, 2nd edn. Academic Press, San Diego (1999)
79. Morin, P., Nochetto, R.H., Siebert, K.G.: Convergence of adaptive finite element methods. SIAM Rev. 44, 631–658 (2002)
80. Mumford, D., Shah, K.: Optimal approximation by piecewise smooth functions and associated variational problems. Communications on Pure and Applied Mathematics 42, 577–684 (1989)
81. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematic Programming, Ser. A 103, 127–152 (2005)
82. Osher, S., Burger, M., Goldfarb, D., Xu, J., Yin, W.: An Iterative Regularization Method for Total Variation-Based Image Restoration. Multiscale Model. Simul. 4(2), 460–489 (2005)
83. Oswald, P.: Frames and space splittings in Hilbert spaces, Survey lectures on multilevel schemes for elliptic problems in Sobolev spaces (1997), `http://www.faculty.iu-bremen.de/poswald/bonn1.pdf`

84. Oswald, P.: Multilevel frames and Riesz bases in Sobolev spaces, Survey lectures on multilevel schemes for elliptic problems in Sobolev spaces (1997), `http://www.faculty.iu-bremen.de/poswald/bonn2.pdf`
85. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D 60(1-4), 259–268 (1992)
86. Santosa, F., Symes, W.W.: Linear inversion of band-limited reflection seismograms. SIAM J. Sci. Stat. Comput. 7(4), 1307–1330 (1986)
87. Starck, J.-L., Candès, E.J., Donoho, D.L.: Astronomical image representation by curvelet transform. Astronomy and Astrophysics 298, 785–800 (2003)
88. Starck, J.-L., Nguyen, M.K., Murtagh, F.: Wavelets and curvelets for image deconvolution: a combined approach. Signal Proc. 83, 2279–2283 (2003)
89. Stevenson, R.: Adaptive solution of operator equations using wavelet frames. SIAM J. Numer. Anal. 41(3), 1074–1100 (2003)
90. Stevenson, R.: An optimal adaptive finite element method. SIAM J. Numer. Anal. 42(5), 2188–2217 (2005)
91. Stevenson, R.: Optimality of a standard adaptive finite element method. Found. Comput. Math. 7(2), 245–269 (2007)
92. Stevenson, R., Werner, M.: Computation of differential operators in aggregated wavelet frame coordinates. IMA J. Numer. Anal. 28(2), 354–381 (2008)
93. Stevenson, R., Werner, M.: A multiplicative Schwarz adaptive wavelet method for elliptic boundary value problems. Math. Comput. 78(266), 619–644 (2009)
94. Tadmor, E.: Numerical methods for nonlinear partial differential equations (preprint) (2009)
95. Taylor, H.L., Banks, S.C., McCoy, J.F.: Deconvolution with the ℓ_1 norm. Geophysics 44(1), 39–52 (1979)
96. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Statist. Soc. Ser. B 58(1), 267–288 (1996)
97. Vese, L.: A study in the BV space of a denoising-deblurring variational problem. Appl. Math. Optim. 44, 131–161 (2001)
98. Weiss, P., Blanc-Féraud, L., Aubert, G.: Efficient schemes for total variation minimization under constraints in image processing. To appear in SIAM J. Sci. Comput. (2009)
99. Zou, J., Gilbert, A.C., Strauss, M., Daubechies, I.: Theoretical and experimental analysis of a randomized algorithm for sparse Fourier transform analysis. J. Comput. Phys. 211, 572–595 (2005)

Symmetry-Aware Mesh Processing

A. Golovinskiy, J. Podolak, and T. Funkhouser

Department of Computer Science, Princeton University, USA
{agolovin,jpodolak,funk}@cs.princeton.edu

Abstract. Perfect, partial, and approximate symmetries are pervasive in 3D surface meshes of real-world objects. However, current digital geometry processing algorithms generally ignore them, instead focusing on local shape features and differential surface properties. This paper investigates how detection of large-scale symmetries can be used to guide processing of 3D meshes. It investigates a framework for mesh processing that includes steps for symmetrization (applying a warp to make a surface more symmetric) and symmetric remeshing (approximating a surface with a mesh having symmetric topology). These steps can be used to enhance the symmetries of a mesh, to decompose a mesh into its symmetric parts and asymmetric residuals, and to establish correspondences between symmetric mesh features. Applications are demonstrated for modeling, beautification, and simplification of nearly symmetric surfaces.

Keywords: symmetry analysis, mesh processing.

1 Introduction

Symmetry is ubiquitous in our world. Almost all man-made objects are composed exclusively of symmetric parts, and many organic structures are nearly symmetric (e.g., bodies of animals, leaves of trees, etc.). It is almost impossible to find a real-world object that does not have at least one nearly perfect symmetry and/or is not composed of symmetric parts. Moreover, symmetry is an important cue for shape recognition [1], as humans readily notice departures from perfect symmetry.

For decades, however, mesh reconstruction and processing algorithms in computer graphics have largely ignored symmetries. Most algorithms operate as sequences of mesh processing operations based on local shape features and/or differential surface properties. As a result, they have difficulty reproducing and preserving global shape properties, such as symmetry.

Consider simplification, for example – when presented with an input mesh for a nearly symmetric object (e.g., a face), a simplification algorithm should produce a nearly symmetric mesh. However, to our knowledge, there is no current algorithm that satisfies this basic requirement. Certainly, if the input is perfectly symmetric, then the problem is trivial – simply process half of the mesh and then copy the result. However, if the underlying surface is symmetric but the mesh topology is not, or if the underlying surface is only approximately symmetric, then standard simplification algorithms fail to preserve symmetries present in the underlying object (Figure 10c). The result is potential artifacts in physical simulations, manufacturing processes, animations, and rendered images (e.g., asymmetric specular highlights).

Recently, researchers have introduced several methods for detecting and characterizing the symmetries in 3D data. For example, Zabrodsky et al. [2] provided a measure of approximate symmetry with respect to any transformation, and Mitra et al. [3] and Podolak et al. [4] have described algorithms for extracting the most significant approximate and partial symmetries of a 3D mesh. While symmetry analysis methods like these have been used to guide high-level geometric processing operations, such as registration, matching, segmentation, reconstruction, reverse engineering, editing, and completion, they have only begun to be incorporated into low-level mesh processing algorithms.

The main goal of this paper is to investigate ways in which symmetry analysis can guide the representation and processing of 3D surface meshes. To support this goal, we make the following contributions. First, we describe an algorithm for geometric symmetrization – i.e., deforming a surface to respect a given set of symmetries while retaining its shape as best as possible. Second, we describe an algorithm for topological symmetrization – i.e., remeshing a surface so that symmetric regions have consistent mesh topology. Third, we propose a "symmetry-aware" mesh processing framework in which geometric and/or topological symmetrization algorithms provide high-level shape information (symmetric correspondences and asymmetric residuals) that can guide mesh processing applications to produce more symmetric results for approximately symmetric inputs. Finally, we demonstrate applications of this framework for surface beautification, symmetry enhancement, attribute transfer, and simplification.

2 Background and Previous Work

Understanding the symmetries of shapes is a well studied problem with applications in many disciplines. Perfect symmetries are common in CAD models and used to guide compression, editing, and instancing [5]. However, only considering perfect symmetries is of limited use in geometric processing, in general. First, the presence of noise, numerical round-off error, or small differences in tessellation can cause models of objects that are in fact symmetric to lack perfect symmetry. Second, many asymmetric objects are composed of connected parts with different symmetries. Finally, most organic objects exhibit near, but imperfect, symmetries (leaves of trees, human bodies, etc.), and understanding those types of symmetry is important, too. Thus, it is useful to have methods to detect and utilize partial and approximate symmetries.

Towards this end, Zabrodsky et al. [2] defined the *symmetry distance* of a shape with respect to a transformation as the distance from the given shape to the closest shape that is perfectly symmetric with respect to that transformation.

They provide an algorithm to find the symmetry distance for a set of connected points for any given reflective or rotational transformation. Mitra et al. [3] and Podolak et al. [4] find a set of prominent symmetries by having points on a mesh vote for symmetries in a process similar to a Hough transform.

Measures for partial and approximate symmetry of this type have been used in a variety of computer vision applications. Perhaps the earliest example is by [6], who used deformable models with symmetry-seeking forces to reconstruct 3D surfaces from 2D images. Zabrodsky et al. used a continuous measure of symmetry for completing the outline of partially-occluded 2D contours [7], for locating faces in an image, determining the orientation of a 3D shape [2], for reconstructing 3D models from images, and for symmetrizing 3D surfaces [8].

More recently, symmetry analysis has received attention in computer graphics. Kazhdan et al. [9] constructed a symmetry descriptor and used it for registration and matching. Podolak et al. [4] used a symmetry transform for surface registration, shape matching, mesh segmentation, and viewpoint selection. Mitra et al. [3] described a method to extract a discrete set of significant symmetries and used them for segmentation and editing. Thrun et al. [10] used local symmetries and used them for completion. Gal et al. [11] developed local shape descriptors to look for approximate symmetries in 3D surfaces and used them for visualization and matching. Mills et al. [12] utilized approximate symmetries to guide reverse engineering of CAD structures from range scans. Simari et al. [13] decomposed meshes into a hierarchical tree of symmetric parts to be used for compression and segmentation. Finally, Martinet et al. [5] has detected perfect symmetries in parts of scenes and used them to build instancing hierarchies.

Perhaps closest to our work is the work of Mitra et al. [14], where a method of symmetrizing the geometry of meshes is presented. Their technique is similar to the one we present in Section 4. However, we take the output of geometric symmetrization and go further, providing algorithms for symmetric remeshing and a framework for making mesh-processing algorithms symmetry-aware.

3 Overview

The goal of our work is to provide tools for symmetry-aware processing of 3D surface meshes. We propose a multi-step processing framework, in which approximate and partial symmetries are detected, preserved, exploited, and sometimes even enhanced as meshes are processed. To support this framework, we provide the following tools, which typically will be used in the sequence of steps shown in Figure 1:

1. **Symmetry analysis:** The mesh is analyzed to detect perfect, approximate, and partial symmetries. The output of this step is a set of transformations (e.g., planes of reflection), each with a list of vertices indicating the subset of the surface mapped approximately onto itself by the transformation. For this step, which is not a focus of this paper, we use methods previously described in [4] and [3].

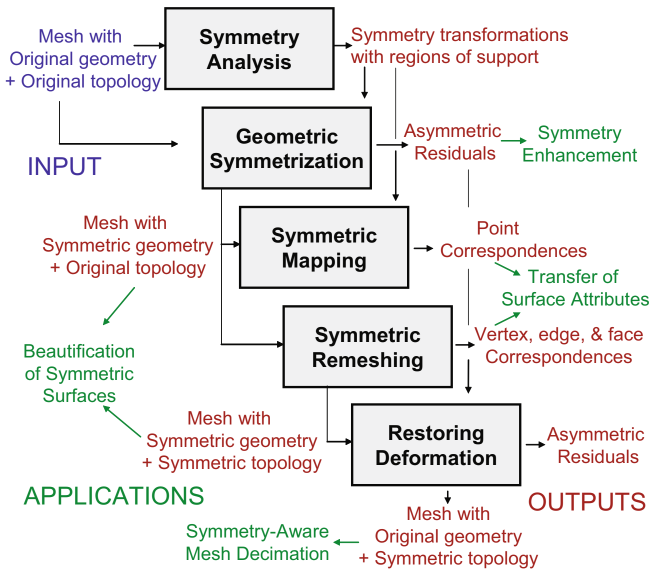


Fig. 1. Symmetry-aware mesh processing framework

2. **Geometric symmetrization:** The surface is warped to make it symmetric with respect to a given set of transformations. The primary output of this step is a “symmetrized” mesh having the same topology as the original, but with geometry that is perfectly symmetric up to the resolution of its tessellation. A secondary output is a set of asymmetric residuals storing the vector difference between the original and symmetrized position of every vertex, which can be used to compute an inverse to the symmetrizing warp.
3. **Symmetric mapping:** Correspondences are established between vertices and their images across every symmetry transformation. The output of this step is a dense set of point pairs, where one point is associated with a vertex and the other is associated with a face and its barycentric coordinates. These point pairs provide a mapping between symmetric surface patches.
4. **Symmetric remeshing:** The surface is remeshed so that every vertex, edge, and face has a one-to-one correspondence with another across every symmetry. The output of this step is a new mesh with perfectly symmetric topology, along with a list of topological correspondences.
5. **Restoring deformation:** The inverse of the symmetrizing warp is applied to the symmetrically remeshed surface to restore the original geometry. The output of this step is a mesh that is topologically symmetric, but geometrically approximates the input mesh.

The motivating idea behind this framework to provide tools that can help mesh processing algorithms to preserve large-scale symmetries present in 3D objects. Our general strategy is to factor a 3D surface into a symmetric mesh and its asymmetric residual and then to perform analysis on the symmetric mesh to gain insight into its symmetric structure. We transfer knowledge about symmetric structure back onto the original geometry so that it can be preserved and exploited as the surface is processed.

In the following sections, we investigate algorithms to support this symmetry-aware mesh processing framework, focusing on the most challenging steps: geometric symmetrization (Section 4) and symmetric remeshing (Section 5). Thereafter, in Section 6, we describe potential applications and present prototype results. Finally, we conclude with a discussion of limitations and topics for future work.

4 Geometric Symmetrization

Our first objective is to provide an algorithm that can take a given surface mesh and output a new mesh with similar shape that is symmetric with respect to a given set of transformations. More formally, given a mesh M and a set of symmetry transformations, each having a possibly local region of support on the mesh, our goal is to find the most shape-preserving warp W that produces a new mesh M' with the same topology as M, but where every vertex of M' is mapped onto corresponding points on the surface of M' by all of its symmetry transformations.

This objective is similar to classical problems in non-rigid alignment for morphing of 3D surfaces, medical imaging, surface reconstruction, and several other fields. The challenge is finding both symmetric point correspondences and the warp that aligns them simultaneously. Mitra et al. [14] solve this problem (while also detecting symmetries) using Generalized Hough Transform and clustering algorithms [14]. Since our problem is a bit simpler (symmetry transformations have already been detected), we follow a more traditional iterative approach [15], employing an algorithm that greedily minimizes alignment error while allowing increasingly non-rigid deformation [16,17,18]. At each iteration, we first propose correspondences from every vertex in the mesh M to its closest compatible point on the transformed surface for every symmetry. Then, given these correspondences, we solve for new vertex positions that minimize a symmetrizing error function (Figure 2). These two steps are iterated until the mesh is fully symmetric (i.e., every vertex transformed by all its symmetries produces a point directly on a face of the mesh).

Our symmetrizing error function balances the primary goal of making the surface more symmetric with the secondary goals of retaining its original shape and position with three error terms:

$$E(M) = \alpha E_{sym}(M) + (1-\alpha)(E_{shape}(M) + \beta E_{disp}(M))$$

The first two error terms are the important ones, as they balance the trade-off between deviations from perfect symmetry and deformations of the surface. The first term, E_{sym}, measures the sum of squared distances between vertices of the mesh and the closest points on the transformed surface. For the second term, E_{shape}, we use the shape preservation function proposed by [18], which

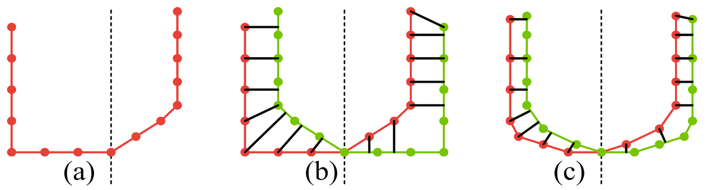


Fig. 2. Schematic of one iteration in our symmetrization process (in 2D). (a) Given a curve (red) and a symmetry transformation (reflection across the dotted line), (b) we find correspondences between vertices and the closest point on the reflected curve (green), and (c) solve for new vertex positions that minimize an error function based on those correspondences.

minimizes a measure of warp distortion. Any deformation error function would work, but this choice has the advantage of being quadratic in vertex positions. This deformation error is not rotationally invariant, but the symmetrizations performed in our experiments do not have large rotational components. The last term, E_{disp}, measures overall displacement using the sum of squared distances between current and original vertex positions. It is required to penalize global translations because E_{shape} is translationally invariant and because the surface is being warped onto itself (in contrast to traditional alignment problems where either the source or target is fixed in space). We set the weight of this error term very low relative to the others ($\beta = 1/100$), and so it has very little influence on the output surface's shape. Since all three terms of the error function are quadratic in the positions of the vertices, we can solve for the minimal error at each iteration with a least-squares solution to a linear system.

Our implementation contains several simple features that help to provide stability and speed as the optimization proceeds. First, it uses multiresolution surface approximations to accelerate convergence and avoid local minima. Prior to the optimization, the input mesh is decimated with *Qslim* [19] to several nested levels of detail. Then, coarser levels are fully symmetrized and used to seed the initial vertex placements for finer levels (new vertices added at each finer level are positioned relative to the current ones using thin-plate splines). Within each level, further stability is gained by slowly shifting emphasis of the error function from shape preservation ($\alpha = 0$) to full symmetrization ($\alpha = 1$). Finally, for each vertex, we use a k-d tree to help find the closest point on the transformed surface, and only retain correspondences to a closest point whose transformed normal does not point in the opposite direction. Overall, compute times on a 3Ghz processor range from 4 seconds for the 1,166 vertex model of the dragon in Figure 8 to 10 minutes for a 240,153 vertex face scan.

For instance, consider Figure 3, which shows the result of symmetrizing a bust of Max Plank with respect to reflection across a single vertical left-right plane. Note that the original input mesh (Figure 3a) is quite asymmetric, as seen by the misalignment of the surface (red) and its reflection (green) in the bottom images of Figure 3a – i.e., significant shape features (eyes and ears) do not map

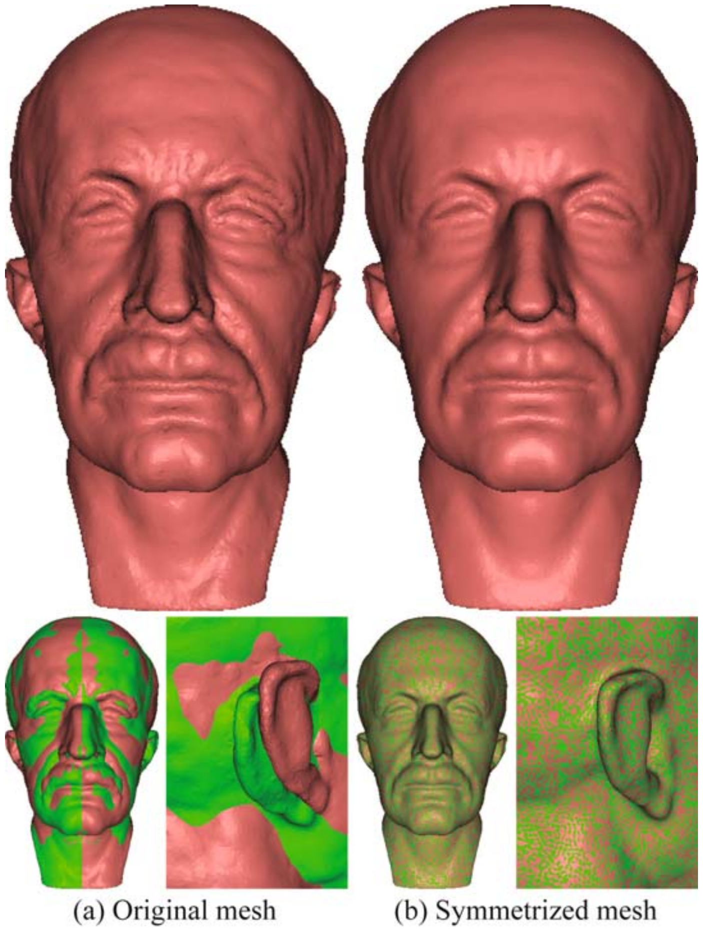


Fig. 3. Symmetrizing Max Planck. The model of a bust of Max Planck (a) asymmetric, as can be seen by overlaying the mesh (red) with its reflection (green) in the bottom images. Our method symmetrizes its geometry (b), while retaining and aligning sharp features like eyes, mouth, and ears.

onto their symmetric counterparts when reflected across the plane. However, we are able to find a non-rigid warp that aligns those features while producing a symmetric mesh with a small amount of shape distortion. The symmetrized mesh, shown Figure 3b, is perfectly symmetric up to the resolution of the mesh, as indicated by the high-frequency interleaving of the original surface (red) with its reflection (green) in the bottom images of Figure 3b.

A more complicated example demonstrates symmetrization across partial, approximate planes of symmetry in the Stanford Bunny (Figure 4). Using the method of [4], the bunny was automatically segmented into two symmetric parts (the head and the body), each supporting a plane of partial symmetry. Of course,

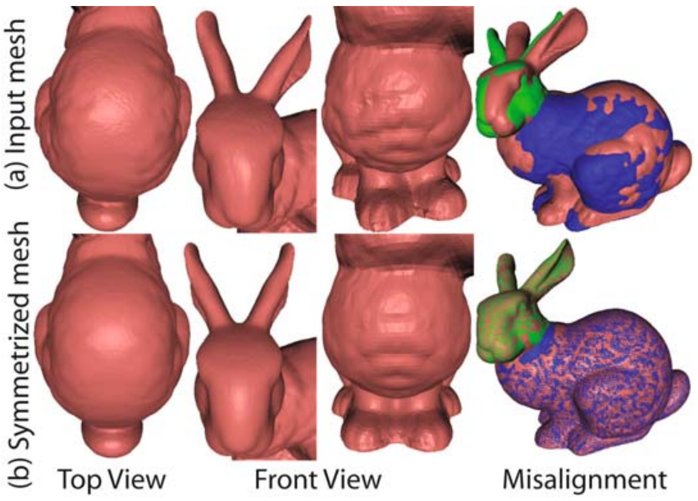


Fig. 4. Symmetrizing the bunny. The top row shows the original bunny, while the bottom row shows the symmetrized result. The bunny contains partial symmetries with respect to planes through the body and through the head. Our method symmetrizes both parts of the bunny while performing a shape-preserving blend in between, preserving features such as the eyes and feet. The right pair of images show the asymmetry of the original model and the accurate alignment of symmetric parts in our result.

both of these parts are highly asymmetric, as can be seen from top-right image in Figure 4, where each part of the bunny (red) is shown along with its reflection (green and blue) over its symmetry plane. Note how poorly the ears and feet align with their reflections. However, our geometric symmetrization algorithm is able to warp both parts into alignment with their reflections simultaneously while retaining significant shape features (e.g., ears, feet) and blending symmetries across the intermediate region of the neck. Note that the crease between the feet is preserved although the symmetry plane does not run through it on the original model, as the surface was warped to align with the plane.

The output of the process is not only a symmetrized mesh, but also a *symmetric map*, a set of point correspondences where every vertex is associated with a point on the surface across every symmetry transformation. We store the corresponding points in barycentric coordinates with respect to triangles of the mesh so that they deform with the surface (e.g., when we apply the inverse of the symmetrizing deformation to restore the original geometry). This is a key point, since it allows us to transfer the symmetric map learned from the perfectly symmetric surface back to the asymmetric one.

5 Symmetric Remeshing

Our second objective is to develop an algorithm that can take a geometrically symmetrized mesh with arbitrary topology and remesh it so that every vertex, edge, and face has a direct correspondence with another with respect to every symmetry transformation. Our motivation is to provide a topology that not only reflects the symmetric structures of the object, but also can provides efficiency in representation and manipulation due to topological redundancies (e.g., compression), cues for preservation of symmetries during topological modifications (e.g., simplification), and symmetric sampling to avoid asymmetric artifacts in photo-realistic renderings and physical simulations (e.g., boundary element methods).

This problem is a special case of compatible remeshing [20,21]. Given the symmetric map from every vertex to a point on the surface for every symmetry transformation provided by the geometric symmetrization algorithm, we aim to find the mesh with perfectly symmetric topology that has the least geometric error and/or fewest extra vertices.

A strawman approach that may be appropriate for highly symmetric and/or oversampled meshes is to partition the mesh into "asymmetric units" and then copy the topology from one instance of others in correspondence and then stitch at the boundaries. For a single planar reflection, this would entail cutting the mesh along the plane, throwing away the mesh connectivity on one side (M_t), and then copying the connectivity over from the other side (M_s). While this simple method would provide symmetric topology with the same number of vertices as the original mesh, it would produce an asymmetry in the quality of the geometric approximation (M_s would have the quality of the original surface, but M_t would have blurring where edges oriented appropriately for the geometry of M_s are not appropriate for M_t), and it would produce artifacts where the topology of M_s provides a poor approximation for M_t.

There are many methods in the literature to overcome this problem, most of which introduce a large number of extra vertices to capture the geometric variations of both M_s and M_t. For example, one way is to create an overlay meta-mesh that contains the original vertices of both M_s and M_t along with new vertices at all edge-edge intersections [22,23]. Another way is to map M_s and M_t to a common base domain (e.g., a sphere [22], or a simplified triangle mesh [23,24]) and then remesh with semi-regular connectivity until all geometric features are resolved. Alternatively, it is possible to create a meta-mesh M_{st} by inserting all the vertices of M_s into M_t, and vice-versa, and then iteratively swapping edges until a compatible mesh topology is achieved [20]. These methods all produce compatible mesh topology and so could be used for symmetric remeshing. However, the resulting mesh would usually be significantly over-sampled.

We provide a simple method to address the problem: *compatibility-preserving mesh decimation* (or, in our case, *symmetry-preserving mesh decimation*). Our general approach is to use any of the above methods to produce compatible mesh topology with vertices from both M_s and M_t, and then to decimate the resulting meta-mesh with a series of edge collapse operations that operate on corresponding edges in lock-step. Specifically, we build clusters of edges whose vertices are in symmetric correspondence and then follow the same basic approach as the original *Qslim* algorithm [19], however working on clusters rather than individual edges. We load the clusters into a priority queue sorted by the

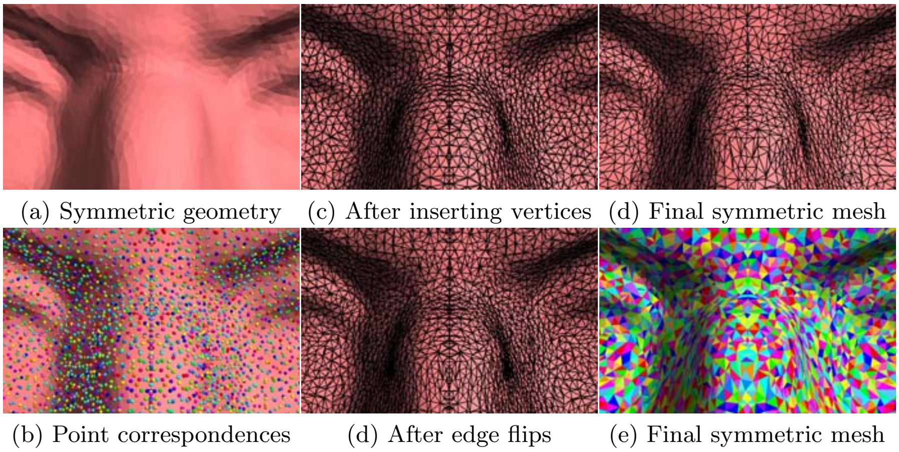


Fig. 5. Symmetric remeshing. The input is shown in the left column (the symmetrized bust of Max Planck, zoomed to the bridge of the nose); intermediate steps are shown in the middle column, and the output is shown in the right column. The final mesh has perfectly symmetric topology (as shown by colored face correspondences in (e)) and still approximates the surface well with the original number of faces.

Quadric Error Measure (QEM) of the edge with highest error in each cluster, and then we iteratively collapse all edges in the cluster with minimal error until a desired number of triangles or a maximum error has been reached. Since all edges in the same cluster are processed atomically, the method is guaranteed to maintain topological symmetries as it decimates the mesh. Yet, it still provides a good approximation of the original surface, as QEMs approximate deviation from the original surface.

This method is similar in goal to the method of [21], which copies the mesh topology of M_s onto M_t and then optimizes the positions and number of vertices to match the geometry of both M_s and M_t with a combination of smoothing and refinement operations. The difference is that we first produce an over-sampled mesh with vertices from both M_s and M_t, and then "optimize" it to minimize the QEM by decimation. Since our process is seeded directly with (a conservatively large set of) compatible vertices and edges from both M_s and M_t, the optimization starts from an initial configuration that encodes features from the entire mesh. So, our challenge is mainly to decide which vertices and edges can be removed, rather than discovering suitable places for new vertices from scratch. As a result, it is easy to produce compatible mesh topologies for any number of surface regions with any number of vertices.

We have experimented with this approach using an algorithm based on the Connectivity Transformation technique of [20] to form an over-sampled mesh with symmetric topology prior to decimation. Given a geometrically symmetrized mesh and a set of vertex-point correspondences (Figure 5a-b), we first produce a meta-mesh M_{st} with symmetric vertex correspondences by inserting all the vertices of M_s into M_t, and vice-versa, splitting faces into three when an inserted vertex maps

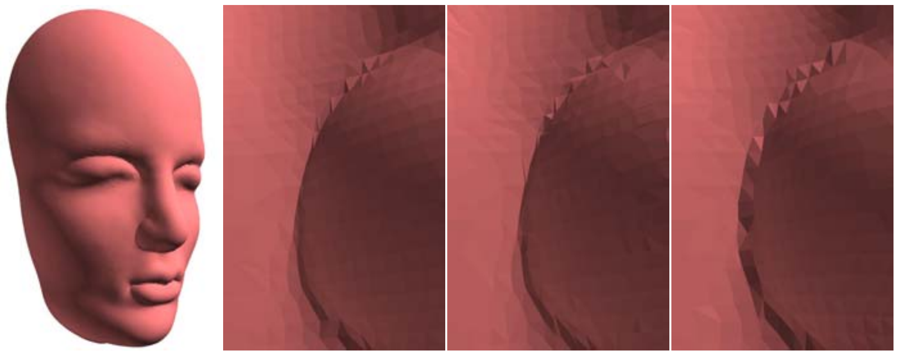


Fig. 6. A zoomed-in comparison the crease along left side of the nose on the surface of the mask shown on the left. Note how our approach (b) does not produce blurring when compared to the original (a), while the simple alternative (c) of copying topology from one side to the other does.

to the interior of an existing face (Figure 5c). We then swap edges in order of an error function that measures the differences in the QEM for edge midpoints before and after the swap, plus a quadratically growing penalty for swaps of an edge multiple times, plus an infinite penalty for any swap that would generate a topological fin in the mesh or break a greater number of symmetric correspondences than it creates. The process terminates when all edges are found to be in symmetric correspondence, or when the minimal error of any cluster exceeds some preset threshold. We have not implemented the edge-crossing constraint and termination criterion of [20], as it only guarantees convergence for meshes on a plane [25]. However, we find that our method finds symmetric correspondences for all but few edges in practice (99.9% in all of our examples). For the remaining edges, we simply copy those edges from one asymmetric unit to the other(s). The net result is a mesh with fully symmetric topology containing approximately twice as many vertices as the original (Figure 5d). We give that mesh as input to the symmetry-preserving version of Qslim to produce the final result – a topologically symmetric mesh with a user-specified number of faces or geometric error. Figure 5e-f shows the result for decimation to the number of faces in the original mesh (98K). Compute times for the entire symmetric remeshing process range between tens seconds for the dragon and two hours for the armadillo on a 3GHz processor. Of course, this process must be done only once per model.

It is difficult to make comparisons of our symmetric meshing method to others, since our problem is somewhat different from previous ones. However, to validate that our approach provides benefits over the simple strawman approach described earlier in the section (copy the topology of the right side over to the left), we provide a comparison of symmetrically remeshed surfaces of a mask along the left crease of the nose (Figure 6). Note how our method (middle) produces a mesh that retains sharp features of the original (left), whereas the simpler approach (right) suffers from blurring due to poorly oriented edges. Besides these differences in surface quality, our method has the additional advantage that it works without modification for partial and multiple symmetries of any type of transformation, and produces symmetric mesh topology at any user-selected face count.

6 Applications

The main theme of this paper is that awareness of symmetries can and should be incorporated into mesh processing algorithms. Since objects with perfect and/or approximate symmetries are prevalent in our world, and since symmetries are often critical to an object's function and/or a human's perception of it, we believe that algorithms processing 3D models should understand their symmetries and preserve them. In this section, we investigate how this can be done for several classes of applications.

Roughly speaking, applications can be divided into classes according to what type of mesh data they process, and almost equivalently, what type of symmetry information they can exploit: (1) Some applications are concerned mainly with creating new geometry (e.g., surface scanning, interactive modeling, etc.). For this class, geometric symmetrization provides a useful tool for coercing the geometry of approximate input (e.g., scanned points, sketched surfaces, etc.) to become more, less, or perfectly symmetric to match the intended structure of the object being modeled. (2) Other applications are concerned with manipulating attributes associated with local regions of a surface (e.g., texture mapping, signal processing, etc.). For them, symmetric mapping provides a way to blend and transfer attributes between symmetric regions. (3) Still other applications are concerned with the manipulating the topology of a mesh (e.g., remeshing). For those applications, symmetric remeshing provides an automatic way to coerce the mesh topology to respect the symmetric structure of an object and provides correspondence information that can be used to preserve topological symmetries as the mesh is processed further. Finally, of course, there are applications that can exploit all three types of symmetry information simultaneously (e.g., beautification, compression, etc.). In the following subsections, we show at least one example from each of these classes.

6.1 Beautification of Meshes for Symmetric Objects

There are many application domains in which scans are acquired for symmetric real-world objects. For example, in rapid prototyping applications, physical mockups are often constructed for a proposed design (e.g., with clay) and then scanned for computer simulation and processing. Likewise, in reverse engineering, objects are scanned when the original design is not available. However, rarely are the scanned models perfectly symmetric, due in part to scanner bias and noise, and due in part to processing tools that introduce asymmetries as a surface mesh is reconstructed. Since so many scanned objects are in fact symmetric, it seems

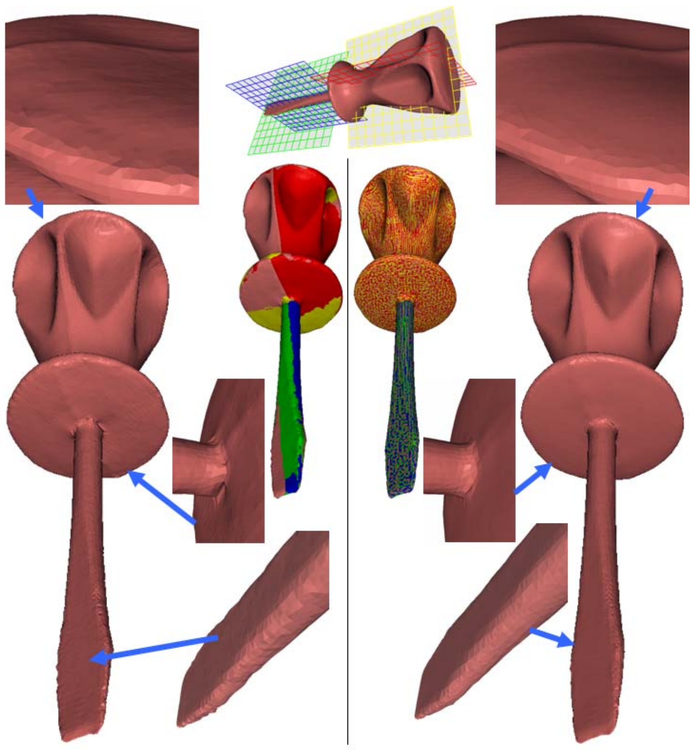


Fig. 7. Symmetrizing a scanned screwdriver model. The input mesh is shown on the left side, and the output mesh is on the right. Note that the output mesh is perfectly symmetric, has less noise (e.g., on the tip and at the junction with the handle), and retains sharp features (e.g., the ridge on the top of the handle).

useful to have a tool that takes a scanned mesh as input and produces the most similar symmetric mesh as output.

As an example, consider the scanned screwdriver downloaded from the Cyberware repository of Desktop 3D Scanner Samples (left side of Figure 7). In this case, the physical object has two parts (handle and tip), each of which is approximately symmetric with respect to two plane reflections (top-middle of Figure 7). Yet, the scanned mesh contains significant asymmetries with respect to all of these planes (e.g., artifacts at the junction of the tip and the handle).

Motivated by the idea of "beautifying" this mesh, we extracted planes of symmetry automatically with the Iterative Symmetric Points algorithm of [4], augmented to ensure that pairs of planes for the same part were perpendicular, and that all four planes aligned on a single axis (note that the planes for the handle are rotated by 20 degrees with respect to those of the tip). Then, we ran our geometric symmetrization algorithm on the entire mesh, with all four planes of symmetry guiding the surface deformation.

The result is shown in the images on the right side of Figure 7. Looking closely at the image in the middle right, it can be verified that the surface is symmetric up to the resolution of the mesh (note the high-frequency interleaved pattern of yellow, red, green, and blue overlaid surfaces). It can also be seen that significant shape features are retained during symmetrization (e.g., the ridge in the top of the handle), while noise is reduced (e.g., the tip shown in close-up on the bottom right). In general, shape features that align across multiple symmetries are retained, while those that do not are diminished. Overall, the mesh on the right of Figure 7 has the principal symmetries of the physical object and lower levels of noise, and thus is probably preferable for most simulation and visualization applications.

6.2 Symmetry Enhancement

In some applications, it may not be desirable to symmetrize a surface completely, but rather to enhance or to diminish symmetries instead. As a concrete example, imagine that a person has drawn the dragon shown in Figure 8a using a sketching tool like Teddy [26], but wants to make it more symmetric (note that the wings are quite misaligned with respect to the left-right symmetry plane). While this type of operation is possible with a series of deformations and local surface edits, it would be tedious with current modeling tools.

Instead, we propose an interactive tool that allows a user to control the degree of symmetrization applied to a surface. We provide a slider that the user can manipulate to make a surface more or less symmetric with respect to a selected transformation while the model is updated with real-time visual feedback. As an example, Figure 8b-c shows screenshots after the user has interactively symmetrized the dragon part-way (middle) and completely (right). In this simple case, the symmetrizing deformation could be computed in real-time. For more complex models, symmetry enhancement can be performed in real-time following

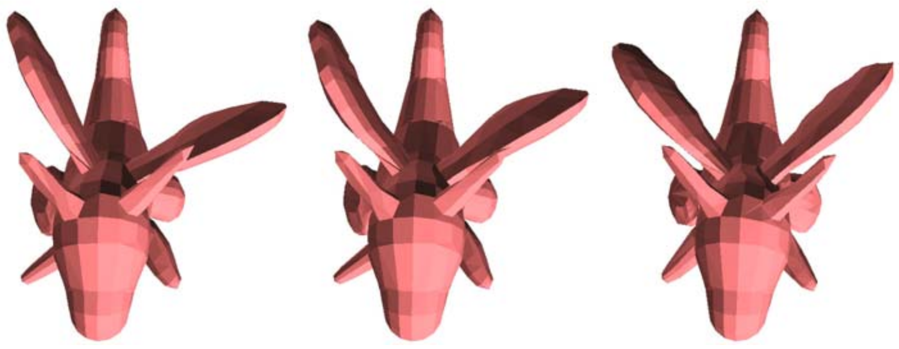

(a) Original (b) Symmetrized Part-way (c) Symmetrized Completely

Fig. 8. Enhancing the symmetries of a sketched model under interactive control

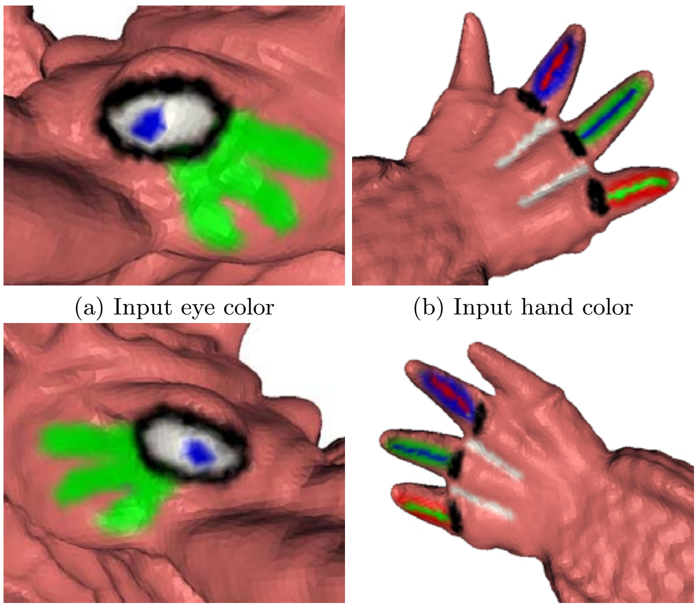
symmetrization as a pre-process (see the video for examples). We believe that such a tool would be a useful addition to the suite of commands for interactive surface design.

6.3 Attribute Transfer

There are many applications that require blending or transferring attributes between semantically related surface regions – for example, texture transfer, denoising, and morphing. The challenge is usually to establish correspondences between semantically related parts. In the case of objects with approximate symmetries, symmetric mapping provides a useful way to solve this problem.

For example, consider the Armadillo model. Although the surface is "semantically symmetric" (e.g., the arm on the left has a functional correspondence with the one on the right), the surface is not symmetric geometrically (e.g., the arms are in significantly different poses). In cases like this, our symmetrization framework provides a natural way to establish correspondences between approximately symmetric parts via symmetric mapping.

This mapping can be used to transfer and blend surface attributes. For example, Figure 9 shows a demonstration of transferring per-vertex colors between symmetric regions of the Armadillo model. In the top row, the user has drawn colors on the eyes and hands on one side of the surface with an interactive


(a) Input eye color (b) Input hand color

(c) Transferred eye color (d) Transferred hand color

Fig. 9. Transferring surface attributes between symmetric parts

painting interface. The system then automatically transfers the colors to the other side (bottom row) via an automatically generated symmetric map.

In this example, the main benefit is to save the user the effort of painting details twice. However, in other examples, perhaps it is important that the surface attributes are applied to both sides in exactly the same way, or that surface details are blended very precisely, which would be difficult without guidance from a symmetric map.

6.4 Simplification

Simplification algorithms take a mesh and produce an approximation with fewer polygons, usually to increase rendering speed, decrease storage, and/or provide a base domain for parameterization. Generally, however, they do not preserve large-scale symmetries (or other global shape features), in favor of minimizing local geometric errors.

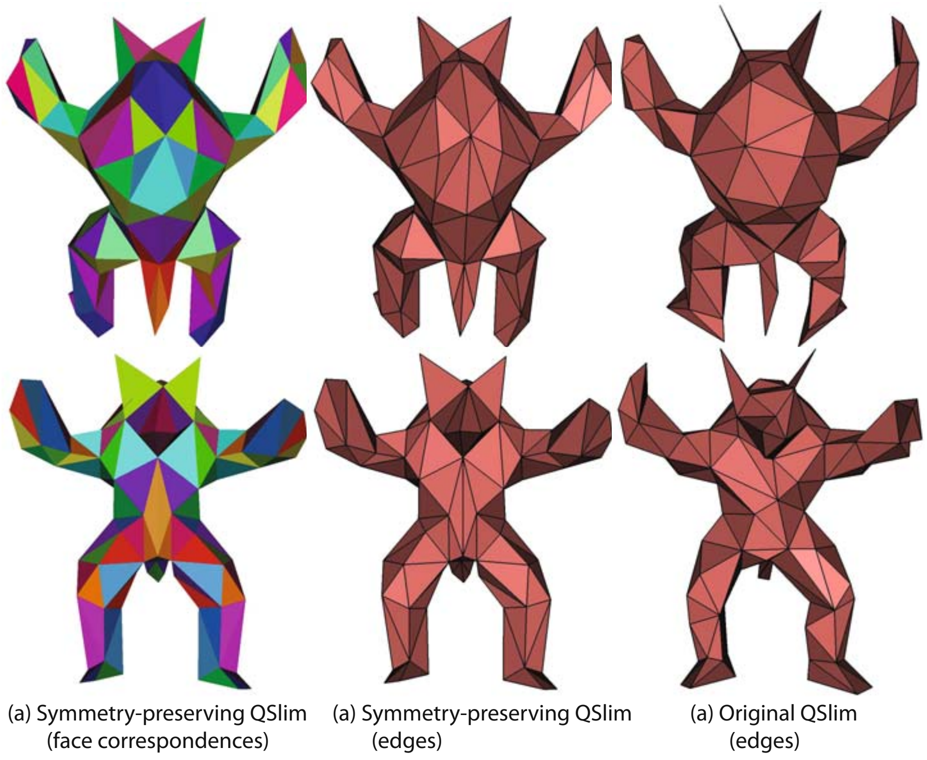

(a) Symmetry-preserving QSlim (face correspondences)
(a) Symmetry-preserving QSlim (edges)
(a) Original QSlim (edges)

Fig. 10. Symmetry-preserving QSlim (a-b) produces a surface approximation comparable to the original algorithm (c) of [19], but guaranteed to have symmetric topology, even for an asymmetric surface (the first column (a) shows symmetric face correspondences preserved during the decimation)

In this section, we investigate whether the symmetry-preserving mesh decimation algorithm described in Section 5 can be used effectively for extreme simplification of approximately symmetric surfaces. Following the general approach outlined in Section 3, we establish symmetric topology for an asymmetric surface by first symmetrizing it, remeshing with symmetric topology, and then warping the new topology and correspondences back to the original geometry. We then perform symmetry-preserving mesh decimation on the symmetric topology over the asymmetric mesh.

Figure 10 shows the results of this method (first two columns) in comparison to the original version of *Qslim* (last column). Note that the topology of the mesh output by our algorithm is perfectly symmetric, even though the geometry of the surface is not. Note also that the geometric approximation achieved with symmetry-aware simplification is similar to the original (according to Metro [27], it has a Hausdorff distance approximately 6% larger). Since the symmetric mesh better reflects the semantic structure of the surface, we believe it may be preferable as a base domain for parameterization, animation, simulation, and other applications.

7 Conclusion

In summary, this paper has investigated methods for and applications of symmetrizing 3D surface meshes. The main idea is that symmetry-aware algorithms can be used to preserve, exploit, and enhance structural symmetries of a surface, even if the underlying geometry is only approximately symmetric. This idea is important because the vast majority of objects in the world have some sort of structural symmetries, and current mesh processing algorithms generally do not preserve them.

The main contribution of this paper is the symmetry-aware mesh processing framework, which includes algorithms for geometric symmetrization and symmetric remeshing. We provide demonstration of the framework for mesh beautification, symmetry enhancement, attribute transfer, and simplification.

The initial results seem promising, but our implementation has limitations, which suggest immediate topics for future work. We have demonstrated our algorithms only for symmetries across planar reflections. Although our code can handle symmetries for arbitrary affine transformations, we have not investigated examples of this type in our study.

Considering steps forward, the most obvious next step is to investigate other applications enabled by symmetry-aware processing. First candidates include compression and denoising. In the former case, it is possible that factoring a mesh into its symmetric part and its asymmetric residual could provide increased compression ratios, since at least half of the symmetric part can be discarded [13]. For denoising, the symmetric map could provide a way to blend noise across symmetric surfaces, as in Smoothing by Example [28]. These are just two examples – considering other applications that exploit symmetries will be a fruitful topic for future work.

The main long-term direction suggested by this work is that digital geometry processing algorithms can and should consider large-scale structural features as well as local surface properties when processing a mesh. So, future work should consider better ways to detect and encode large-scale shape features (such as symmetry) and to preserve and exploit them during surface processing.

Acknowledgments. The authors would like to thank Forrester Cole, Szymon Rusinkiewicz, and the Princeton Graphics group for useful discussions about this project. We are also grateful to several funding agencies that provided partial funding for this work: National Science Foundation Grants IIS-0612231, CCF-0702672, CNS-0406415, and CNS-0831374; Air Force Research Laboratory Grant FA8650-04-1-1718; and the Google Research Grants program.

References

1. Ferguson, R.W.: Modeling orientation effects in symmetry detection: The role of visual structure. In: Proc. Conf. Cognitive Science Society (2000)
2. Zabrodsky, H., Peleg, S., Avnir, D.: Symmetry as a continuous feature. Trans. PAMI 17(12), 1154–1166 (1995)
3. Mitra, N.J., Guibas, L., Pauly, M.: Partial and approximate symmetry detection for 3D geometry 25(3), 560–568 (2006)
4. Podolak, J., Shilane, P., Golovinskiy, A., Rusinkiewicz, S., Funkhouser, T.: A planar-reflective symmetry transform for 3D shapes. ACM Transactions on Graphics (Proc. Siggraph) 25(3) (2006)
5. Martinet, A., Soler, C., Holzschuch, N., Sillion, F.: Accurately detecting symmetries of 3D shapes. Technical Report RR-5692, INRIA (2005)
6. Terzopoulos, D., Witkin, A., Kass, M.: Symmetry-seeking models and 3D object reconstruction 3(1), 221 (1987)
7. Zabrodsky, H., Peleg, S., Avnir, D.: Completion of occluded shapes using symmetry. In: Proc. CVPR, pp. 678–679 (1993)
8. Zabrodsky, H., Weinshall, D.: Using bilateral symmetry to improve 3D reconstruction from image sequences. Comput. Vis. Image Underst. 67(1), 48–57 (1997)
9. Kazhdan, M., Chazelle, B., Dobkin, D., Funkhouser, T., Rusinkiewicz, S.: A reflective symmetry descriptor for 3D models. Algorithmica 38(1) (2003)
10. Thrun, S., Wegbreit, B.: Shape from symmetry. In: Proceedings of the International Conference on Computer Vision (ICCV), Bejing, China. IEEE, Los Alamitos (2005)
11. Gal, R., Cohen-Or, D.: Salient geometric features for partial shape matching and similarity. ACM Transaction on Graphics (2005)
12. Mills, B.I., Langbein, F.C., Marshall, A.D., Martin, R.R.: Approximate symmetry detection for reverse engineering. In: SMA 2001: Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 241–248. ACM Press, New York (2001)
13. Simari, P., Kalogerakis, E., Singh, K.: Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In: Proceedings of the Symposium on Geometry Processing (SGP 2006), pp. 111–119 (2006)
14. Mitra, N.J., Guibas, L., Pauly, M.: Symmetrization. ACM Transactions on Graphics 26, #63, 1–8 (2007)

15. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. PAMI 14(2), 239–256 (1992)
16. Allen, B., Curless, B., Popoviȁc, Z.: The space of human body shapes: reconstruction and parameterization from range scans. In: SIGGRAPH 2003: ACM SIGGRAPH 2003 Papers, pp. 587–594. ACM Press, New York (2003)
17. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. In: SIGGRAPH 2004: ACM SIGGRAPH 2004 Papers, pp. 399–405. ACM Press, New York (2004)
18. Pauly, M., Mitra, N.J., Giesen, J., Gross, M., Guibas, L.: Example-based 3D scan completion. In: Symposium on Geometry Processing, pp. 23–32 (2005)
19. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Computer Graphics (Siggraph 1997), pp. 209–216 (1997)
20. Ahn, M., Lee, S., Seidel, H.: Connectivity transformation for mesh metamorphosis. In: Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 75–82 (2004)
21. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3D models. ACM Transactions on Graphics (Proc. SIGGRAPH 2004) 23(3), 861–869 (2004)
22. Alexa, M.: Merging polyhedral shapes with scattered features. The Visual Computer 16(1), 26–37 (2000)
23. Lee, A., Dobkin, D., Sweldens, W., Schroeder, P.: Multiresolution mesh morphing. ACM Transactions on Graphics (Proc. SIGGRAPH 2001), 343–350 (1999)
24. Praun, E., Sweldens, W., Schroeder, P.: Consistent mesh parameterizations. ACM Transactions on Graphics (Proc. SIGGRAPH 2001), 179–184 (2001)
25. Hanke, S., Ottmann, T., Schuierer, S.: The edge-flipping distance of triangulations. Journal of Universal Computer Science 2(8), 570–579 (1996)
26. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3D freeform design. In: Computer Graphics (Siggraph 1999), pp. 409–416. Addison Wesley Longman, Amsterdam (1999)
27. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Computer Graphics Forum 17(2), 167–174 (1998)
28. Yoshizawa, S., Belyaev, A., Seidel, H.: Smoothing by example: Mesh denoising by averaging with similarity-based weights. In: Shape Modeling International, pp. 38–44 (2006)

Recent Advances in Computational Conformal Geometry

X.D. Gu[1], F. Luo[2], and S.-T. Yau[3]

[1] Computer Science Department, State University of New York at Stony Brook, USA
[2] Mathematics Department, Rutgers University, USA
[3] Mathematics Department, Harvard University, USA

Abstract. Computational conformal geometry focuses on developing the computational methodologies on discrete surfaces to discover conformal geometric invariants. In this work, we briefly summarize the recent developments for methods and related applications in computational conformal geometry. There are two major approaches, holomorphic differentials and curvature flow. The holomorphic differential method is a linear method, which is more efficient and robust to triangulations with lower quality. The curvature flow method is nonlinear and requires higher quality triangulations, but more flexible. The conformal geometric methods have been broadly applied in many engineering fields, such as computer graphics, vision, geometric modeling and medical imaging. The algorithms are robust for surfaces scanned from real life, general for surfaces with different topologies. The efficiency and efficacy of the algorithms are demonstrated by the experimental results.

Keywords: Computational Conformal Geometry, Holomorphic Differentials, Curvature Flow.

1 Introduction

Computational conformal geometry focuses on developing the computational methodologies on discrete surfaces to discover conformal geometric invariants. Computational conformal geometry is an emerging field, which combines differential geometry, algebraic topology, complex analysis, Riemann surface theory, algebraic geometry with computer science. It has broad applications in many fields in both pure theoretic research, such as mathematics, theoretical physics, and engineering applications, such as mechanics, computer graphics, computer vision, geometric modeling, network and medical imaging.

Classical computational complex analysis focuses on the mappings among domains on the complex plane $\mathbb{C}$. Classical methods for constructing conformal mappings include Schwarz-Christoffel maps, osculation method, polynomial expansion method, circle packing method and many other methods. For details, we refer readers to [1,2] for a more thorough discussion.

With the development of 3D data acquisition technologies, a huge amount of surface and volumetric data has been accumulated. For example, the state

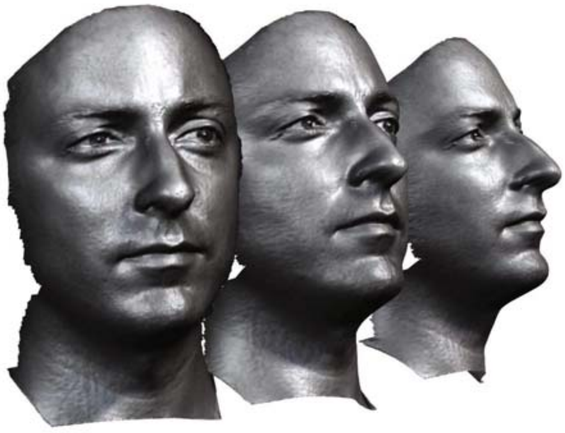


Fig. 1. Geometric surfaces captured by 3D scanner based on phase shifting technology

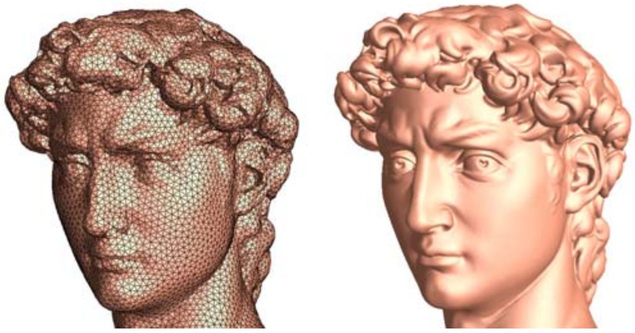


Fig. 2. Surfaces are represented as triangular meshes on computers

of the art 3D scanner based on phase shifting technology can capture dynamic surfaces with a quarter of million samples per frame, at the frame rate as high as 180 per second. Figure 1 illustrates one example of the acquired human face surface. These geometric surfaces are stored and represented in computers as polyhedral surfaces, most commonly simplicial complexes with a piecewise linear embedding in the Euclidean space, or triangular meshes in engineering field as shown in Figure 2.

The demands for processing discrete surfaces accurately and efficiently are pressing. Rigorous algorithms for surface matching, registration, classification have fundamental importance in almost every engineering field. It also becomes great challenges for mathematician to generalize the geometric theories from smooth manifolds to discrete settings.

With the great efforts of both mathematicians and computer scientists, major breakthroughs have been made in the last several years. Several theoretic frameworks have been systematically developed. The discrete counter parts of many fundamental theorems in classical conformal geometry have been discovered. Many computational algorithms have been invented and applied for engineering and medicine fields. There are still many profound facts in conformal geometry: the discretization method and the computational strategy are still widely open. Furthermore, the urge of practical applications has advanced the computational algorithms of this field, but the theoretic foundations need to be rigorously laid down in near future.

Generally speaking, there are two major approaches for computing conformal structures, one is along the harmonic analysis approach to construct various holomorphic or meromorphic differentials on the surface, then build conformal mappings and quasi-conformal mappings based on the differential forms; the other is along the surface Ricci curvature flow approach to design conformal Riemannian metrics which satisfy the prescribed curvatures. These two approaches each have advantages and disadvantages, and they are closely related. This work focuses on the recent developments of these approaches and their applications in practice.

2 Previous Work

Conventional computational complex analysis methods focus on conformal mappings on planar domains. A thorough introduction to the conventional methods can be found in the books [1] and [2].

Recently, with the development of digital scanning technology, computing conformal mappings between surfaces becomes more and more important. In computer graphics and discrete mathematics, much sound research has focused on discrete conformal mappings.

2.1 Holomorphic Differentials

The computational method of current work is mainly based on harmonic maps and holomorphic differential forms. Here, we briefly overview most related work, and refer readers to [3] and [4] for thorough surveys.

Discrete harmonic maps were constructed in [5], where the cotan formula was introduced. First order finite element approximations of the Cauchy-Riemann equations were introduced by Levy et al. [6]. Discrete intrinsic parameterization by minimizing Dirichlet energy was introduced by [7]. Mean value coordinates were introduced in [8] to compute generalized harmonic maps; Discrete spherical conformal mappings are used in [9] and [10].

Discrete holomorphic forms are introduced by Gu and Yau [11] to compute global conformal surface parameterizations for high genus surfaces. Another approach of discrete holomorphy was introduced in [12] using discrete exterior calculus [13]. The problem of computing optimal holomorphic 1-forms to reduce area distortion was considered in [14]. Gortler et al. [15] generalized 1-forms to the discrete case, using them to parameterize genus one meshes. Tong et al. [16]

generalized the 1-form method to incorporate cone singularities. Discrete one-forms have been applied for meshing point clouds in [17], surface tiling [18], surface quadrangulation [19]. The holomorphic 1-form method has been applied to virtual colonoscopy [20]. The colon surface is reconstructed from MRI images, and conformally mapped to the planar rectangle. This improves the efficiency and accuracy for detecting polyps. Conformal mapping is used for brain cortex surface morphology study in [10]. By mapping brain surfaces to spheres, cortex surface registration and comparison become straightforward. The holomorphic 1-form method has also been applied in computer vision [21, 22] for 3D shape matching, recognition and stitching. In geometric modeling field, constructing splines on general surfaces is one of the most fundamental problems. It is proven in [23] that if the surface has an affine structure, then splines can be generalized to it directly. Holomorphic 1-forms can be applied for computing the affine structures of general surfaces.

The Ricci flow on surfaces. The Ricci flow was introduced by R. Hamilton in a seminal paper [24] for Riemannian manifolds of any dimension. The Ricci flow has revolutionized the study of geometry of surfaces and 3-manifolds and has inspired huge research activities in geometry. In particular, it leads to a proof of the 3-dimensional Poincaré conjecture. In the paper [25], Hamilton used the 2-dimensional Ricci flow to give a proof of the uniformization theorem for surfaces of positive genus. This leads a way for potential applications to computer graphics.

There are many ways to discretize smooth surfaces. The one which is particularly related to a discretization of conformality is the circle packing metric introduced by Thurston [26]. The notion of circle packing has appeared in the work of Koebe [27]. Thurston conjectured in [28] that for a discretization of the Jordan domain in the plane, the sequence of circle packings converge to the Riemann mapping. This was proved by Rodin and Sullivan [29].

Colin de Verdiere [30] established the first variational principle for circle packing and proved Thurston's existence of circle packing metrics. This paved a way for a fast algorithmic implementation of finding the circle packing metric, such as the one by Collins and Stephenson [31]. In [32], Chow and Luo generalized Colin de Verdiere's work and introduced the discrete Ricci flow and discrete Ricci energy on surfaces. They proved a general existence and convergence theorem for the discrete Ricci flow and proved that the Ricci energy is convex. The algorithmic implementation of the discrete Ricci flow was carried out by Jin et al. [33].

Another related discretization method is called circle patterns; it considers both the combinatorics and the geometry of the original mesh, and can be looked as a variant of circle packings. Circle pattern was proposed by Bowers and Hurdal [34], and has been proven to be a minimizer of a convex energy by Bobenko and Springborn [35]. An efficient circle pattern algorithm was developed by Kharevych et al. [36].

The Yamabe flow on surfaces. The Yamabe problem aims at finding a conformal metric with constant scalar curvature for compact Riemannian manifolds.

The first proof (with flaws) was given by Yamabe [37], which was corrected and extended to a complete proof by several researchers including Trudinger [38], Aubin [39] and Schoen [40]. A comprehensive survey on this topic was given by Lee and Parker in [41].

In [42] Luo studied the discrete Yamabe flow on surfaces. He introduced a notion of discrete conformal change of polyhedral metric, which plays a key role in developing the discrete Yamabe flow and the associated variational principle in the field. Based on the discrete conformal class and geometric consideration, Luo gave the discrete Yamabe energy as an integration of a differential 1-form and proved that this energy is a locally convex function. He also deduced from it that the curvature evolution of the Yamabe flow is a heat equation.

In a very nice recent work by Springborn et al. [43] they were able to identify the Yamabe energy introduced by Luo with the Milnor-Lobachevsky function and the heat equation for the curvature evolution with the cotangent Laplace equation. They constructed an algorithm based on their explicit formula. Another recent work by Gu et al [44], which used the original discrete Yamabe energy from [42], has produced an equally efficient algorithm in finding the discrete conformal metrics. In addition, discrete hyperbolic Yamabe Flow was presented in [45] for computing hyperbolic structure and the canonical homotopy class representative.

3 Theoretic Background

This section reviews the preliminary theoretic background for conformal geometry.

3.1 Harmonic Maps

Suppose $f : S \to \mathbb{R}$ is a function defined on a surface S: the *harmonic energy* of f is defined as

$$E(f) = \int_S |\nabla f|^2 ds,$$

where ∇f is the gradient of f. A *harmonic function* is a critical point of the harmonic energy. The harmonic function satisfies the following Laplace equation

$$\Delta f = 0,$$

where Δ is the Laplace-Beltrami operator. If S is a domain on the Euclidean plane, then the Laplace-Beltrami operator has the form

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

A harmonic map between two surfaces can be defined similarly, which is a critical point of the harmonic energy. Suppose $\phi : S_1 \to S_2$ is a harmonic map, S_2 is a convex planar domain, and the restriction of ϕ on the boundaries $\phi_{\partial S_1} : \partial S_1 \to \partial S_2$ is a homeomorphism, then ϕ is a diffeomorphism in the interior. Figure 3 shows one such kind of map.

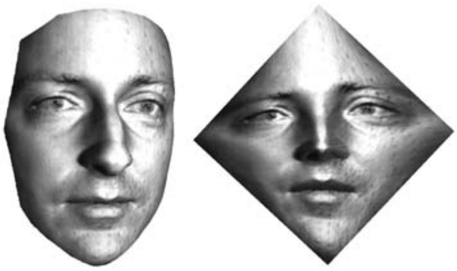


Fig. 3. A harmonic map from a face surface to a planar convex domain

3.2 Conformal Mappings

Let S_1 and S_2 be two surfaces with Riemannian metrics $\mathbf{g}_1$ and $\mathbf{g}_2$ and let $\phi : (S_1, \mathbf{g}_1) \rightarrow (S_2, \mathbf{g}_2)$ be a homeomorphism between them. We say that ϕ is *conformal*, if it preserves angles. In details, as shown in Figure 4, let $\gamma_1, \gamma_2 : [0, 1] \rightarrow S_1$ be two arbitrary curves on S_1, intersecting at the point p, and the angle between the two tangent vectors $\frac{d\gamma_1}{dt}(p)$ and $\frac{d\gamma_2}{dt}(p)$ be equal to θ. Therefore $\phi \circ \gamma_1(t)$ and $\phi \circ \gamma_2(t)$ are two curves on S_2, intersecting at $\phi(p)$. Then their intersection angle also equals θ.

Locally, conformal mapping is a scaling transformation; it preserves local shapes. For example, it maps infinitesimal circles to infinitesimal circles. As shown in Figure 5 frame (a), the bunny surface is mapped to the plane via a conformal mapping. A circle packing is defined on the plane, and pulled back onto the bunny surface, and all small circles are preserved. If we put a checkerboard on the plane, then on the bunny surface, all the right angles of the checkers are well preserved, as illustrated in the same figure frame (a).

Mathematically, the pull back metric induced by ϕ differs from the original metric by a scaling function, which is called the *conformal factor* and measures the area distortion.

$$\phi_*(\mathbf{g}_2) = e^{2u}\mathbf{g}_1.$$

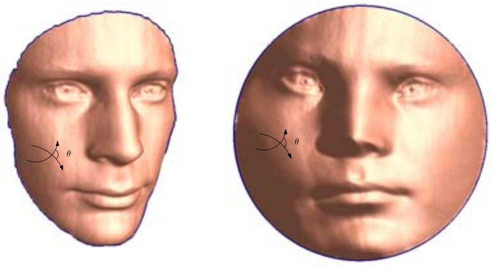


Fig. 4. Conformal mappings preserve angles

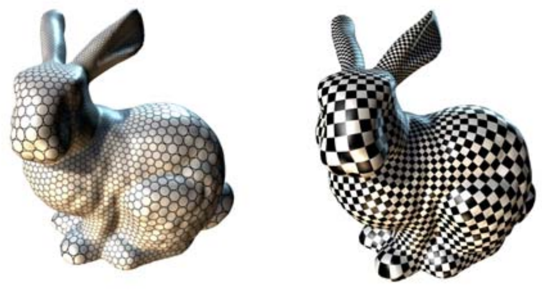

(a) Circle Packing (b) Checker board

Fig. 5. Conformal texture mapping

Conformal mappings have deep relation with complex analysis. Conformal mappings between two planar domains can be represented as holomorphic functions. Let $\phi : \mathbb{C} \to \mathbb{C}$ be a complex function, $\phi : z \to w$, where $z = x + iy$, $w = u + iv$, then ϕ is *holomorphic*, if it satisfies the following Riemann-Cauchy equation

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

If ϕ is invertible and the inverse is also holomorphic, then ϕ is *bi-holomorphic.*

3.3 Conformal Structure

Let S be a topological surface, U_α be an open set on S, $\phi_\alpha : U_\alpha \to \mathbb{C}$ be a homeomorphism, then (U_α, ϕ_α) is a local chart of S. Suppose (U_β, ϕ_β) is another local chart overlapping with (U_α, ϕ_α), the transition function $\phi_{\alpha\beta} : \phi_\alpha(U_\alpha \cap U_\beta) \to \phi_\beta(U_\alpha \cap U_\beta)$ is given by $\phi_\beta \circ \phi_\alpha^{-1}$. An atlas is a collection of local charts $\{(U_\alpha, \phi_\alpha)\}$, such that the union of the charts cover the surface, as shown in Figure 6.

An atlas is a *conformal atlas* if all its transition functions are biholomorphic. Two conformal atlases are equivalent if their union is still a conformal atlas. Each equivalence class of conformal atlases is called a *conformal structure* of the surface. A surface with a conformal structure is called a *Riemann surface.*

If a Riemann surface S has a Riemannian metric $\mathbf{g}$, then we say its conformal structure is compatible with the metric if local representation of the metric on a chart (U_α, ϕ_α)

$$\mathbf{g} = e^{2u} dz_\alpha d\bar{z}_\alpha,$$

where z_α is the local complex parameter. We also call such local complex parameters *isothermal coordinates.* The *Laplace-Beltrami* operator Δ on an isothermal coordinates has a simple representation

$$\Delta = \frac{1}{e^{2u(x_\alpha, y_\alpha)}} \left(\frac{\partial^2}{\partial x_\alpha^2} + \frac{\partial^2}{\partial y_\alpha^2}\right).$$

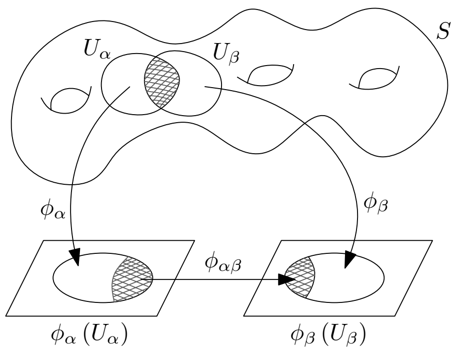


Fig. 6. Conformal structure

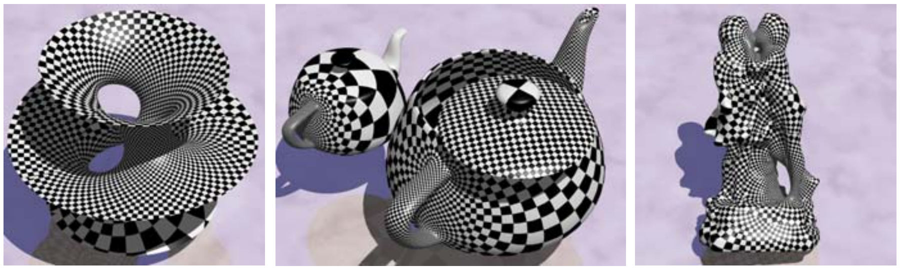


Fig. 7. Conformal structures (isothermal coordinates) for surfaces in real life

All oriented metric surfaces are Riemann surfaces and with isothermal coordinates. Therefore conformal geometric concepts and methods are general to all surfaces in real life. Figure 7 shows the isothermal coordinates on surfaces from real life.

Suppose $(S_1, \mathcal{A}_1)$ and $(S_2, \mathcal{A}_2)$ are two Riemann surfaces, $\mathcal{A}_i$'s are their conformal structures, (U_α, ϕ_α) is a local chart of $\mathcal{A}_1$ and (V_β, ψ_β) is a local chart of $\mathcal{A}_2$, then $\phi : S_1 \to S_2$ is a *conformal map* if and only if

$$\psi_\beta \circ \phi \circ \phi_\alpha^{-1} : \phi_\alpha(U_\alpha) \to \psi_\beta(V_\beta)$$

is biholomorphic. A conformal map preserves angles.

3.4 Holomorphic Differentials

Differential forms play important role in conformal geometry. A function $f : S \to \mathbb{R}$ is called a *0-form*. The exterior differential operator d is the generalization of conventional grad, curl and divergence operators. The exterior differentiation of a 0-form is its gradient. Let (U_α, ϕ_α) be a local chart with local parameters (x_α, y_α), then

$$d_0 f = \frac{\partial f}{\partial x_\alpha} dx_\alpha + \frac{\partial f}{\partial y_\alpha} dy_\alpha.$$

A *differential 1-form* has local representation

$$\omega = f_\alpha dx_\alpha + g_\alpha dy_\alpha,$$

and also has local representation $f_\beta dx_\beta + g_\beta dy_\beta$, such that

$$(f_\alpha \; g_\alpha) \begin{pmatrix} \frac{\partial x_\alpha}{\partial x_\beta} & \frac{\partial x_\alpha}{\partial y_\beta} \\ \frac{\partial y_\alpha}{\partial x_\beta} & \frac{\partial y_\alpha}{\partial y_\beta} \end{pmatrix} = (f_\beta \; g_\beta).$$

The exterior differentiation of a 1-form is its circulation

$$d_1\omega = (\frac{\partial g}{\partial x_\alpha} - \frac{\partial f}{\partial y_\alpha})dx_\alpha \wedge dy_\alpha.$$

A differential 2-form has local representation

$$\Omega = f_\alpha dx_\alpha \wedge dy_\alpha.$$

On another chart (x_β, y_β) and its associated representation $\Omega = f_\beta dx_\beta \wedge dy_\beta$, the two f_α and f_β satisfy

$$f_\alpha \begin{vmatrix} \frac{\partial x_\alpha}{\partial x_\beta} & \frac{\partial x_\alpha}{\partial y_\beta} \\ \frac{\partial y_\alpha}{\partial x_\beta} & \frac{\partial y_\alpha}{\partial y_\beta} \end{vmatrix} = f_\beta.$$

The exterior differentiation of a 2-form is zero, $d_2\Omega = 0$.

Let ω be a k-form: if $d_k\omega = 0$, ω is called a *closed form.* If there exists a $(k-1)$-form τ, such that $\omega = d_{k-1}\tau$, then ω is called an *exact form.* A fundamental fact is that

$$d_k \circ d_{k-1} \equiv 0, k = 1, 2$$

therefore all exact forms are closed. The difference between exact forms and closed forms conveys the topological information of the surface. The de Rham cohomology group is defined as

$$H^k(S, \mathbb{R}) = \frac{Ker\ d_k}{Img\ d_{k-1}}.$$

If two closed k-forms, ω_1, ω_2, are cohomologous, then they differ by an exact form $d\tau = \omega_1 - \omega_2$, for some $(k-1)$-form τ.

Let Ω_0 be the area element on S with local representation $dx_\alpha \wedge dy_\alpha$, ω be a k-form. The *Hodge star* operator converts ω to a $(n-k)$-form, ${}^*\omega$, such that

$$\omega \wedge {}^*\omega = \Omega_0.$$

We say ${}^*\omega$ is conjugate to ω.

The co-differentiation operator δ is defined as

$$\delta = *d* .$$

A *harmonic form* ω satisfies the following condition

$$d\omega = 0, \delta\omega = 0.$$

A *holomorphic 1-form* is a pair of harmonic 1-forms, the imaginary part is conjugate to the real part:

$$\tau = \omega + i^*\omega.$$

On the conformal atlas, τ has a local representation on (U_α, ϕ_α) with local parameter z_α,

$$\tau = f_\alpha(z_\alpha)dz_\alpha,$$

where f_α is a holomorphic function. On another chart (U_β, ϕ_β) with local parameter z_β, $\tau = f_\beta dz_\beta$, such that

$$f_\alpha \frac{dz_\alpha}{dz_\beta} = f_\beta.$$

The Hodge theorem claims that each cohomologous class has a unique harmonic 1-form. Therefore the group of all harmonic 1-forms is isomorphic to the first cohomology group $H^1(S, \mathbb{R})$. Also, the group of all holomorphic 1-forms is isomorphic to $H^1(S, \mathbb{R})$.

3.5 Surface Ricci Flow

Let S be a topological surface. Consider all Riemannian metrics on S. Two metrics $\mathbf{g}_1, \mathbf{g}_2$ are *conformal equivalent*, if there is a function $u : S \to \mathbb{R}$ defined on the surface, such that $\mathbf{g}_2 = e^{2u}\mathbf{g}_1$. Each conformal equivalence class of Riemannian metrics is a *conformal structure* of S.

The Gaussian curvatures determined by $\mathbf{g}_1$ and $\mathbf{g}_2$ are denoted as K_1 and K_2, they are related by the following *Yamabe equation*

$$K_2 = \frac{1}{e^{2u}}(-\Delta u + K_1),$$

where Δ is the Laplace-Beltrami operator determined by the metric $\mathbf{g}_1$.

Hamilton introduced the surface Ricci flow

$$\frac{dg_{ij}}{dt} = -Kg_{ij}.$$

During the flow, if the total area is preserved, then the Gaussian curvature will evolve according to a heat diffusion process. Hamilton and Chow together proved that the surface Ricci flow converges to a special metric, whose Gaussian curvature is constant everywhere. In fact, their proofs give another approach for the Poincaré uniformization theorem, which states that for any metric surface $(S, \mathbf{g})$, there exists a metric conformal to the original metric, that induces constant Gaussian curvature. As shown in Figure 8, all closed surfaces can be conformally deformed to three canonical shapes, the unit sphere $\mathbb{S}^2$, the plane $\mathbb{E}^2$ and the hyperbolic disk $\mathbb{H}^2$.

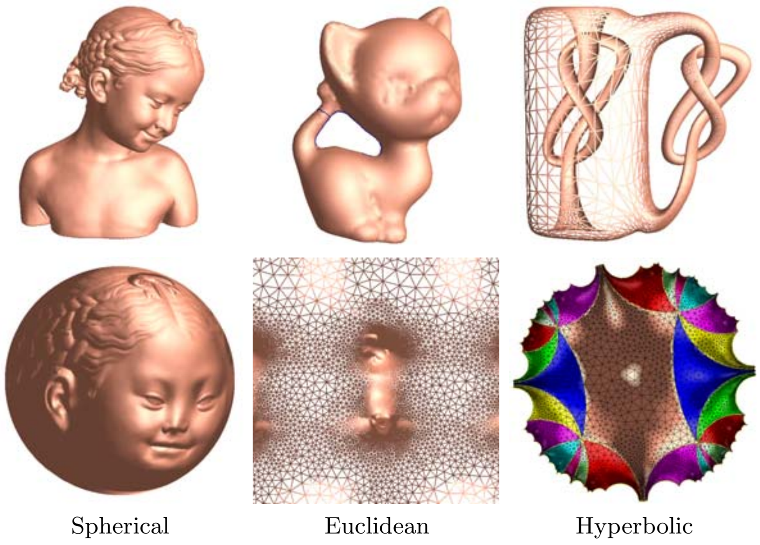


Fig. 8. Surface uniformization theorem

3.6 Quasi-Conformal Maps

A generalization of the conformal map is called the *quasi-conformal* map which is an orientation-preserving homeomorphism between Riemann surfaces with bounded conformality distortion, in the sense that the first order approximation of the quasi-conformal homeomorphism takes small circles to small ellipses of bounded eccentricity. Thus, a conformal homeomorphism that maps a small circle to a small circle can also be regarded as quasi-conformal.

Mathematically, ϕ is quasi-conformal provided that it satisfies Beltrami's equation 1 on a local chart for some complex valued Lebesgue measurable μ satisfying $|\mu|_\infty < 1$,

$$\frac{\partial \phi}{\partial \bar{z}} = \mu(z)\frac{\partial \phi}{\partial z}, \tag{1}$$

μ is called the *Beltrami coefficient*, which is a measure of conformality. In particular, the map ϕ is conformal around a small neighborhood of p when $\mu(p) = 0$. In general, ϕ maps an infinitesimal circle to a infinitesimal ellipse. From $\mu(p)$, we can determine the angles of the directions of maximal magnification as well as the amount of maximal magnification and maximal shrinking. Specifically, the angle of maximal magnification is $arg\mu(p)/2$ with magnifying factor $1 + |\mu(p)|$; the angle of maximal shrinking is the orthogonal angle $(arg\mu(p) - \pi)/2$ with shrinking factor $1 - |\mu(p)|$. The distortion or dilation is given by:

$$K = \frac{1 + |\mu(p)|}{1 - |\mu(p)|}. \tag{2}$$

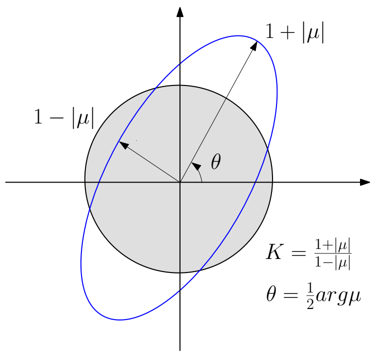
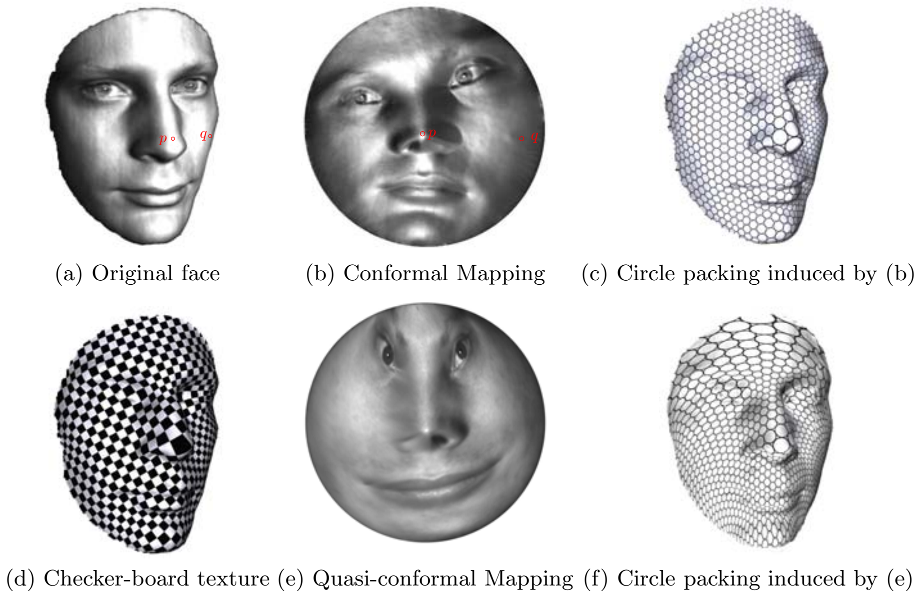


Fig. 9. Illustration of how Beltrami coefficient μ measures the distortion by a quasi-conformal map that is an ellipse with dilation K



(a) Original face (b) Conformal Mapping (c) Circle packing induced by (b)

(d) Checker-board texture (e) Quasi-conformal Mapping (f) Circle packing induced by (e)

Fig. 10. Conformal and quasi-conformal mappings for a topological disk

Thus, the Beltrami coefficient μ gives us all the information about the conformality of the map (see Figure 9).

In terms of the metric tensor, considering the effect of the pullback of the canonical Euclidean metric $\mathbf{g}_0$ under ϕ, the resulting metric is given by:

$$\phi^*(\mathbf{g}_0) = |\frac{\partial \phi}{\partial z}|^2 |dz + \mu(z) d\overline{z})|^2. \quad (3)$$

Figure 10 shows the conformal and quasi-conformal mappings for a topological disk, where the Beltrami coefficient is set to be $\mu = z$. From the texture mappings in frames (c) and (d), we can see that the conformality is kept well around nose tip (circles to circles), while it is changed a lot along the boundary area (circles to quasi-ellipses).

3.7 Teichmüller Space

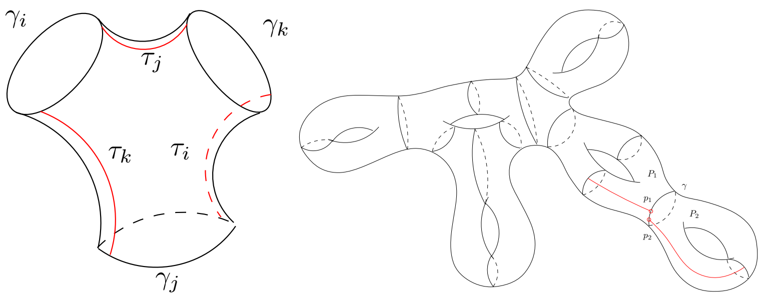
Given a topological surface S with genus $g > 1$, all the conformal structures on S form the *Moduli space*. In general, this space is complicated to compute. Instead, we study its universal covering space *Teichmüller space*.

Let S be a Riemann surface of genus $g > 1$. We mark a set of fundamental group generators $\{a_1, a_2, \cdots, a_g, b_1, b_2, \cdots, b_g\}$, then S is a *marked Riemann surface*. Two given marked Riemann surfaces are equivalent, if there is a conformal (bi-holomorphic) map $\phi : S_1 \to S_2$, such that ϕ is isotopic to the identity up to the marking. Each equivalence class is represented as one point in the Teichmüller space T_g.

The dimension of Teichmüller space is 2 for genus one surfaces, and $6g - 6$ if $g > 1$. Because each conformal structure has a unique hyperbolic metric, it is enough to consider surfaces with hyperbolic metrics for computing the Teichmüller space.

Assuming that S has a hyperbolic metric, its Fenchel-Nielsen coordinates in T_g can be constructed as follows. Given a genus g surface, it can be decomposed to $2g - 2$ pairs of pants. Figure 11 illustrates one example. Assume all the cutting loops are geodesics $\{\gamma_1, \gamma_2, \cdots, \gamma_{3g-3}\}$, then each pair of pants is a pair of hyperbolic pants.

On each pair of hyperbolic pants P with three boundaries $\gamma_i, \gamma_j, \gamma_k$ there are three shortest paths connecting each pair of boundaries, e.g. τ_i connects γ_j, γ_k. Then τ_i, τ_j, τ_k separate S into two congruent hyperbolic hexagons with right corner angles.



(a) A pair of pants (b) Pants decomposition

Fig. 11. A surface of genus g with a hyperbolic metric is decomposed to $2g-2$ pairs of pants by cutting along closed geodesics. The twisting angle and length of each cutting loop give the Fenchel-Nielsen coordinates in the shape space.

Suppose two pairs of hyperbolic pants P_1 and P_2 are glued together along γ. The shortest path τ_1 on P_1 intersects γ at p_1, the shortest path τ_2 on P_2 intersects γ at p_2, then the *twisting angle* on γ is given by

$$\theta = 2\pi \frac{d(p_1, p_2)}{|\gamma|}$$

where $d(p_1, p_2)$ is the geodesic distance between p_1 and p_2, $|\gamma|$ is the length of γ. The Fenchel-Nielsen Coordinates are given by the lengths of γ_k and the twisting angles on γ_k, $k = 1, 2, \cdots, 3g - 3$.

4 Computational Methods

4.1 Harmonic Maps

Harmonic maps in $\mathbb{R}^3$ can be computed using heat flow method. For example, in case we want to compute a harmonic map from a genus zero closed surface to the unit sphere $\phi : S \to \mathbb{S}^2$. We can initialize the map by the canonical Gauss map, then minimize the harmonic energy by the heat flow. First we compute the Laplacian of the map $\Delta\phi : S \to \mathbb{R}^3$. Then we compute the tangential component of the Laplacian. Suppose $p \in S$, then $\phi(p) \in \mathbb{S}^2$.

$$\Delta^{\perp}\phi(p) =< \Delta^{\perp}\phi(p), \phi(p) > \phi(p).$$

The tangential component of the Laplacian is given by

$$\Delta^{\|}\phi(p) = \Delta\phi(p) - \Delta^{\perp}\phi(p).$$

The heat flow is defined as

$$\frac{d\phi(p,t)}{dt} = -\Delta^{\|}\phi(p).$$

Because the harmonic maps are not unique, they differ by a Möbius transformation on the sphere, special normalization condition needs to be added during the flow. The following is a common condition,

$$\int_S \phi(p) ds = 0.$$

For genus zero closed surfaces, harmonic maps are conformal. Figure 12 shows one example computed using this method.

For a genus zero surface with a single boundary, we can convert it to a symmetric closed surface by double covering. Then by mapping the doubled surface to the unit sphere, we can compute the conformal mapping of the original surface. The mapping is not unique; figure 13 shows that two such conformal mappings differ by a Möbius transformation.

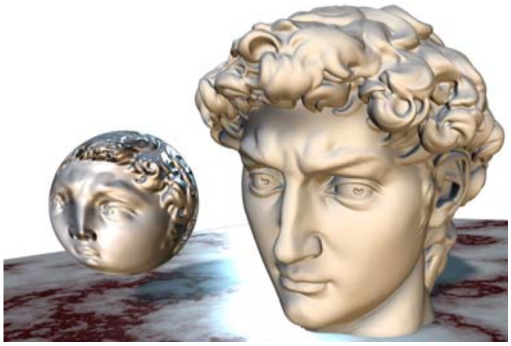


Fig. 12. Harmonic map from a genus zero closed surface to the unit sphere

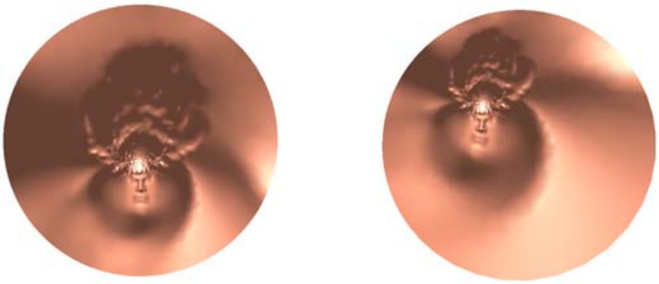


Fig. 13. Conformal mappings from a topological disk to the unit disk differ by a Möbius transformation

4.2 Holomorphic Differential Approach

Homology Basis. Given a surface S embedded in $\mathbb{R}^3$, we first compute its fundamental group generators. We compute its CW-cell decomposition

$$S_0 \subset S_1 \subset S_2 = S,$$

where $S_k = S_{k-1} \cup D_k^1 \cup D_k^2 \cdots \cup D_k^n$, and D_k^i are k-dimensional cells (disks), such that the boundaries of these cells are on S_{k-1},

$$\partial D_k^i \subset S_{k-1}.$$

Then the fundamental group of S_1 has the same generators as the fundamental group of S. Then we compute a spanning tree T of S_0 in S_1, the complement of T in S_1 are disconnected 1-cells, denoted as $e_1, e_2, \cdots e_k$, then the union of T and e_i has a unique loop γ_i. All such loops $\{\gamma_1, \gamma_2, \cdots, \gamma_{2g}\}$ form a basis for the fundamental group $\pi_1(S)$. These loops also form a basis of the first homology basis $H_1(S, \mathbb{Z})$. Figure 14 shows the homology group generators of a genus two surface.

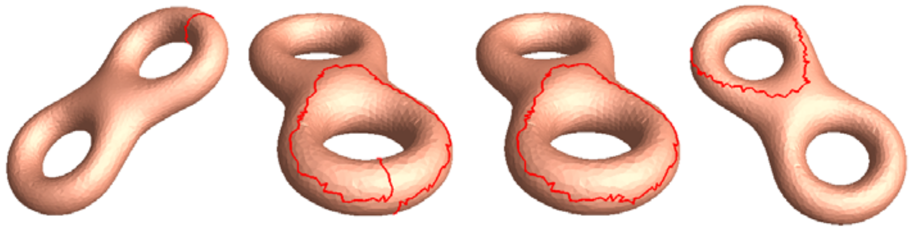


Fig. 14. Computing homology group basis

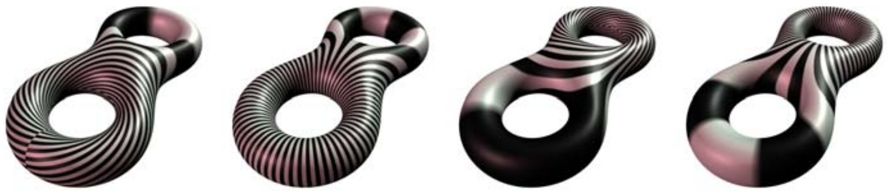


Fig. 15. Computing harmonic 1-form group basis

Cohomology Basis. Let γ_k be a base loop for $H_1(S,\mathbb{Z})$, then we slice S along γ_k to get an open surface S_k, such that the boundary of S_k is given by

$$\partial S_k = \gamma_k^+ - \gamma_k^-,$$

$\gamma_k^+, \gamma_k^{-1}$ are the two boundary loops on S_k. Then we randomly construct a function $h_k : S_k \to \mathbb{R}$, such that

$$h_k(p) = 1, \forall p \in \gamma_k^+; \quad h_k(p) = 0, \forall p \in \gamma_k^-;$$

and $h_k(p)$ is random for all interior points on S_k. Then dh_k is an exact 1-form on S_k. Because of the consistency along the boundaries, dh_k is also a closed 1-form on S. We denote τ_k as dh_k on S. Then $\{\tau_1, \tau_2, \cdots, \tau_{2g}\}$ form a basis for $H^1(S,\mathbb{R})$.

Harmonic 1-form Basis. According to Hodge theory, for each closed 1-form τ_k, there exists a 0-form $g_k : S \to \mathbb{R}$, such that $\tau_k + dg_k$ is a harmonic 1-form. The 0-form g_k can be obtained by solving

$$d * (\tau_k + dg_k) = 0.$$

We denote the harmonic 1-form as $\omega_k = \tau_k + dg_k$. Figure 15 shows the harmonic group generators of a genus two surface.

Holomorphic 1-form Basis. Holomorphic 1-form can be constructed by harmonic 1-form and its conjugate $\omega_k + i^*\omega_k$. Then $\{\omega_1 + i^*\omega_1, \omega_2 + i^*\omega_2, \cdots, \omega_{2g} + i^*\omega_{2g}\}$

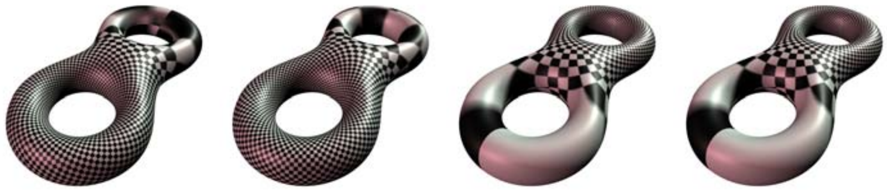


Fig. 16. Computing holomorphic 1-form group basis

form a basis for holomorphic 1-form group. Figure 16 shows the holomorphic 1-form group basis for the genus two surface.

For surfaces with boundaries, we can convert the surface to a symmetric closed surface by the double covering technique.

4.3 Discrete Surface Ricci Flow

In the engineering field, smooth surfaces are often approximated by simplicial complexes (triangle meshes). Major concepts, such as metric, curvature, and conformal deformation in the continuous setting can be generalized to the discrete setting. We denote a triangle mesh as Σ, a vertex set as V, an edge set as E, and a face set as F. e_{ij} represents the edge connecting vertices v_i and v_j, and f_{ijk} denotes the face formed by v_i, v_j, and v_k.

Background Geometry. In the engineering field, it is always assumed that a mesh Σ is embedded in the three dimensional Euclidean space $\mathbb{R}^3$, and therefore each face is Euclidean. In this case, we say the mesh is with Euclidean background geometry. The angles and edge lengths of each face satisfy the Euclidean cosine law.

Similarly, we can assume that when a mesh is embedded in the three dimensional sphere $\mathbb{S}^3$ or hyperbolic space $\mathbb{H}^3$, then each face is a spherical or a hyperbolic triangle. We say the mesh is with spherical or hyperbolic background geometry. The angles and the edge lengths of each face satisfy the spherical or hyperbolic cosine law.

Discrete Riemannian Metric. A discrete Riemannian metric on a mesh Σ is a piecewise constant metric with cone singularities. A metric on a mesh with Euclidean metric is a discrete Euclidean metric with cone singularities. Each vertex is a cone singularity. Similarly, a metric on a mesh with spherical background geometry is a discrete spherical metric with cone singularities; a metric on a mesh with hyperbolic background geometry is a discrete hyperbolic metric with cone singularities.

The edge lengths of a mesh Σ are sufficient to define a discrete Riemannian metric,

$$l : E \to \mathbb{R}^+, \qquad (4)$$

as long as, for each face f_{ijk}, the edge lengths satisfy the triangle inequality: $l_{ij} + l_{jk} > l_{ki}$ for all the three background geometries, and another inequality: $l_{ij} + l_{jk} + l_{ki} < 2\pi$ for spherical geometry.

Discrete Gaussian Curvature. The discrete Gaussian curvature K_i at a vertex $v_i \in \Sigma$ can be computed from the angle deficit,

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, \, v_i \notin \partial\Sigma \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, \, v_i \in \partial\Sigma \end{cases} \tag{5}$$

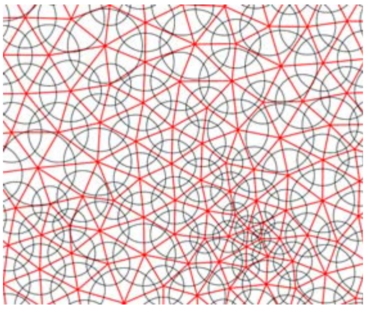
where θ_i^{jk} represents the corner angle attached to vertex v_i in the face f_{ijk}, and $\partial\Sigma$ represents the boundary of the mesh. The discrete Gaussian curvatures are determined by the discrete metrics.

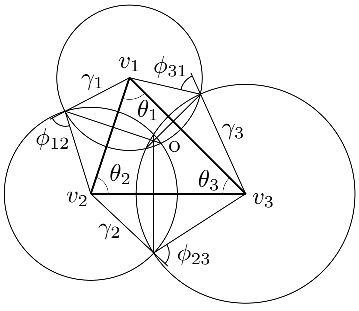
Discrete Gauss-Bonnet Theorem. The Gauss-Bonnet theorem states that the total curvature is a topological invariant. It still holds on meshes as follows.

$$\sum_{v_i \in V} K_i + \lambda \sum_{f_i \in F} A_i = 2\pi\chi(M), \tag{6}$$

where A_i denotes the area of face f_i, and λ represents the constant curvature for the background geometry; $+1$ for the spherical geometry, 0 for the Euclidean geometry, and -1 for the hyperbolic geometry.

Circle Packing Metric. The concept of the circle packing metric was introduced by Thurston in [46] as shown in Figure 17. Let Γ be a function defined on the vertices, $\Gamma : V \to \mathbb{R}^+$, which assigns a radius γ_i to the vertex v_i. Similarly, let Φ be a function defined on the edges, $\Phi : E \to [0, \frac{\pi}{2}]$, which assigns an acute angle $\Phi(e_{ij})$ to each edge e_{ij} and is called a *weight* function on the edges. Geometrically, $\Phi(e_{ij})$ is the intersection angle of two circles centered at v_i and v_j. The pair of vertex radius function and edge weight function on a mesh Σ, (Γ, Φ), is called a *circle packing metric* of Σ. Two circle packing metrics (Γ_1, Φ_1) and (Γ_2, Φ_2) on the same mesh are *conformally equivalent* if $\Phi_1 \equiv \Phi_2$.


(a)



(b)

Fig. 17. Circle Packing Metric. (a) Flat circle packing metric (b) Circle packing metric on a triangle.

Admissible Curvature Space. A mesh Σ with edge weight Φ is called a *weighted mesh*, which is denoted as (Σ, Φ). In the following, we want to clarify the spaces of all possible circle packing metrics and all possible curvatures of a weighted mesh.

Let the vertex set be $V = \{v_1, v_2, \cdots, v_n\}$, and the radii $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_n\}$. Let u_i be

$$u_i = \begin{cases} \log \gamma_i & \mathbb{E}^2 \\ \log \tanh \frac{\gamma_i}{2} & \mathbb{H}^2 \\ \log \tan \frac{\gamma_i}{2} & \mathbb{S}^2 \end{cases} \tag{7}$$

where $\mathbb{E}^2$, $\mathbb{H}^2$, and $\mathbb{S}^2$ indicate the background geometry of the mesh. We represent a circle packing metric on (Σ, Φ) by a vector $\mathbf{u} = (u_1, u_2, \cdots, u_n)^T$. Similarly, we represent the Gaussian curvatures at mesh vertices by the curvature vector $\mathbf{k} = (K_1, K_2, \cdots, K_n)^T$. All the possible $\mathbf{u}$'s form the *admissible metric space*, and all the possible $\mathbf{k}$'s form the *admissible curvature space.*

According to the Gauss-Bonnet theory (see Eqn. 6), the total curvature must be $2\pi\chi(\Sigma)$, and therefore the curvature space is $n-1$ dimensional. We add one linear constraint to the metric vector $\mathbf{u}$, $\sum u_i = 0$, for the normalized metric. As a result, the metric space is also $n-1$ dimensional. If all the intersection angles are acute, then the edge lengths induced by a circle packing satisfy the triangle inequality. There is no further constraint on $\mathbf{u}$. Therefore, the admissible metric space is simply $\mathbb{R}^{n-1}$.

A curvature vector $\mathbf{k}$ is *admissible* if there exists a metric vector $\mathbf{u}$, which induces $\mathbf{k}$. The admissible curvature space of a weighted mesh (Σ, Φ) is a convex polytope, specified by the following theorem. The detailed proof can be found in [32].

The admissible curvature space for weighted meshes with hyperbolic or spherical background geometries is more complicated. We refer the readers to [47] for detailed discussion.

Ricci Flow. Suppose (Σ, Φ) is a weighted mesh with an initial circle packing metric. The discrete Ricci flow is defined as follows.

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i), \tag{8}$$

where $\bar{\mathbf{k}} = (\bar{K}_1, \bar{K}_2, \cdots, \bar{K}_n)^T$ is the user defined target curvature. The discrete Ricci flow has exactly the same form as the smooth Ricci flow, which deforms the circle packing metric according to the Gaussian curvature, as in Eqn. 8.

The discrete Ricci flow can be formulated in the variational setting, namely, it is a negative gradient flow of a special energy form. The energy is given by

$$f(\mathbf{u}) = \int_{\mathbf{u_0}}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i, \tag{9}$$

where $\mathbf{u_0}$ is an arbitrary initial metric. The energy is called the *discrete Ricci energy.*

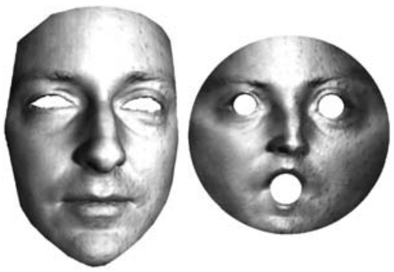


Fig. 18. Euclidean Ricci flow method to compute a conformal mapping for a multiply connected domain

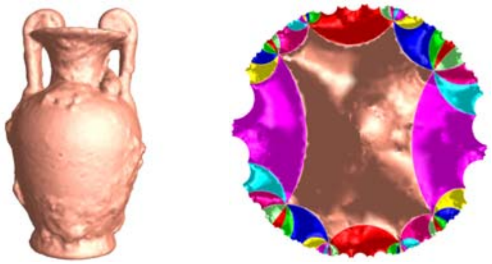


Fig. 19. Hyperbolic Ricci flow method to compute a conformal mapping for a genus two surface

Computing the desired metric with user-defined curvature $\bar{\mathbf{k}}$ is equivalent to minimizing the discrete Ricci energy. For Euclidean or hyperbolic cases, the discrete Ricci energy (see Eqn. 9) was first proved to be strictly convex in the seminal work of Colin de Verdiere [30] for the $\Phi = 0$ case, and was generalized to all cases of $\Phi \leq \pi/2$ in [32]. The global minimum uniquely exists, corresponding to the metric $\bar{\mathbf{u}}$, which induces $\bar{\mathbf{k}}$. The discrete Ricci flow converges to this global minimum. Although the spherical Ricci energy is not strictly convex, the desired metric $\bar{\mathbf{u}}$ is still a critical point of the energy.

The energy can be optimized using Newton's method. As shown in figure 17 (b), for each face $[v_i, v_j, v_k]$ there exists a unique circle orthogonal to all three circles at the vertices, whose center is o. The distance from the center to edge $[v_i, v_j]$ is denoted as d_{ij}^k. The weight for an edge $[v_i, v_j]$ adjacent to $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$ is defined as

$$\mu_{ij} = d_{ij}^k + d_{ij}^l.$$

The Hessian matrix $H = (h_{ij})$ is given by the discrete Laplace form

$$h_{ij} = \begin{cases} 0, \, [v_i, v_j] \notin E \\ d_{ij}, \, i \neq j \\ \sum_k d_{ik}, \, i = j \end{cases}$$

Figure 18 shows one example for computing the conformal mapping of a multiply connected domain onto the plane. Figure 19 shows the result for computing the uniformization hyperbolic metric for a genus two surface. The universal covering space of the surface with the uniformization metric is isometrically embedded in the Poicaré disk.

4.4 Discrete Surface Yamabe Flow

For smooth surfaces, the Ricci flow and Yamabe flow are equivalent. Using the symbols in the previous discussion, let Σ be a triangle mesh embedded in $\mathbb{R}^3$. Let e_{ij} be an edge with end vertices v_i and v_j. d_{ij} is the edge length of e_{ij} induced by the Euclidean metric of $\mathbb{R}^3$. A function defined on the vertices $\mathbf{u} : V \to \mathbb{R}$ is the *discrete conformal factor*. The edge length l_{ij} is defined as

$$l_{ij} = e^{u_i + u_j} d_{ij}. \tag{10}$$

Let K_i and $\bar{K}_i$ denote the current vertex curvature and the target vertex curvature respectively. The discrete Yamabe flow is defined as

$$\frac{du_i(t)}{dt} = \bar{K}_i - K_i, \tag{11}$$

with initial condition $u_i(0) = 0$. The convergence of Yamabe flow is proven in [42]. Furthermore, Yamabe flow is the gradient flow of the following Yamabe energy, let $\mathbf{u} = (u_1, u_2, \cdots, u_n)$,$n$ is the total number of vertices,

$$f(\mathbf{u}) = \int_{\mathbf{u}_0}^{\mathbf{u}} \sum_i^n (\bar{K}_i - K_i) du_i. \tag{12}$$

The Yamabe energy is well defined and convex. The Hessian matrix can be easily constructed as follows. Suppose faces $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$ are adjacent to the edge e_{ij}, define the *weight* of the edge e_{ij} as

$$w_{ij} = \cot\theta_k + \cot\theta_l, \tag{13}$$

where θ_k is the angle at v_k in f_{ijk}, θ_l is the angle at v_l in face f_{jil}. If the edge is on the boundary, and only attaches to f_{ijk}, then

$$\mu_{ij} = \cot\theta_k.$$

It can be shown by direct computation, the differential relation between the curvature and the conformal factor is

$$dK_i = \sum_j \mu_{ij}(du_i - du_j). \tag{14}$$

So the Hessian matrix of the yamabe energy $H = (h_{ij})$ is given by

$$h_{ij} = \begin{cases} 0 & , [v_i, v_j] \notin E \\ \mu_{ij} & , i \neq j \\ -\sum_k \mu_{ik} & , i = j \end{cases} \tag{15}$$

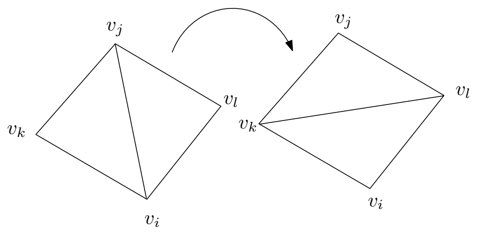


Fig. 20. Edge swap

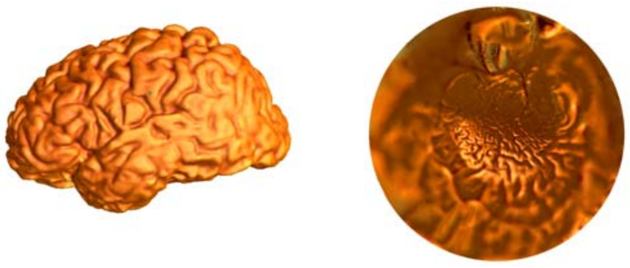


Fig. 21. Conformal brain mapping using Yamabe flow method

The Hessian matrix is positive definite on the linear subspace $\sum_i u_i = 0$. By using the Hessian matrix (see Eqn. 15), the Yamabe energy (see Eqn. 12) can be optimized effectively. But the major difficulty is that the *admissible metric space* $\Omega(\mathbf{u})$ for a mesh with fixed connectivity is not convex,

$$\Omega(\mathbf{u}) = \{\mathbf{u} | \forall f_{ijk} \in M, l_{ij} + l_{jl} > l_{li}\}$$

Therefore, during the optimization process using Newton's method, we need to ensure that the metric $\mathbf{u}$ is in the admissible metric space $\Omega(u)$ at each step. If a degenerated triangle f_{ijk} is detected, then we swap the longest edge of it. For example, if θ_k exceeds π, then we swap edge e_{ij} as shown in Figure 20. The major difficulty for the discrete Ricci flow is to find a good initial circle packing with all acute edge intersection angles. This problem does not exist for discrete Yamabe flow. Therefore, Yamabe flow in general produces better conformality in practice. Figure 21 shows the conformal brain mapping result using Yamabe flow method.

4.5 Quasi-Conformal Mapping by Solving Beltrami Equations

Given a surface S, with a conformal structure, also given a measurable complex value function defined on the surface $\mu : S \to \mathbb{C}$, we want to find a quasi-conformal map $\phi : S \to \mathbb{C}$, such that ϕ satisfies the Beltrami equation:

$$\frac{\partial \phi}{\partial \bar{z}} = \mu \frac{\partial \phi}{\partial z}.$$

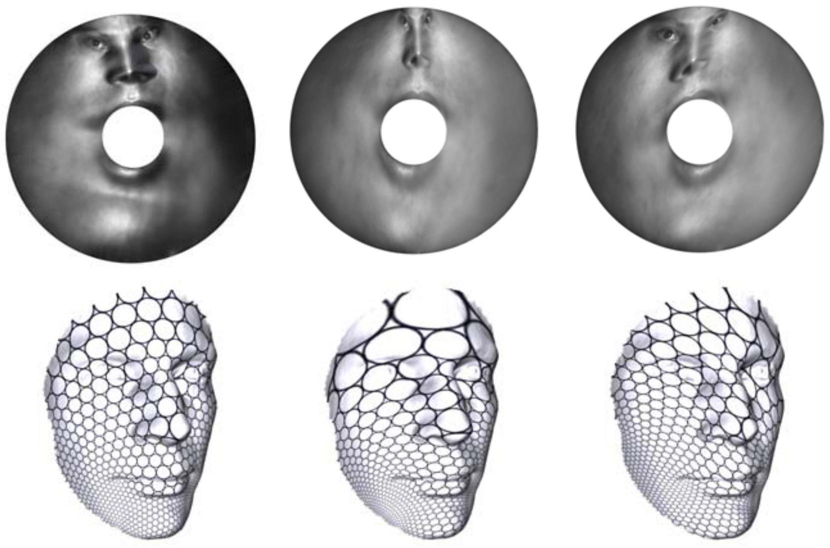


Fig. 22. Quasi-Conformal mapping for doubly connected domain

First we construct a conformal mapping $\phi_1 : S \to \mathbb{D}_1$, where $\mathbb{D}_1$ is a planar domain on $\mathbb{C}$. Let $\mathbf{g}_0$ be the canonical Euclidean metric on the plane $\mathbf{g}_0 = dzd\bar{z}$. ϕ_1 is a conformal map from metric surface $(S, \mathbf{g})$ to a metric surface $(\mathbb{D}_1, \mathbf{g}_0)$. Then we construct a new metric on $(\mathbb{D}_1, \mathbf{g}_0)$, such that

$$\mathbf{g}_1 = |dz + \mu d\bar{z}|^2.$$

Then we construct another conformal map $\phi_2 : (\mathbb{D}_1, \mathbf{g}_1) \to (\mathbb{D}_2, \mathbf{g}_0)$. Then the composition

$$\phi = \phi_2 \circ \phi_1 : (S, \mathbf{g}) \to (\mathbb{D}_2, \mathbf{g}_0)$$

is the desired quasi-conformal mapping. Figure 22 illustrates quasi-conformal mappings for a doubly connected domain with different Beltrami coefficients. For details, we refer readers to [48].

5 Applications

The conformal geometric methods have been broadly applied in many engineering fields. In the following, we briefly introduce their major applications in graphics, vision, geometric modeling and medical imaging.

5.1 Computing Conformal Mappings

Doubly Connected Domain. As shown in Figure 23, a doubly connected domain is conformally mapped to a canonical planar annulus. The mapping can be constructed using holomorphic 1-form. The holomorphic 1-form group on the surface

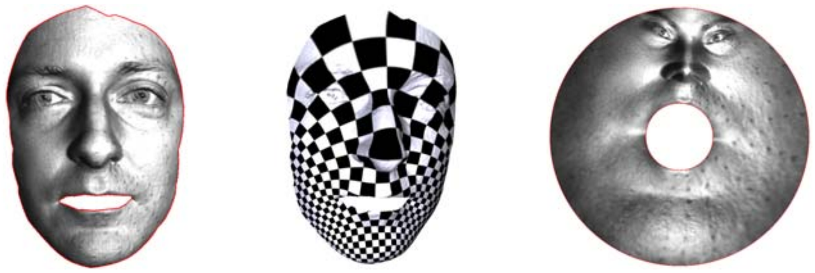


Fig. 23. Conformal mapping for a doubly connected domain

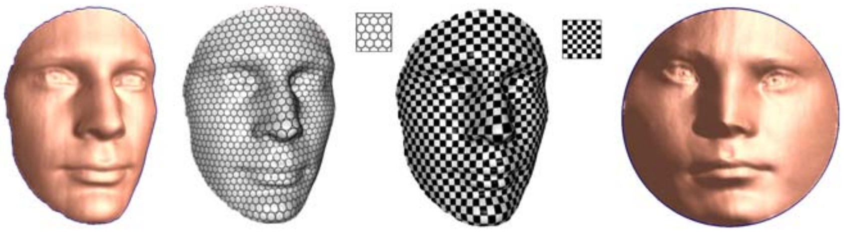


Fig. 24. Conformal mapping for a simply connected domain

is of one dimension. Let ω be the generator. Suppose $\partial S = \gamma_1 - \gamma_2$, then we can find a constant c, such that

$$Im(\int_{\gamma_1} c\omega) = 2\pi i, \qquad Im(\int_{\gamma_2} c\omega) = -2\pi i.$$

Then we choose one point $p \in S$: for any point $q \in S$, find a path $\gamma \subset S$, and define $\phi(q)$ so that

$$\phi(q) : q \to exp\int_{\gamma} c\omega,$$

maps the surface to the planar annulus.

The conformal mapping can be constructed using either Ricci flow or Yamabe flow. We set the target curvature equal to zero for both interior vertices and boundary vertices. Then by using curvature flow, we get a flat metric $\bar{\mathbf{g}}$. We isometrically embed the universal covering space of S with $\bar{\mathbf{g}}$ onto the plane, then make each fundamental polygon a rectangle, with the period equal to 2π, and then by using the exponential map, we can map each fundamental polygon to a planar annulus.

Simply Connected Domains. The Riemann mapping for a topological disk to the canonical planar unit disk can be constructed in the following way (as shown in Figure 24). We choose a interior point p, and a point q on the boundary. Then remove a small neighborhood around p, then we get an annulus $\tilde{S}$. Using holomorphic 1-form or curvature flow method, we map $\tilde{S}$ to a planar unit annulus,

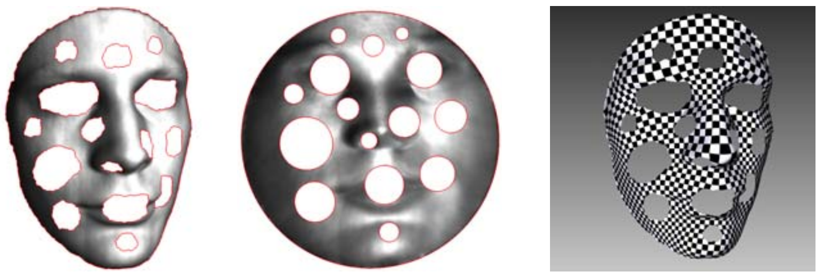


Fig. 25. Conformal mapping for a multiply connected domain

the two boundary circles are concentric. Then by a rotation, we can map q to the point 1. We denote the mapping as $\tilde{\phi} : \tilde{S} \to A$. By shrinking the neighborhood of p, $\tilde{\phi}$ converges to the unique Riemann mapping $\phi : S \to \mathbb{D}$, such that $\phi(p) = 0$, $\phi(q) = 1$.

Multiply Connected Domains. A multiply connected domain as shown in Figure 25 can be conformally mapped to the unit disk with circular holes. Let the boundary of the surface be

$$\partial S = \gamma_0 - \gamma_1 - \gamma_2 - \cdots - \gamma_n,$$

where γ_0 is the exterior boundary. The computation procedure is as follows: choose $\gamma_k, 1 < k < n$, fill the γ_i's with disks, where $i \neq 0, i \neq k$, denote the resulting surface as S_k. Then we can find a conformal mapping $\phi_k : S_k \to \mathbb{D}_k$ and remove all filled disks from D_k. Choosing a different inner boundary, repeat the above procedure. The inner holes become rounder and rounder. With an appropriate normalization condition, (the conformal mapping is not unique, different mappings differ by a Möbius transformation), the mappings converge with very fast rate.

Comparison. Curvature flow method is non-linear and requires higher quality of triangulation, therefore it is less robust and less efficient comparing to the holomorphic 1-form method.

Quasi-Conformal Maps. All the above algorithms can be directly generalized to compute quasi-conformal mappings by solving Beltrami equations.

5.2 Geometric Modeling

One of the most fundamental problems in geometric modeling is to systematically generalize conventional Spline schemes from Euclidean domains to manifold domains. Conventional spline schemes are constructed based on Affine invariance. If the manifold has an affine structure, then affine geometry can be defined on the manifold, therefore, conventional splines can be directly defined on the manifold. Due to the topological obstruction, general manifolds don't have affine

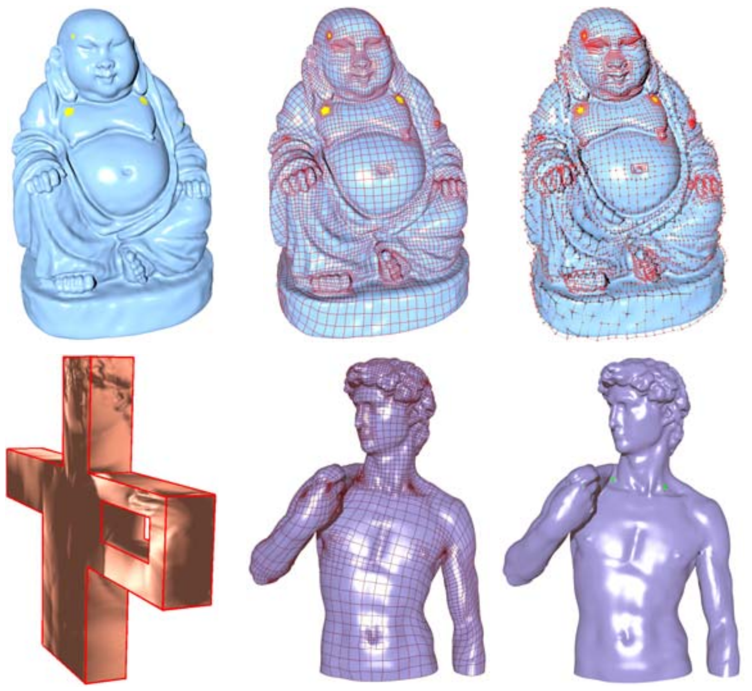


Fig. 26. Manifold splines. The yellow points are extraordinary points.

structures, but by removing several singularities, general surfaces can admit affine structures.

Affine structures can be explicitly constructed using the methods introduced above. For example, we can concentrate all the curvatures at the prescribed singularity positions, and set the target curvatures to be zeros everywhere else. Then we use curvature flow to compute a flat metric with cone singularities from the prescribed curvature. The flat metric induces an atlas on the punctured surface (with singularities removed), such that all the transition functions are rigid motions on the plane. Another approach is to use holomorphic 1-forms, a holomorphic 1-form induces a flat metric with cone singularities at the zeros, where the curvatures are $-2k\pi$. Figure 26 shows the manifold splines constructed using the curvature flow method.

5.3 Medical Imaging

Conformal geometry has been applied for manifold fields in medical imaging. For example, in brain imaging field, it is crucial to register different brain cortex surfaces. Because brain surfaces are highly convoluted, and different persons have different anatomical structures, it is quite challenging to find good matching between cortex surfaces. Figure 27 illustrates one solution by mapping brains

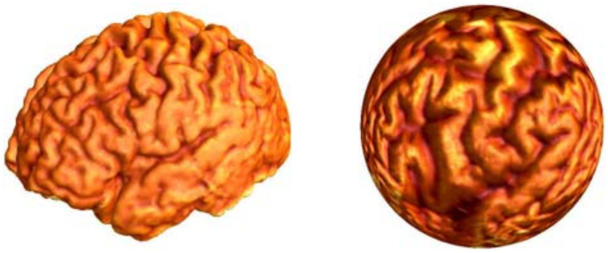


Fig. 27. Brain spherical conformal mapping

to the unit sphere in a canonical way. Then by finding an automorphism, the registration between surfaces can be easily established.

In virtual colonoscopy, the colon surface is reconstructed from CT images. By using the conformal geometric method, one can flatten the whole colon surface onto a planar rectangle. Then polyps and other abnormalities can be found efficiently on the planar image. Figure 28 shows an example for virtual colon flattening based on conformal mapping.

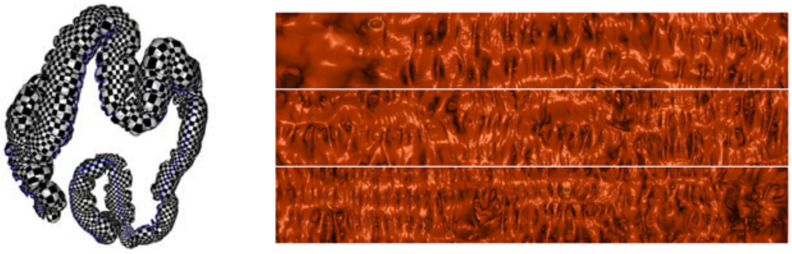


Fig. 28. Colon conformal flattening

5.4 Vision

Surface matching is a fundamental problem in computer vision. The main framework of surface matching can be formulated in the commutative diagram in Figure 29.

S_1, S_2 are two given surfaces and $f : S_1 \to S_2$ is the desired matching. We compute $\phi_i : S_i \to D_i$ which maps S_i conformally onto the canonical domain D_i. D_1 and D_2 can also be surfaces other than simple planar domains. The topology and the curvature of D_1 and D_2 incorporate the major feature information of the original surfaces S_1 and S_2. We construct a diffeomorphism map $\bar{f} : D_1 \to D_2$. If there are certain feature constraints, they can be incorporated in $\bar{f}$. The final map ϕ is induced by $f = \phi_2 \circ \bar{f} \circ \phi_1^{-1}$.

Figure 30 shows one example of surface matching among different expressions on a human face. The first row shows the surfaces in $\mathbb{R}^3$. The second row illustrates the matching results using consistent texture mapping. The intermediate

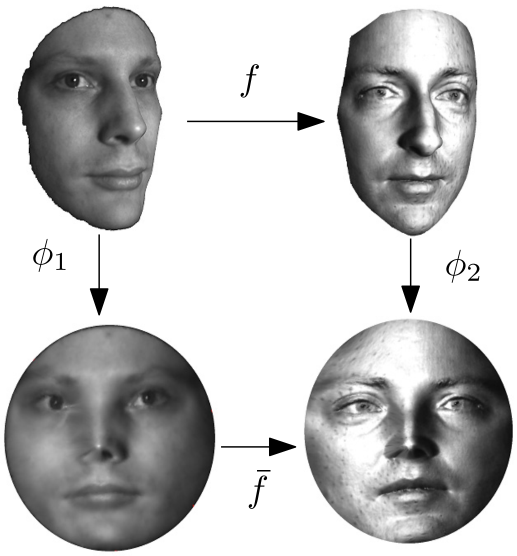


Fig. 29. Surface matching framework

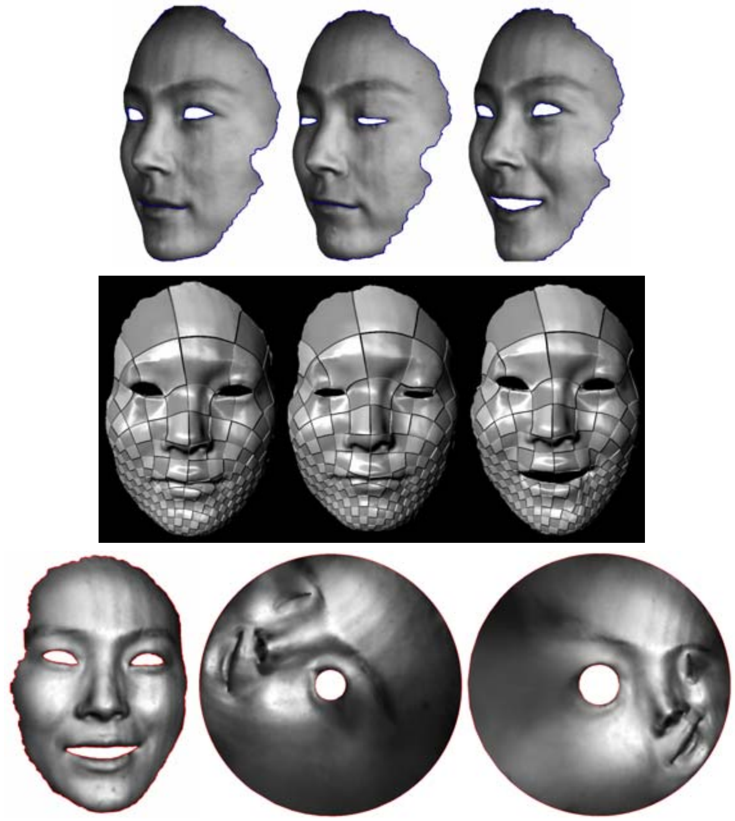


Fig. 30. Matching among faces with different expressions

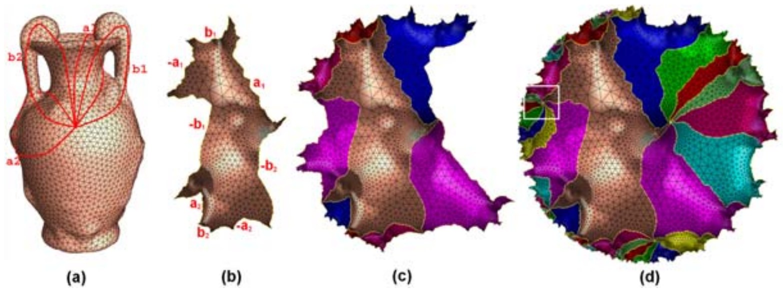


Fig. 31. Computing finite portion of the universal covering space on the hyperbolic space

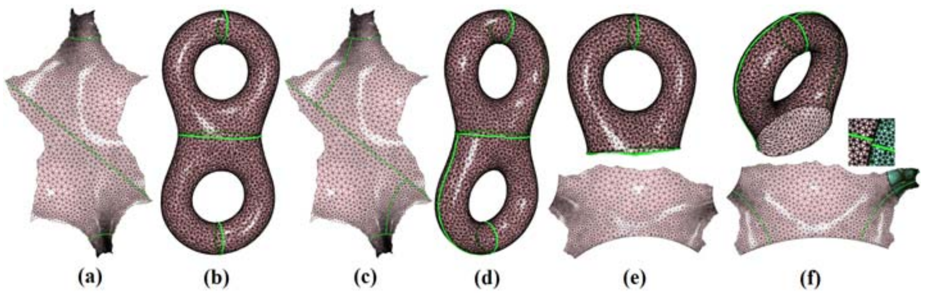


Fig. 32. Computing the Fenchel-Nielsen coordinates in the Teichmüller space for a genus two surface

conformal slit mappings are shown in the third row. For details, we refer readers to [22]. Conformal geometric invariants can also be applied for shape analysis and recognition, details can be found in [49].

Teichmüller theory can be applied for surface classification. By using Ricci curvature flow, we can compute the hyperbolic uniformization metric. Then we compute the pants decomposition using geodesics and compute the Fenchel-Nielsen coordinates. In Figure 31, a set of canonical fundamental group basis is computed (a). Then a fundamental domain is isometrically mapped to the Poincaré disk with the uniformization metric (b). By using Fuchsian transformation, the fundamental domain is transferred (c) and a finite portion of the universal covering space is constructed in (d). Figure 32 shows the pipeline for computing the Teichmüller coordinates. The geodesics on the hyperbolic disk is found in (a) and the surface is decomposed by geodesics (b). The shortest geodesics between two boundaries of each pair of hyperbolic pants are computed in (c),(d) and (e). The twisting angle is computed in (f). Details can be found in [50].

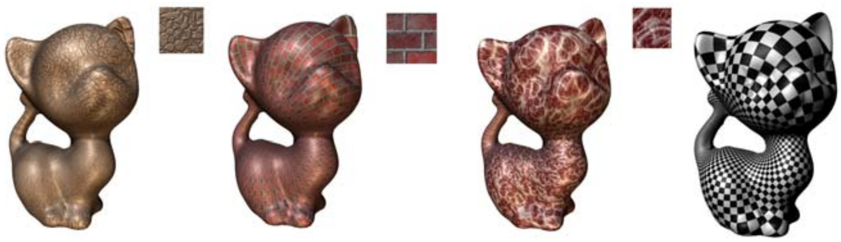


Fig. 33. Global conformal surface parameterization using holomorphic 1-form

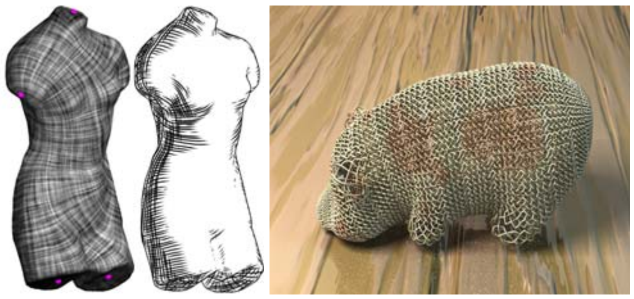


Fig. 34. Vector field design using special flat metrics

5.5 Graphics

Conformal geometric methods have broad applications in computer graphics. Isothermal coordinates are natural for global surface parameterization purpose. Because conformal mapping doesn't distort the local shapes, it is desirable for texture mapping. Figure 33 shows one example of using holomorphic 1-form for texture mapping.

Special flat metrics are valuable for designing vector fields on surfaces, which plays an important role in non-photo-realistic rendering and special art form design. Figure 34 shows the examples for vector fields design on surfaces using curvature flow method.

6 Conclusion

In this work, we briefly summarize the recent developments in computational conformal geometry. There are two major approaches, holomorphic differentials and curvature flow. Holomorphic differential method is a linear method, which is more efficient and robust to triangulations with lower quality. Curvature flow method is nonlinear and requires higher quality triangulations, but more flexible.

There are many future directions in both theories and algorithms:

1. The theoretic foundations for the current computational methods need to be rigorously laid down. The convergence of the discrete conformal structure to the smooth counterpart needs to be proven. The convergence rate needs to be estimated.
2. More research focuses on quasi-conformal mappings. Especially the computational method for extremal quasi-conformal mappings and Teichmüller theories.
3. Generalization of the computational methods for discrete 3-manifolds. Especially the curvature flow method for canonical geometric structures of 3-manifolds.
4. Novel scanning technology which can measure and capture the Beltrami coefficients from real life.

Acknowledgements

We want to thank our collaborators Paul Thompson, Tony Chan, Arie Kaufman, Hong Qin, Dimitris Samaras and many other colleagues. We want to thank our students. The research has been supported by NSF.

References

1. Henrici, P.: Applied and Computational Complex Analysis, Discrete Fourier Analysis, Cauchy Integrals, Construction of Conformal Maps, Univalent Functions, vol. 3. Wiley Interscience, Hoboken (1993)
2. Henrici, P.: Applied and Computational Complex Analysis, Power Series Integration Conformal Mapping Location of Zero, vol. 1. Wiley Interscience, Hoboken (1988)
3. Floater, M.S., Hormann, K.: Surface parameterization: a tutorial and survey. In: Advances in Multiresolution for Geometric Modelling, pp. 157–186. Springer, Heidelberg (2005)
4. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3D models. ACM Transactions on Graphics 23(3), 861–869 (2004)
5. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 2(1), 15–36 (1993)
6. Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: SIGGRAPH 2002, pp. 362–371 (2002)
7. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic parameterizations of surface meshes. Computer Graphics Forum (Proc. Eurographics 2002) 21(3), 209–218 (2002)
8. Floater, M.S.: Mean value coordinates. Computer Aided Geometric Design 20(1), 19–27 (2003)
9. Gotsman, C., Gu, X., Sheffer, A.: Fundamentals of spherical parameterization for 3D meshes. ACM Transactions on Graphics 22(3), 358–363 (2003)
10. Gu, X., Wang, Y., Chan, T.F., Thompson, P.M., Yau, S.T.: Genus zero surface conformal mapping and its application to brain surface mapping. IEEE Trans. Med. Imaging 23(8), 949–958 (2004)

11. Gu, X., Yau, S.T.: Global conformal parameterization. In: Symposium on Geometry Processing, pp. 127–137 (2003)
12. Mercat, C.: Discrete Riemann surfaces and the Ising model. Communications in Mathematical Physics 218(1), 177–216 (2004)
13. Hirani, A.N.: Discrete exterior calculus. PhD thesis, California Institute of Technology (2003)
14. Jin, M., Wang, Y., Yau, S.T., Gu, X.: Optimal global conformal surface parameterization. In: IEEE Visualization 2004, pp. 267–274 (2004)
15. Gortler, S.J., Gotsman, C., Thurston, D.: Discrete one-forms on meshes and applications to 3D mesh parameterization. Computer Aided Geometric Design 23(2), 83–112 (2005)
16. Tong, Y., Alliez, P., Cohen-Steiner, D., Desbrun, M.: Designing quadrangulations with discrete harmonic forms. In: Symposium on Geometry Processing, pp. 201–210 (2006)
17. Tewari, G., Gotsman, C., Gortler, S.J.: Meshing genus-1 point clouds using discrete one-forms. Comput. Graph. 30(6), 917–926 (2006)
18. Desbrun, M.: Discrete differential forms and applications to surface tiling. In: SCG 2006: Proceedings of the twenty-second annual symposium on Computational geometry, p. 40. ACM, New York (2006)
19. Tong, Y., Alliez, P., Cohen-Steiner, D., Desbrun, M.: Designing quadrangulations with discrete harmonic forms. In: SGP 2006: Proceedings of the fourth Eurographics symposium on Geometry processing, pp. 201–210 (2006)
20. Hong, W., Gu, X., Qiu, F., Jin, M., Kaufman, A.E.: Conformal virtual colon flattening. In: Symposium on Solid and Physical Modeling, pp. 85–93 (2006)
21. Wang, S., Wang, Y., Jin, M., Gu, X.D., Samaras, D.: Conformal geometry and its applications on 3d shape matching, recognition, and stitching. IEEE Trans. Pattern Anal. Mach. Intell. 29(7), 1209–1220 (2007)
22. Zeng, W., Zeng, Y., Wang, Y., Yin, X., Gu, X., Samaras, D.: 3d non-rigid surface matching and registration based on holomorphic differentials. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 1–14. Springer, Heidelberg (2008)
23. Gu, X., He, Y., Qin, H.: Manifold splines. Graphical Models 68(3), 237–254 (2006)
24. Hamilton, R.S.: Three manifolds with positive Ricci curvature. Journal of Differential Geometry 17, 255–306 (1982)
25. Hamilton, R.S.: The Ricci flow on surfaces. Mathematics and general relativity (Santa Cruz, CA, 1986), Contemp. Math. Amer. Math. Soc. Providence, RI 71 (1988)
26. Thurston, W.P.: Geometry and Topology of Three-Manifolds. Lecture notes at Princeton university (1980)
27. Koebe, P.: Kontaktprobleme der Konformen Abbildung. Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Kl. 88, 141–164 (1936)
28. Thurston, W.P.: The finite Riemann mapping theorem (1985)
29. Rodin, B., Sullivan, D.: The convergence of circle packings to the Riemann mapping. Journal of Differential Geometry 26(2), 349–360 (1987)
30. Colin de Verdière, Y.: Un principe variationnel pour les empilements de cercles. Invent. Math. 104(3), 655–669 (1991)
31. Collins, C., Stephenson, K.: A circle packing algorithm. Computational Geometry: Theory and Applications 25, 233–256 (2003)
32. Chow, B., Luo, F.: Combinatorial Ricci flows on surfaces. Journal Differential Geometry 63(1), 97–129 (2003)

33. Jin, M., Kim, J., Luo, F., Gu, X.: Discrete surface Ricci flow. IEEE Transaction on Visualization and Computer Graphics (2008)
34. Bowers, P.L., Hurdal, M.K.: Planar conformal mapping of piecewise flat surfaces. In: Visualization and Mathematics III, pp. 3–34. Springer, Berlin (2003)
35. Bobenko, A.I., Springborn, B.A.: Variational principles for circle patterns and Koebe's theorem. Transactions of the American Mathematical Society 356, 659–689 (2004)
36. Kharevych, L., Springborn, B., Schröder, P.: Discrete conformal mappings via circle patterns. ACM Trans. Graph. 25(2), 412–438 (2006)
37. Yamabe, H.: The Yamabe problem. Osaka Math. J. 12(1), 21–37 (1960)
38. Trudinger, N.S.: Remarks concerning the conformal deformation of Riemannian structures on compact manifolds. Ann. Scuola Norm. Sup. Pisa 22(2), 265–274 (1968)
39. Aubin, T.: Équations différentielles non-linéaires et problème de Yamabe concernant la courbure scalaire. J. Math. Pures Appl. 55(3), 269–296 (1976)
40. Schoen, R.: Conformal deformation of a Riemannian metric to constant scalar curvature. J. Differential Geom. 20(2), 479–495 (1984)
41. Lee, J.M., Parker, T.H.: The Yamabe problem. Bulletin of the American Mathematical Society 17(1), 37–91 (1987)
42. Luo, F.: Combinatorial Yamabe flow on surfaces. Commun. Contemp. Math. 6(5), 765–780 (2004)
43. Springborn, B., Schröoder, P., Pinkall, U.: Conformal equivalence of triangle meshes. ACM Transactions on Graphics 27(3), 1–11 (2008)
44. Computational conformal geometry library 1.1, http://www.cs.sunysb.edu/~manifold/CCGL1.1/
45. Zeng, W., Jin, M., Luo, F., Gu, X.: Computing canonical homotopy class representative using hyperbolic structure. In: IEEE International Conference on Shape Modeling and Applications, SMI 2009 (2009)
46. Thurston, W.P.: Geometry and Topology of Three-Manifolds. Princeton lecture notes (1976)
47. Luo, F., Gu, X., Dai, J.: Variational Principles for Discrete Surfaces. Advanced Lectures in Mathematics. High Education Press and International Press (2007)
48. Zeng, W., Luo, F., Yau, S.T., Gu, X.D.: Surface quasi-conformal mapping by solving Beltrami equations. In: IMA: 13th International Conference on The Mathematics of Surfaces (2009)
49. Zeng, W., Lui, L.M., Gu, X., Yau, S.T.: Shape analysis by conformal modules. In: Methods and Applications of Analysis (2009)
50. Jin, M., Zeng, W., Ning, D., Gu, X.: Computing Fenchel-Nielsen coordinates in teichmuller shape space. In: IEEE International Conference on Shape Modeling and Applications, SMI 2009 (2009)

Finite Curvature Continuous Polar Patchworks

Kęstutis Karčiauskas[1] and Jörg Peters[2]

[1] Vilnius University, Lithuania
[2] University of Florida, USA

Abstract. We present an algorithm for completing a C^2 surface of up to degree bi-6 by capping an n-sided hole with polar layout. The cap consists of n tensor-product patches, each of degree 6 in the periodic and degree 5 in the radial direction. To match the polar layout, one edge of these patches is collapsed.

We explore and compare with alternative constructions, based on more pieces or using total-degree, triangular patches.

1 Introduction

Vertices of high valence occur in control nets of surfaces of revolution and similar objects and as higher-order saddle points; see Figure 1. Representing the neighborhood of such a high-valent vertex in *polar layout*, i.e. surrounding the point by one layer of triangles while the next layer of facets consists of quadrilaterals with always four joining at a vertex, is a natural and compatible addition to the established tensor-product layout of patches (see e.g. [1]). Polar layout can be used to define subdivision surfaces (see Section 1.1), but engineers may well prefer finite constructions, in the sense that the final output surface consists of a finite number of polynomial or rational patches. Here, we therefore present *finite polar curvature continuous patchworks* that fill an n-sided hole in an existing tensor-product spline complex.

Away from the central *polar vertex*, the net as in Figure 1 can be interpreted as the control net of a C^2 spline; for example a bi-3 spline. However, assuming that the tensor-border is obtained from a piecewise bi-3 layer of patches generated from a polar mesh is in general too restrictive for high-quality modeling. More

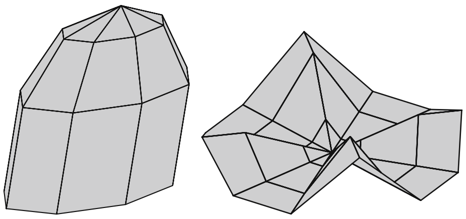


Fig. 1. Polar layout: (*left*) elliptic 8-sided dome; (*right*) 12-sided higher-order saddle

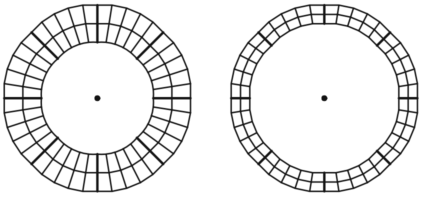


Fig. 2. Tensor-border input: (*left*) A degree 3 polar tensor-border derived from a polar control net, raised to degree 5; the central point is set to the limit point of bi-3 polar subdivision [2]. (*right*) A degree 5 tensor-border not generated from a polar control net; the derivatives may be scaled before applying Algorithm I. The central point is user-set.

generally, we consider the following input: position, first and second derivative along a closed curve of degree up to 6. We call such input a *tensor-border* of degree p if curve and derivatives are of degree at most p. Figure 2 shows two tensor-borders represented by coefficients in tensor-product BB-form (Bernstein-Bézier form; see e.g. [3]).

In the following, we discuss and compare several possible approaches to constructing a cap that matches a given tensor-border. Section 2 contains the main construction: Algorithm I generates one patch for each of the n sectors of the polar layout. Each patch is of *degree 6-5*, degree 6 in the periodic direction (circulating around the pole) and degree 5 in the radial direction (emanating from the pole); one periodic edge is collapsed to form the pole. Most CAD packages do not mind tensor-product patches with one collapsed edge. However, it raises the question, whether we can achieve the same good quality with similar polynomial degree and without collapse, using, say, 'triangular' patches. Section 3.1 shows that this is indeed possible (Algorithm II) and also presents a non-collapsed construction using multiple tensor-product pieces (Algorithm III). Section 3.2 shows that we can alternatively reduce the degree to bi-5 using a collapsed-edge representation. The resulting curvature continuous patchworks show good curvature distribution but the construction comes at the cost of (considerably) more patches per sector (Algorithm IV). The discussion section points out that non-uniformly spaced parameterizations are easy to realize within the same framework.

1.1 Related Literature

A simple mesh-based algorithm generating C^1 limit surfaces of degree bi-3 with polar layout was introduced in [2]. The bi-3 construction can be joined with Catmull-Clark subdivision [4] in a natural fashion [1]. *Guided* C^2 subdivision, both for the polar and for the all-quadrilateral layout of Catmull-Clark appeared in [5]. [1] showed a construction with a fixed, finite number of patches that is functionally equivalent to the polar subdivision scheme defined in [2]. Similarly, a number of constructions have recently been developed to improve on

Catmull-Clark surfaces in a finite setting [6,7,8,9,10]. Finite guided C^2 surface constructions via G^2 patchworks with the layout of a Catmull-Clark input mesh, using patches of degree bi-6 or even only bi-5 have been announced. Degenerate (triangular) Bézier patches joining with continuous curvature were analyzed in [11].

2 C^2 Polar Cap Construction Using One Patch of Degree 6-5 per Sector

Figure 3 illustrates the structure of each patch of degree 6-5 and Figure 4 shows a constructed surface. The construction has four ingredients explained in detail in Section 2.1:

1. a guide surface cap $\mathbf{q} : \Omega \subsetneq \mathbb{R}^2 \mapsto \mathbb{R}^3$ consisting of n triangular patches of total degree 2 (gray region in Figure 5 *middle*) ;

2. a polar concentric tessellation map (*ct-map*), $\rho : [0..1]^2 \times \{1, \ldots, n\} \to \mathbb{R}^2$ (Figure 5 *left*) ;

3. the operator $H_{n,p}$ that defines a surface ring or cap of n patches of periodic degree p and radial degree 5 by (Hermite-)interpolating position, first and second derivative at its two periodic edges ;

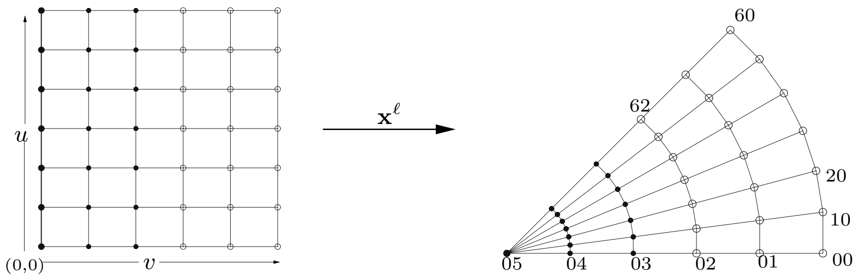


Fig. 3. A **6-5 patch** $\mathbf{x}^\ell : [0..1]^2 \to \mathbb{R}^3$ of degree 6 in the periodic and degree 5 in the radial direction. (*left*) Domain, (*right*) patch with one edge collapsed: for $i = 0, \ldots, 6$, $\mathbf{x}_{i5}$ is the pole. The BB-coefficients displayed as circles are defined by the tensor-border. The BB-coefficients displayed as black disks are defined by a quadratic expansion at the central point.

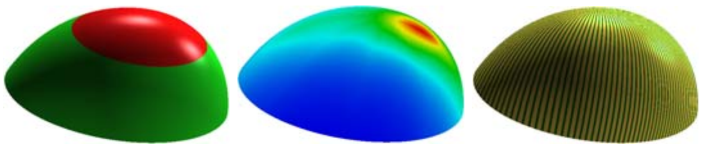


Fig. 4. A **6-5 convex completion** of a surface. (*left*) Bi-3 input ring (green, defining the tensor-border) and 6-5 cap (red); (*middle*) Gaussian curvature shading; (*right*) highlight shading.

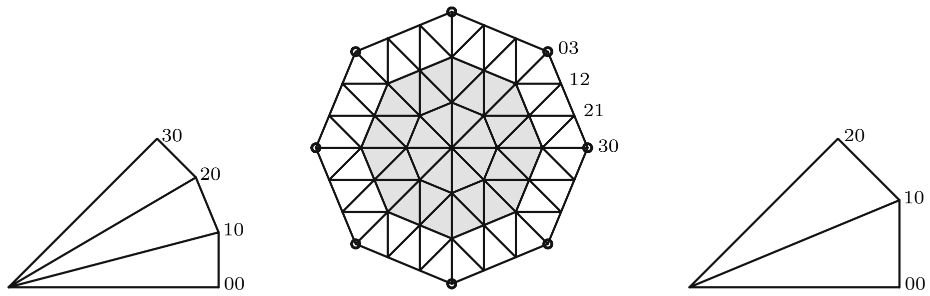


Fig. 5. Polar reparameterizations (ct-maps) and guide surfaces. One sector (*left*) of the C^2 bi-31 ct-map ρ, (*right*) of the G^2 bi-21 ct-map $\rho_{\mathbf{2}}$. (*middle*) BB control net $\mathbf{p}^{\ell}_{jk}$ of a guide patch $\mathbf{q}$ of degree 2 (gray area, 2-link) or $\mathbf{c}$ of degree 3. (Note the different indexing conventions for (collapsed) tensor-product and total degree patches.)

4. the functional

$$\mathcal{F} : C^m([0..1]^2) \ni f \to \mathcal{F}_m(f) := \int_{\square} \sum_{i+j=m; i,j\geq 0} \binom{m}{ij} (\partial^m_{u^i v^j} f)^2 \tag{1}$$

that acts on sufficiently smooth functions f defined over the unit square $[0..1]^2$.

2.1 The Ingredients

1. The piecewise quadratic guide $\mathbf{q}$ consisting of n triangular, total-degree 2 polynomial pieces forms a C^2 map if all pieces define the same quadratic. In other words, its Bernstein-Bézier (BB)-coefficients $\mathbf{p}^{\ell}_{00}$, $\mathbf{p}^{\ell}_{10}$, $\mathbf{p}^{\ell}_{01}$, $\mathbf{p}^{\ell}_{20}$, $\mathbf{p}^{\ell}_{11}$, $\mathbf{p}^{\ell}_{02}$ (gray region in Figure 5 *middle*) are defined by one piece, say $\ell = 0$.

2. The polar concentric tessellation map (*ct-map*) (Figure 5 *left*),

$$\rho : [0..1]^2 \times \{1, \ldots, n\} \to \mathbb{R}^2$$

that reparameterizes the domain is defined as follows. Its ℓth sector is defined by a template map $r : [0..1]^2 \to \mathbb{R}^2$ rotated about the origin by $\ell\frac{2\pi}{n}$. The template map is of degree bi-31, i.e. of degree 3 in the periodic u-direction and degree 1 in the radial v-direction (see Figure 5 *left*). Its BB-coefficients are $r_{i1} := \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$, $i = 0, 1, 2, 3$,

$$r_{00} := \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], r_{10} := \left[\begin{smallmatrix}1\\ \frac{\sin\alpha}{2+\cos\alpha}\end{smallmatrix}\right],\ r_{20} := R_\alpha r_{10},\ r_{30} := R_\alpha r_{00} = \left[\begin{smallmatrix}\cos\alpha\\ \sin\alpha\end{smallmatrix}\right],\ \alpha := \frac{2\pi}{n},$$

where R_α is the reflection across the line through the origin and $[\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}]$. By construction, the polynomial pieces of ρ join formally C^2 across the sector boundaries.

3. The operator $H_{n,p}(\mathbf{b}_0, \mathbf{b}_1)$ defines a C^2 surface ring or cap $\mathbf{x} \in C^2([0..1]^2 \times \{1, \ldots, n\})$ consisting of n polynomial patches of periodic degree p and radial degree 5 uniquely determined by (Hermite-)interpolating position, first and second derivative (the tensor-borders $\mathbf{b}_k$ of degree p) at its two periodic edges corresponding to $v = 0$ and $v = 1$, respectively. We denote by $J(\mathbf{f}, v)$, a tensor-border read off from a patch $\mathbf{f} \in C^2([0..1]^2 \times \{1, \ldots, n\})$ at its radial parameter v.

Below we will build a patch $\mathbf{x}$ of degree 6-5 as shown in Figure 3. Three layers of BB-coefficients marked as black disks are defined by $\mathbf{b}_0$ and three layers marked by circles are defined by $\mathbf{b}_1$.

Algorithm I: C^2 cap construction of degree 6-5
Input. The tensor-border $\mathbf{b}$ of the hole in a C^2 patch complex. The position of the pole (see Figure 2).
Output. The n pieces $\mathbf{x}^\ell$ of the output surface cap $\mathbf{x} : [0..1]^2 \times \{1, \ldots, n\} \to \mathbb{R}^3$ (see Figure 3, *right*) of degree 6-5, i.e. of degree 6 in the periodic direction with one periodic edge collapsed to form the pole and degree 5 in the radial direction.

Algorithm: Set the pole, for example to the limit of bi-3 polar subdivision [2] (cf. Figure 2). Form $\mathbf{q} \circ \rho$ (with five BB-coefficients $\mathbf{q}_k$, $k = 1, \ldots, 5$ still undetermined) and solve the linear 5×5 system

$$\min_{\mathbf{q}_k} \mathcal{F}_3 H_{n,6}\big(J(\mathbf{q} \circ \rho, 0); \mathbf{b}\big), \tag{2}$$

i.e. minimize the sum of the functionals applied to each of the n patches in terms of the unknown five BB-coefficients.

Theorem 1. *The 6-5 construction yields a C^2 surface.*

Proof. Since $\mathbf{b}$ extends the C^2 patch complex and $\mathbf{q} \circ \rho$ is parametrically C^2 except possibly where it is singular, we need only show that the surface is also C^2 at the central point. For this we construct an auxiliary subdivision algorithm of type (1/2,1/4,0) [12, Ch 7] as follows. Denote the univariate degree 3 boundary of the template of ρ by $r_\circ \in \mathbb{R}^2$ and the $\frac{2\pi}{n}$ rotation matrix by R. With $v = 0$ corresponding to the collapsed edge, and recalling that the Taylor expansion of any 6-5 patch $\mathbf{x}^\ell$ generated by Algorithm I at $(u, 0)$ is up to second order determined by $\mathbf{q} \circ \rho^\ell = \mathbf{q}(vR^\ell r_\circ(u))$, we can write

$$\mathbf{x}^\ell(u, v) := \mathbf{q}_0 + v[\mathbf{q}_1, \mathbf{q}_2](R^\ell r_\circ(u)) + v^2 (R^\ell r_\circ(u))^T \begin{bmatrix} \mathbf{q}_3 & \mathbf{q}_4 \\ \mathbf{q}_4 & \mathbf{q}_5 \end{bmatrix} (R^\ell r_\circ(u)) + o(v^2).$$

Repeated subdivision (by De Casteljau's algorithm) of this representation in the radial direction, at $v = 1/2^m$ yields a sequence of C^2 polar 6-5 rings converging to the pole with $v-$monomial eigenfunctions and eigencoefficients $\mathbf{q}_k$. In particular, for $j = 1$ there are two (rather than n) distinct C^2 eigenfunctions e_{10} and e_{11} and for $j = 2$ there are three, e_{20}, e_{21} and e_{22} such that $e_{20}, e_{21}, e_{22} \in \text{span}\{e_{10}^2, e_{10}e_{11}, e_{11}^2\}$. [12, Thm 7.16] then applies and completes the proof. |||

A similar proof, for C^1 continuity, appeared in [1].

Theorem 2 (periodic degree estimation). *For generic tensor-border data, a parametrically C^2 polar cap assembled from n patches by a symmetric (invariant under index shift and flip) algorithm must have periodic degree at least 6.*

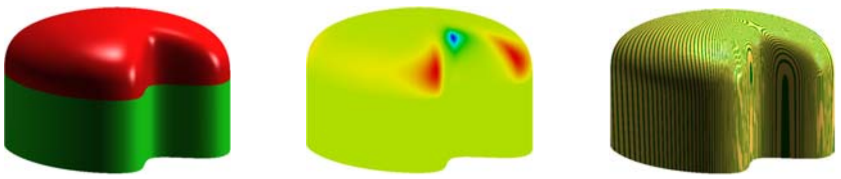


Fig. 6. A **6-5** construction for input not derived from a polar control net. (*left*) Input and cap; (*middle*) Gaussian shading; (*right*) highlights.

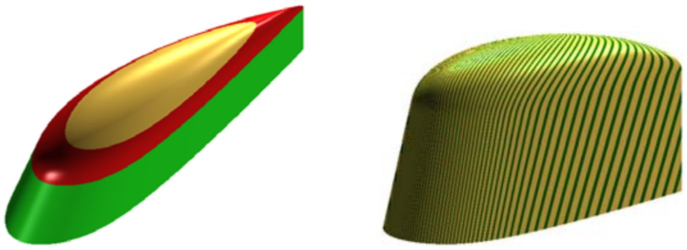


Fig. 7. **Wing tips** feathering out a sharp edge. (*left*) Input data (green) with a sharp edge, transition ring (red) mediating from the sharp edge to the smooth cap and C^2 6-5 cap (gold); (*right*) highlight rendering thereof.

Proof. By [12, Thm 7.16], a (polar) subdivision algorithm of type (1/2,1/4,0) must be of periodic degree at least twice the periodic degree of its C^2 (polar) characteristic ring. Since the periodic degree is at least three the algorithm must generically generate surfaces of periodic degree 6. The proof of Theorem 1 shows that any symmetric parametrically C^2 polar cap of periodic degree m induces a polar C^2 subdivision algorithm of type (1/2,1/4,0) that is of periodic degree m. Therefore $m \geq 6$ must hold. |||

We conclude with examples where the tensor border has not been derived from a control mesh. In Figure 6, the input tensor-border (green in *left*) is defined by nine C^2-connected generalized cylinders of periodic degree 5. In the airplane wing tip data of Figure 7, the challenge is to slowly blend the sharp edge into a rounded cap. To obtain the transition, Algorithm I is applied with a crease in the tensor-border along a sector partitioning curve. The patches are then subdivided in the radial direction and the resulting transition surface ring is adjusted to have three C^2-connected innermost layers of BB-coefficients. Then Algorithm I is applied a second time with the transition ring now providing the tensor-border **b** (*red* in Figure 7 *left*).

3 Alternative Constructions

Below, we explore constructions without edge collapse (Algorithm II and III) and with edge collapse (Algorithm IV) where we trade lower degree for (many) more patches compared to the surface construction of Algorithm I. While these three alternative constructions yield equally good shape and continuity as Algorithm I, we think that the trade-off will only rarely be justified. However, we present these alternatives to complete the picture but move fast on the details.

3.1 C^2 Capping without Collapsed Edge

In this section, we present two C^2 surface constructions without edge collapse. We leverage the 6-5 cap $\mathbf{x} : [0..1]^2 \times \{1, \dots, n\} \to \mathbb{R}^3$ constructed by Algorithm I to provide data from which to determine a C^2 guide surface $\mathbf{p} : \Omega \subsetneq \mathbb{R}^2 \mapsto \mathbb{R}^3$ (see below and Figure 5 *middle*) of degree 3, that is then composed with a reparametrization $\tau : [0..1]^2 \times \{1, \dots, n\} \to \Omega$. Specifically, we
(i) extract data from $\mathbf{x}$ to be matched,
(ii) extract the same data from $\mathbf{p} \circ \tau$ (in terms of free coefficients of $\mathbf{p}$), and
(iii) set the free coefficients by minimizing the difference of the data in (i) and (ii).
Then the final surface is essentially (ii) with the coefficients set by (iii).

Ingredients. Analogous to Section 2.1, we define

1'. a guide $\mathbf{c}$ consisting of n polynomial pieces of total degree 3. The conditions on its BB-coefficients $\mathbf{p}^\ell_{jk}$ to be part of a piecewise C^2 map [3] require, (a) the 2-link of the central point $\mathbf{p}^\ell_{00}$ define the same quadratic polynomial (in possibly degree-raised BB-form) for all ℓ (see the earlier construction of $\mathbf{q}$), and (b) The coefficients $\mathbf{p}^\ell_{30}$ can be chosen freely; then, for $n \geq 7$ (which will hold since we trisect each sector),

$$\begin{aligned}\mathbf{p}^j_{21} &:= \frac{1}{n}\sum_{\ell=0}^{n-1}\sum_{k=0}^{n-1}\frac{R_\ell \cos((j-\ell)k\alpha)}{2\mathsf{c}+\cos(k\alpha)}, \qquad (3)\\ R_\ell &:= 2\mathsf{c}^2\mathbf{p}^\ell_{30} + \mathsf{c}\mathbf{p}^\ell_{03} + 4\mathsf{c}(1-\mathsf{c})\mathbf{p}^\ell_{20}\\ &\quad - 2(1-\mathsf{c})\mathbf{p}^\ell_{11} + (1-\mathsf{c})\mathbf{p}^\ell_{02} + 2(1-\mathsf{c})^2\mathbf{p}^\ell_{10},\\ \mathsf{c} &:= \cos\alpha, \quad \alpha := 2\pi/n,\end{aligned}$$

and setting subsequently the coefficients $\mathbf{p}^\ell_{12}$ to satisfy C^1 constraints across the sector lines, makes $\mathbf{p}$ a C^2 function.
2'. Analogous to the definition of ρ in the previous section, we define the piecewise quadratic G^2 ct-map $\boldsymbol{\rho_2}$ by rotations of a bi-21 template r with BB-coefficients

$$r_{i1} := \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \quad r_{00} := \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], \quad r_{10} := \left[\begin{smallmatrix}1\\\tan(\alpha/2)\end{smallmatrix}\right], \quad r_{20} := \left[\begin{smallmatrix}\cos\alpha\\\sin\alpha\end{smallmatrix}\right].$$

The bi-21 map $\boldsymbol{\rho_2}$ has a sibling $\boldsymbol{\sigma_2}$ of total degree 2 defined by the template

$$r_{0,0,2} := r_{00}, r_{1,0,1} := r_{10}, r_{2,0,0} := r_{20},\ r_{0,1,1} := \frac{r_{00}}{2}, r_{1,1,0} := \frac{r_{20}}{2},\ r_{0,2,0} := \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right].$$

Analogously, ρ has a sibling σ of total degree 3 defined by the cubic curve of ρ and the origin.
3'. We introduce the operator H^{55} that samples 2×2 jets $\partial^i_u\partial^j_v\mathbf{x}$, $i, j \in \{0, 1, 2\}$ at the corners of the polynomial pieces to construct tensor-borders (of degree 5).

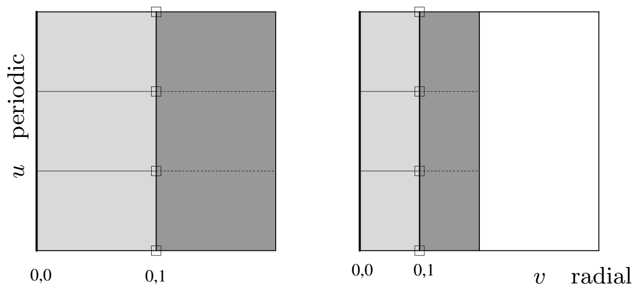


Fig. 8. Radial bisections followed by tri-section at the pole. (*left*) Once, (*right*) twice. In both cases, we associate $v = 0$ with the collapsed edge of $\mathbf{x}$ and $v = 1$ at the boundary between the gray regions of the domain. The light grey region is the focus of Algorithms II, III, IV.

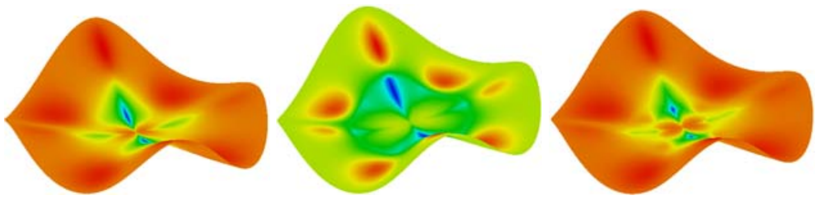


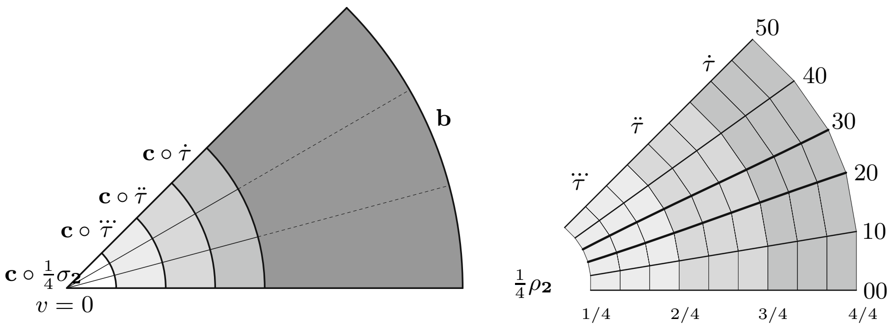
Fig. 9. Single vs double split. (*left*) Gauss curvature shading of a 6-5 cap (input Figure 2 *middle left*); (*middle*) Gauss shading of degree 9 construction corresponding to one bisection; (*right*) Gauss shading of degree 9 construction corresponding to the double split in Figure 8 *right*.

***C^2* capping of total degree 9.** To cap without a collapsed edge, we here choose the central pieces to be 'triangular', i.e. of total degree d (which could in turn be represented by three quad patches of degree bi-d).

Algorithm II: C^2 cap construction of total degree 9
Input: The C^2 cap $\mathbf{x}$ of degree 6-5 constructed by Algorithm I and the tensor-border $\mathbf{b} := J(\mathbf{x}, 1)$ ($\mathbf{b} := J(\mathbf{x}, \frac{1}{2})$ for double bisection; see * below).
Output: A C^2 cap consisting of $N := 3n$ patches of total degree 9; plus N patches of degree bi-95 for one bisection (plus n patches of degree 6-5 for double bisection).
Algorithm. (i) Bisect $\mathbf{x}$ radially once or twice (see Figure 8).

* For elliptic configurations such as in Figure 1 *left*, the single bisection displayed in Figure 8 *left* suffices. For higher-order saddles, we recommend the double split of Figure 8 *right*, resulting in Figure 9 *right*.

Then tri-sect the subpatches at the pole and in the layer next to it so that the new polar valence is N. Apply H^{55} at the locations marked by $\square$ in Figure 8 to generate a tensor-border of degree 5 in B-spline form.



$\mathbf{b}$
$\mathbf{c}\circ\dot\tau$
$\mathbf{c}\circ\ddot\tau$
$\mathbf{c}\circ\dddot\tau$
$\mathbf{c}\circ\frac14\sigma_2$
$v=0$
50
40
30
20
10
00
$\dot\tau$
$\ddot\tau$
$\dddot\tau$
$\frac14\rho_\mathbf{2}$
1/4
2/4
3/4
4/4

Fig. 10. G^2 **capping with a** 5×3**-split**. (*left*) One sector is covered by 15 patches. (*right*) One sector of the domain transition map τ of degree bi-53 (corresponding to the middle three surface rings of one sector after *tri-section* in the circular direction *left*, i.e. valence $N = 3n$; τ is symmetric with respect to the bisectrix of the sector. The radial curves starting inward at 00 and 10 and symmetrically at 40, 50 are straight lines. Only the two heavy-set lines starting at 20 and 30 are truly of piecewise degree 3.

(ii) Compose $\mathbf{c}\circ\rho$ to yield a map of degree bi-93 with $6+N$ undetermined coefficients of $\mathbf{c}$. Apply H^{55} at the same locations as in (i) to form a second tensor-border of degree 5 in B-spline form.
(iii) Set the central point of $\mathbf{c}$ to the central point of the 6-5 cap $\mathbf{x}$. Determine the remaining $N+5$ coefficients of $\mathbf{c}$ by minizing the sum of squared distances between the B-spline control points generated in (i) and (ii).
(iv) The inner cap corresponding to $v\in[0..1]$ is defined by $\mathbf{c}\circ\sigma$, in essence trimming $\mathbf{c}$ along the degree 3 boundary of ρ. Since we trisected, this yields three patches of total degree 9 per sector.
(v) For radial parameter $v\in[1..2]$ the surrounding surface ring consists of N patches of degree bi-95 defined by

$$H_{N,9}(J(\mathbf{c}\circ\rho,1);\mathbf{b}),$$

i.e. the tensor-border of $\mathbf{c}\circ\rho$ at $v=1$ (consisting of $N=3n$ pieces) and the tri-sected and degree-raised tensor-border.

***G*²** **capping of total degree 6.** Here we use a map $\tau:[0..1]^2\to\mathbb{R}^2$ (Figure 10 *right*) to transition from C^2 to G^2 constraints. It consists of three C^2-connected pieces $\dot\tau$, $\ddot\tau$, $\dddot\tau$ of piecewise degree bi-53. Since the radial degree of each piece is 3, $J(\tau,\frac14) := J(\frac14\rho_\mathbf{2},1)$ (the C^2 prolongation of $\frac14\rho_\mathbf{2}$) and $J(\tau,1)$, defined by rotations of the sector template

$$\dot\tau_{00} := (1,0)\ ,\ \dot\tau_{10} := (1,\frac25\tan\frac{\alpha}{2}), \dot\tau_{20} := (\frac{1}{10}(9+\cos\alpha),\frac45\tan\frac{\alpha}{2}); \tag{4}$$

$$\dot\tau_{\ell 1} := \frac{11}{12}\dot\tau_{\ell 0}\ , \dot\tau_{\ell 2} := \frac56\dot\tau_{\ell 0},\quad \ell=0,1,2, \tag{5}$$

determine τ on $[0..1]\times[\frac14..1]$ as a C^2-connected map in radial direction whose outermost tensor-border is C^2 in the periodic direction.

Algorithm III: G^2 cap construction of total degree 6
Input: The tensor-border $\mathbf{b}$ and the cubic C^2 cap $\mathbf{c}$ constructed by Algorithm II; the G^2 polar parameterizations $\boldsymbol{\rho_2}$ and $\boldsymbol{\sigma_2}$.
Output: $N := 3n$ triangular patches of total degree 6, plus $2N$ patches of degree 6-5 corresponding to $v = [\frac{1}{4}..\frac{1}{2}]$ and a surrounding ring (darkest gray in Figure 10) plus $2N$ patches of degree bi-5 corresponding to $v = [\frac{1}{2}..1]$.

Algorithm: (a) Compose $\mathbf{c} \circ \frac{1}{4}\boldsymbol{\sigma_2}$ to obtain the inner cap, corresponding to $v \in [0..\frac{1}{4}]$, of N triangular Bézier patches of degree 6 (see Figure 10 *left*).
(b) Apply H^{55} to the composition $\mathbf{c} \circ \tau$ to form three intermediate bi-5 rings corresponding to $v = [\frac{1}{4}..1]$. The N patches corresponding to $v = [\frac{1}{4}..\frac{1}{2}]$ are made to match $J(\mathbf{c} \circ \frac{1}{4}\rho)$ and thereby to join with curvature continuity the inner cap constructed in (a).
(c) The outermost N patches are constructed as in Algorithm II to meet the outer tensor-border of the patches constructed in (b) in a parametrically C^2 fashion.

Since the degree of $\partial_u^j(\tau)$ at $u = 0$ is 1,1,3 for $j = 0, 1, 2$ (and symmetrically at $u = 1$, the other sector boundary) the degree of $\partial_u^j(\mathbf{c} \circ \tau)$ at $u = 0$ is 3,3,5 for $j = 0, 1, 2$. So we can apply H^{55} to obtain a surface ring of degree bi-5 that reproduces this second order expansion at both sector boundaries. At $v = \frac{1}{4}$, the 2×2 jets also coincide with that of $\mathbf{c} \circ \frac{1}{4}\boldsymbol{\rho_2}$. Therefore the construction in (b) ensures curvature continuity across $v = \frac{1}{4}$ and this completes the proof of curvature continuity of the overall construction.

The number of patches is high ($15n$ if one bisection is applied, $16n$ if we radially bisect twice as in Figure 8 *right*). Figure 11 justifies trisection in the periodic direction, echoing the theme that polar layout prefers high valencies.

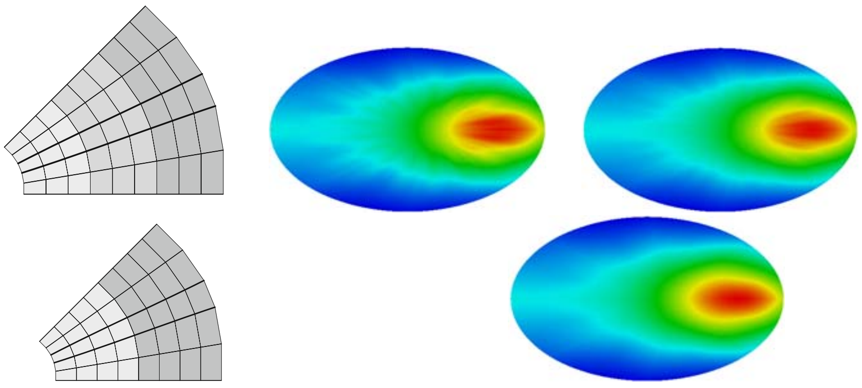


Fig. 11. Bi-6 G^2 construction of Algorithm III applied to the net of Figure 2. (*top*) (*left*) three rings of one third of a sector of τ. (*middle* and *right*) Gauss curvature of the surface is shown for three rings and a tensor-border split into (*top*, *middle*) two subsectors, (*top*, *right*) three subsectors. (*bottom*) Construction with two subsectors and two rings of degree bi-6.

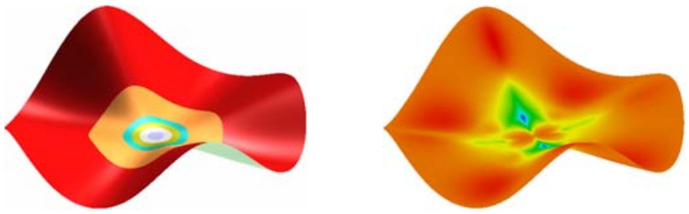


Fig. 12. Bi-5 construction. (*left*) Rings of the cap surface; (*right*) its Gauss curvature shading.

Instead of trisecting in the periodic direction, we can improve the shape by bisecting, replacing the three rings of degree 5-3 in τ by two rings of degree 5-4 and applying the operator H^{66} of [5] to obtain bi-6 patches. Figure 11 *bottom*, shows the reparameterization patchwork, reduced to $2 \times 2n$ patches instead of $3 \times 3n$ resulting in a better curvature distribution.

3.2 G^2 Capping of Degree Bi-5

A slight modification of Algorithm III yields a curvature continuous cap of degree bi-5 with one collapsed edge. This does not contradict Theorem 2 since the transitions between sectors are G^2, not parametrically C^2. Note that Algorithm IV accepts as input a tensor-border of at most degree 5.

Algorithm IV: G^2 cap construction of degree bi-5
Input: The tensor-border $\mathbf{b}$ and the cubic C^2 cap $\mathbf{c}$ constructed by Algorithm II; the G^2 polar parameterization $\rho_{\mathbf{2}}$.
Output: $5N := 15n$ patches of degree bi-5 with the N innermost having one edge collapsed.
Algorithm: (a) Apply H^{55} to $\mathbf{c} \circ \frac{1}{4}\rho_{\mathbf{2}}$ to yield an innermost bi-5 cap with collapsed edge. (b) and (c) mimic those of Algorithm III.

Since the degree of $\partial_u^j \rho_{\mathbf{2}}$, for $j = 0, 1, 2$ at $u = 0$ and $u = 1$ is 1,1,2, the operator H^{55} applied to $\mathbf{c} \circ \frac{1}{4}\rho_{\mathbf{2}}$ reproduces the second order expansion at both sector boundaries. Therefore the innermost bi-5 patches with collapsing edges form a G^2 connected cap except possibly at the pole. Since the expansion at the pole is determined by $\partial_v^j(\mathbf{q} \circ \frac{1}{4}\rho_{\mathbf{2}})$, $j = 0, 1, 2$, we can retrace the proof of curvature continuity at the pole from Theorem 1. Figure 12 shows the bi-5 patchwork corresponding to Figure 2.

4 Extensions and Discussion

The main construction, Algorithm I, generates a 6-5 cap appropriate for most geometry processing pipelines. Due to the resulting good shape, the caps were used as a guide surface in the subsequent sections. Algorithms III and IV generate

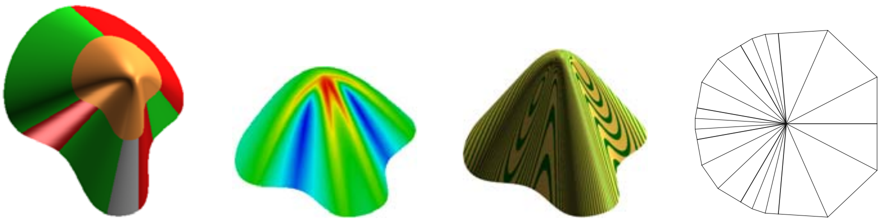


Fig. 13. (*left*) Highly non-uniform input from conical surfaces capped by Algorithm I with non-uniform ct-map. (*middle left*) Gauss curvature shading of capping; (*middle right*) highlights. (*right*) The non-uniform bi-31 ct-map.

many ($15n$) patches mainly to obtain a fair transition from G^2 connections at the central point to the C^2 input.

If the tensor-border is appropriately G^2, we can use $\boldsymbol{\rho_2}$ to replace the 6-5 cap of Algorithm I by an outer ring of patches of degree bi-4 and an inner cap of degree bi-4 with edges collapsed to the central point. A C^2 bi-3 polar subdivision can be obtained by converting the 6-5 patch to the accelerated bi-3 form of [13]. (We need only apply de Casteljeau's algorithm repeatedly in the periodic direction to obtain the required 2×2 jets in bi-3 form.)

Alternatively, sampling the corner jets of the 6-5 patches results in high quality cappings that are C^1 at the pole and C^2 away from it. We can generate such surfaces of bidegree bi-5 with one patch per sector, bi-4 with four patches per sector, or bi-3 with nine patches per sector, trading degree for number of pieces. As with the curvature bounded subdivision algorithms in [14] and the constructions displayed in Figure 11, the lower the degree, the more the curvature fluctuates.

An important bonus of the polar layout, illustrated in Figure 13, is that we can adjust the C^2 bi-31 ct-map ρ to non-uniform spacing, simply by manipulating its cubic spline boundary.

References

1. Myles, A., Karčiauskas, K., Peters, J.: Extending Catmull-Clark subdivision and PCCM with polar structures. In: PG 2007: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, Washington, DC, USA, pp. 313–320. IEEE Computer Society, Los Alamitos (2007)
2. Karčiauskas, K., Peters, J.: Bicubic polar subdivision. ACM Trans. Graph. 26(4), 14 (2007)
3. Prautzsch, H., Boehm, W., Paluzny, M.: Bézier and B-Spline Techniques. Springer, Heidelberg (2002)
4. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 10, 350–355 (1978)
5. Karčiauskas, K., Peters, J.: Concentric tesselation maps and curvature continuous guided surfaces. Computer-Aided Geometric Design 24(2), 99–111 (2007)
6. Prautzsch, H.: Freeform splines. Computer Aided Geometric Design 14(3), 201–206 (1997)

7. Reif, U.: TURBS—topologically unrestricted rational B-splines. Constructive Approximation. An International Journal for Approximations and Expansions 14(1), 57–77 (1998)
8. Loop, C.: Second order smoothness over extraordinary vertices. In: Symposium on Geometry Processing, pp. 169–178 (2004)
9. Karčiauskas, K., Peters, J.: Guided spline surfaces. Computer Aided Geometric Design, 1–20 (2009)
10. Loop, C.T., Schaefer, S.: G^2 tensor product splines over extraordinary vertices. Computer Graphics Forum (Proceedings of 2008 Symposium on Geometry Processing) 27(5), 1373–1382 (2008)
11. Bohl, H., Reif, U.: Degenerate Bézier patches with continuous curvature. Computer Aided Geometric Design 14(8), 749–761 (1997)
12. Peters, J., Reif, U.: Subdivision Surfaces. In: Geometry and Computing, vol. 3. Springer, New York (2008)
13. Karčiauskas, K., Peters, J.: Guided subdivision. Technical Report 2008-464, Dept CISE, University of Florida (2008), posted since 2005, `http://www.cise.ufl.edu/research/SurfLab/pubs.shtml`
14. Karčiauskas, K., Peters, J.: On the curvature of guided surfaces. Computer Aided Geometric Design 25(2), 69–79 (2008)

A New Approach to Point Membership Classification in B-rep Solids

Fritz Klein

The Boeing Company, USA

Abstract. A fundamental problem in computational geometry is determining whether a point is inside a B-rep solid. Methods currently used for such point classification are unreliable or inefficient or both. A new approach is illustrated by showing how a simple method for loops of planar curves represented by B-splines can be extended from two dimensions to three. The plan in two dimensions is to construct a polygon so that the point will be inside the loop if and only if it is inside the polygon. Once such a polygon is found, it is easy to compute its winding number with respect to the point. In three dimensions, an analogous (although more complicated) method is robust and efficient.

1 Introduction

For a computational representation of a solid to be useful, one must be able to reliably answer geometric questions about it. The most basic of such questions is whether a given point lies inside, on, or outside the boundary of a solid. The importance and difficulty of this question have been recognized for several decades [1,2,3,4].

The standard method for deciding if a point is in the interior of an n-dimensional region is to create a ray starting at the point and aiming in some arbitrary direction, and then to count the number of "crossing" intersections with the boundary of the region. The point is inside the region (or on its boundary) if and only if there are an odd number of such intersections, but it can be difficult to count them properly. A tangent intersection may or may not count as a crossing; similar problems arise when the intersection is at a point where the boundary's derivative is not continuous. For loops of curves on a plane, the algorithm by Nishita, Sederberg, and Kakimoto [5] is a reliable variant of the standard method. It never actually computes an intersection; instead, it recursively splits the curves into smaller pieces until the parity of the ray-curve intersections is certain.

For B-rep solids, there are more serious problems. The boundaries of B-rep solids are collections of trimmed surfaces, where some of the trimming curves are typically obtained by intersecting one surface with another. These intersection curves are generally computed to within some tolerance, but this doesn't guarantee that a particular ray will intersect the surfaces consistently with respect to the curves. Ensuring such consistency is very difficult. In fact, if the surfaces that comprise the boundary of the solid do not fit together perfectly, the concepts of "inside" and "outside" cannot even be defined unambiguously. However, if a B-rep solid is within some tolerance of being watertight, then there should be a practical way to classify points whose distance to the boundary of the solid is greater than that tolerance.

One approach to determining point membership classification more robustly is to create algorithms that apply to restricted classes of B-rep solids. This is the approach taken in [6] and [7], for example. An important recent method [8] applies to sweeping solids, but a reliable method for general B-rep solids is still needed.

This paper presents a new approach that is both robust and efficient. Furthermore, it can yield more information than simply whether the point is inside or outside the solid. It applies to solids whose component surfaces and boundary curves are represented by B-splines, with only a few minor restrictions.

In Section 2 of this paper, a variant of the algorithm for curves in a plane by Nishita, et al. [5] is presented; this variant will be easier to extend to three dimensions. Section 3 describes an algorithm for solids whose boundaries consist of triangular planar facets. In Section 4, that algorithm is extended to more general B-rep solids, while Section 5 covers some of the complications involved in this more general case. Section 6 describes other applications of the algorithm, and Section 7 discusses possible future improvements to it.

2 Two Dimensions

Given a planar region R determined by a boundary and a point P in the plane, we wish to determine if P is inside the boundary. Let us start by assuming that the boundary is a single loop of consistently oriented planar curves. We also assume (for now) that the loop is oriented counter-clockwise. If the curves are line segments, a well-known alternative to ray-casting is to compute the winding number of the loop as a sum of angles normalized by 2π. P is inside the boundary if the winding number is 1 and outside if the winding number is 0.

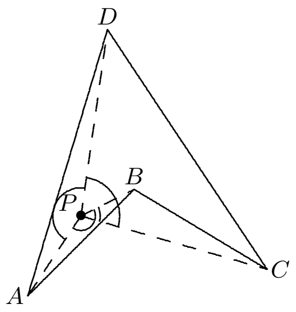


Fig. 1. Winding number calculation for a polygon

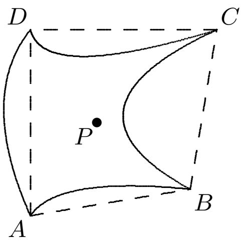


Fig. 2. Winding numbers are the same

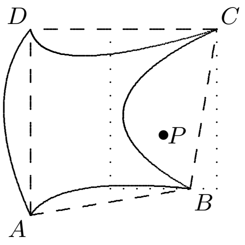


Fig. 3. Winding numbers are not the same

In Figure 1, R is bounded by the polygon $ABCD$. The winding number is the sum of the angle between PA and PB, the angle between PB and PC (which in this example happens to be negative), the angle between PC and PD, and the angle between PD and PA, all divided by 2π.

It should be noted that Poincaré found a more efficient way to compute the winding number of a point with respect to a polygon that does not involve computing any of the angles between segments; see [9] for details. However, it is not clear how to make his two-dimensional algorithm work in three dimensions, so I will not take advantage of it here.

In Figure 2, we have an example in which the winding number of a loop of 4 curves with respect to P is the same as the winding number of the polygon obtained by replacing each curve with the line segment that connects its end points. However, in Figure 3, the winding numbers for the loop of curves and the polygon are not the same.

Suppose the curves in Figures 2 and 3 are represented by B-splines. For each such curve, the minimum and maximum values of the coefficients determine a rectangle that encloses the curve. Note that for each curve in Figure 2, the rectangle that encloses the curve does not contain P. However, in Figure 3, P is inside the rectangle that encloses the curve that connects points B and C (this rectangle is drawn with dotted lines). Nishita, Sederberg, and Kakimoto [5] had the important insight that a divide-and-conquer strategy could be employed, recursively splitting such a curve into smaller pieces. We will use the following

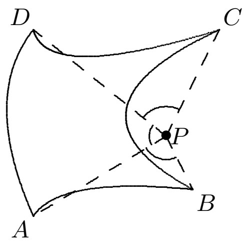


Fig. 4. Three curves that do not need to be split

variant of their method in order to find the contribution of each curve to the winding number:

1. Find the bounding rectangle that encloses the curve.
2. Check whether the bounding rectangle contains P.
   - If P is outside the bounding rectangle, measure the angle whose vertex is P, and whose first side contains the starting point of the curve and whose other side contains the ending point of the curve.
   - If P is inside the bounding rectangle, split the curve into two pieces, and add up the angles obtained by applying this procedure recursively to the two shorter curves.

We apply this procedure to each curve, add up the results, and then divide the final sum by 2π to get the winding number.

In the example of Figure 3, three of the four curves have bounding rectangles that do not contain P. The contributions of those three curves to the winding number are illustrated in Figure 4.

However, as shown in Figure 5, the curve connecting points B and C needs to be split at point E. The bounding rectangle of the curve connecting E to C does not contain P, so we can use the angle between PE and PC as before. However, P is still inside the bounding rectangle of the curve connecting B to E, so we split it at F. Now none of the bounding rectangles contain P, the positive sum of the angles in Figure 4 is negated by the sum of the angles shown in Figure 5, and we get the correct result: the winding number is 0 and P is outside the boundary.

Several properties of curves represented by B-splines are used here. Because of the convex hull properties of such curves, we can easily determine bounding rectangles for them. It is also easy to split such a curve into smaller pieces by using knot insertion. Furthermore, the bounding rectangles of the smaller curves will generally be much smaller than the bounding rectangle of the larger one. The procedure will work for curves other than those represented by B-splines provided they share these necessary properties.

Suppose P lies on one of the curves. If we had infinite precision, the procedure would never end. However, with finite precision, we would eventually get to a

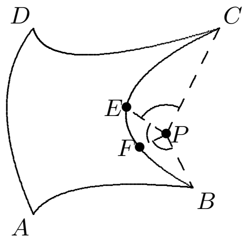


Fig. 5. One curve must be split

point at which we could no longer split the curve, and we could use this eventuality as a stopping criterion. Alternatively, we could choose to stop splitting a curve once each of the sides of its bounding rectangle is smaller than a specified minimum length; if P is still inside the bounding rectangle when this minimum size is reached, it would be categorized as being too close to the boundary to be definitely inside or outside the region.

Let us now consider how this method fares when the curves forming the boundary don't fit together very well. That is, suppose there are small gaps or overlaps between the ends. In this case, the computed winding number may not be exactly 1 or 0, but it will usually be close to one or the other of those values. The only two ways the winding number could be significantly different from both 1 and from 0 are that P could lie on one of the curves or it could be very close to a gap or overlap between two consecutive curves. In the latter case, the algorithm could simply return the value of the winding number and allow the user to decide how to handle this ambiguous situation.

An alternative for dealing with consecutive curves that don't meet precisely is to add additional line segments connecting them. Now the winding number can be computed more precisely, but this precision should not be trusted if P is within (or close to) the bounding rectangle of one of the added segments. For more on these issues, see [9].

3 Boundaries That Are All Triangular Facets

The first step in extending this algorithm to three dimensions is to note that in the two-dimensional case, the angles can be thought of as signed lengths of arcs that are portions of a unit circle centered at P. The obvious extension to three dimensions is to take each surface that is part of the boundary of a solid, and find its solid angle with respect to P. This is determined by projecting the boundaries of the surface onto the unit sphere centered at P, and computing the area of the region of the sphere bounded by these projections. If the sum of these projected areas is normalized by 4π (the surface area of the unit sphere), we have the equivalent of the winding number in three dimensions.

This idea works very nicely if all the surfaces are planar with triangular boundaries. The projected areas can be easily obtained using the Gauss-Bonnet

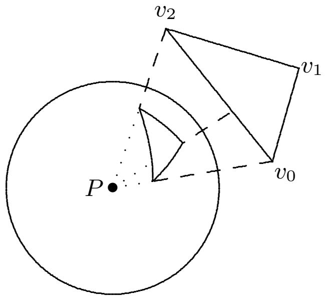


Fig. 6. Projection of a triangle onto a sphere

theorem. In fact, because the projections of straight lines are geodesics, we need only the following special case (known since the early 17th century): the area of a geodesic triangle on the unit sphere is simply the sum of its angles minus π.

It is important, of course, to be careful about orientation. In Figure 6, the vertices v_0, v_1, and v_2 are arranged counter-clockwise as viewed from the positive normal direction, and their projections onto the sphere are also counter-clockwise. In this case, the projected angles are all positive, so we add them together and subract π to get a positive area for the geodesic triangle. If the triangle were to be rotated so that the projections of the vertices are arranged clockwise on the sphere, the projected angles would then all be negative; again we would add them together, but would add π to get a negative area for the geodesic triangle.

If P lies in the plane of a triangular face but outside its boundary, the face can be ignored; its projected area is obviously zero. Two of the projected angles will be usually be zero, and the other will be π or $-\pi$; either way the calculation described in the previous paragraph will give us the correct (zero) area. Unfortunately, this calculation does not handle the case in which P is collinear with two of the vertices but not between them. Now both vertices project to the same point on the sphere and the corresponding angles are undefined. As a consequence, this collinear case needs to be checked for specifically. Note that if P is not close to the face, and close to but not on its plane, the projected area will be small enough that any numerical instability with regard to any individual angles or their signs does not affect the stability of the algorithm as a whole.

The instability that does matter, of course, arises when the point is on or close to the face. It turns out, however, that this type of instability can be handled more easily in the general case, to which we now turn.

4 General B-rep Solids

We can now extend the ideas of the previous two sections to more general B-rep solids. However, three restrictions will be imposed here; later, I will discuss the possibility of relaxing them.

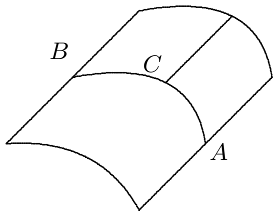


Fig. 7. Curve association

As with the two-dimensional procedure (and for the same reasons), we require that the surfaces be represented as tensor product B-splines, and that their trimming curves also be represented as B-splines. The second restriction is that each trimming curve of a surface that does not collapse to a point in model space must be associated with exactly one trimming curve of an adjoining surface (which could be itself). This requirement can be illustrated with Figure 7. If one surface with a boundary connecting points A and B borders on two surfaces, one with a boundary connecting B to C, and the other with a boundary connecting C to A, the boundary curve for the first surface needs to be split into two pieces at (or near) C, so that each piece is associated with exactly one curve from one of the other surfaces. Finally, we require each pair of adjoining curves to be parametrically aligned. This could be accomplished by representing the pair of curves as a single curve with four dependent variables (two for each surface). Such curves are produced by the surface-intersection algorithm described in [10], and can also be produced more generally by the method described in [11]. The alignment does not need to be exact; the purpose of this restriction will be discussed in Section 5.3.

One approach that would not require all these restrictions would be to project each surface onto the unit sphere whose center is P, and then to use the Gauss-Bonnet theorem to determine the area of the projection. One problem with this idea is that it would take too much time to compute and integrate the geodesic curvature of the projected boundary curves. A more serious problem is that the projections of the curves could cross one another, in which case the conditions for using Gauss-Bonnet are no longer satisified.

Instead, recall that in the two-dimensional algorithm, what we are essentially doing is finding a polygon such that P is inside the region bounded by the curves if and only if it is inside the polygon. What we will do for B-rep solids is find a polytope with triangular facets such that P is inside the solid if and only if it is inside the polytope, and then use the method of Section 3 to determine whether P is inside the polytope. However, as we extend the algorithm from two dimensions to three, we will encounter a number of additional problems that must be solved along the way.

For the remainder of this section, the B-rep solids used as examples will be bounded only by faces that could be represented as untrimmed surfaces; that is,

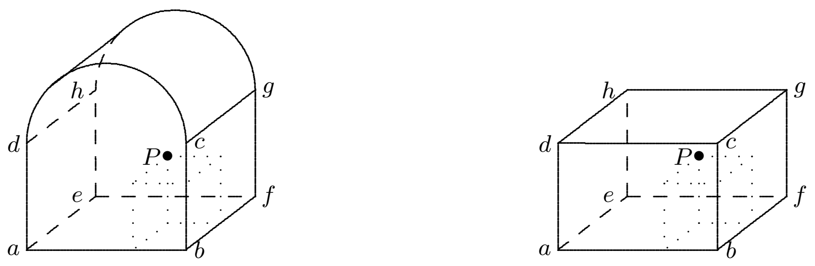


Fig. 8. No surfaces need to be split

as surface functions whose domain is rectangular with nothing trimmed away. For these examples, the issues of surface splitting and triangulation are straightforward; I will deal with more complicated faces in Section 5.

Consider the solid on the left side of Figure 8. It consists of 5 planar faces and one face that is a portion of a cylinder. The point P is located at a corner of the cube drawn in dotted lines; 3 of this cube's sides are aligned with the 3 faces of the solid that meet at point b. Note that P is not inside the bounding boxes of any of the faces. Suppose we now replace each boundary curve with a straight line segment, resulting in the object on the right side of the figure. In this particular example, this is already a polytope, but in general we would need to add line segments so that all the facets are triangular. For visual clarity, I have not drawn these additional line segments in this figure, or in the two that follow.

On the left side of Figure 8, the only 2 curves that are not already line segments are the curve connecting points c and d and the curve connecting points g and h. When these curves are replaced by line segments, the resulting polytope is simply a rectangular block; we can now classify the point with respect to the solid by classifying it with respect to the polytope. With the diagonals of each of the six sides of the block, we have a polytope with 12 triangular facets, and can use the method described in Section 3.

In Figure 9, we start with the same solid, but P is now positioned so that it is within the bounding box of the curved face. So, that face must be split. In fact, after splitting the face along the segment connecting points j and k, P is still inside the bounding box of one of the two pieces of the original face, so that piece must be split again, this time along the segment connecting points m and n. We now have 3 surfaces that are parts of the original upper surface along with the other 5 surfaces that are the same as in Figure 8. On the right side of the figure, all the curved pieces are replaced by line segments. Since the planar surface in front was never split, there must be a line segment connecting d and c, but this results in a "gap" bounded by the segments connecting d to c, c to m, m to j, and j to d. A similar gap is bounded by the segments connecting g, h, k, and n. We need to fill in these gaps by triangulating them; these triangles along with the triangulations of the 8 surfaces give us the polytope we want.

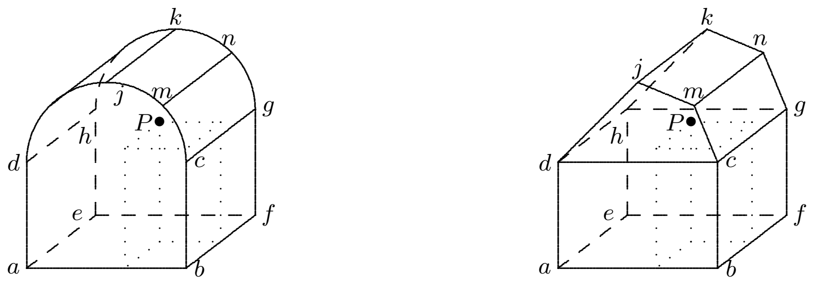


Fig. 9. One surface must be split

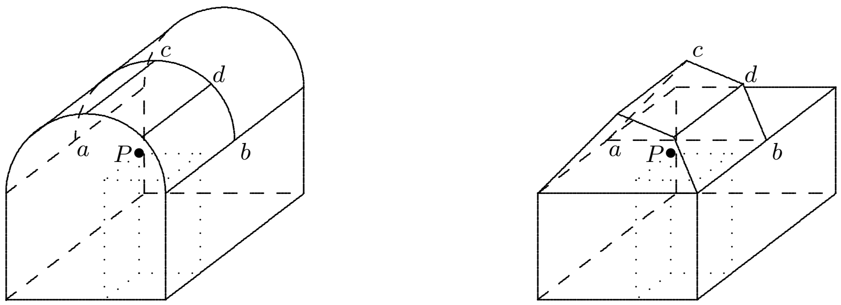


Fig. 10. Gap between pieces of same original surface

A similar gap can arise between two portions of the same surface. The solid on the left side of Figure 10 is topologically the same as the one in Figure 9, but now suppose that the first split of the cylindrical surface is perpendicular to the cylinder's axis, with the boundary between the two halves shown as the curve connecting a and b. Only one of these halves is subject to further splitting because it is the only one whose bounding box contains P. The triangulation of the unsplit half must include the line segment from a to b; there is a gap between it and the piecewise linear path that connects a and b via points c and d. This gap, as well as the gap between the front surface and the split cylindrical surface, must be triangulated along with the 5 planar faces and the 4 pieces of the cylindrical surface.

These examples suggest an algorithm that can be used when the solid fits together perfectly, with no gaps or overlaps between surfaces. As we will see, a fairly simple modification to this algorithm will allow it to be used more generally.

I will first present the recursive portion of an algorithm for finding W, the three-dimensional analog of the winding number. We will refer to this recursive portion as

Procedure R:
Given the point P, a face that is a trimmed surface, and a minimum box size,

1. Set W_f to zero.
2. Find the bounding box of the face; we will define its "size" as its maximum dimension; that is, $max(x_{max} - x_{min}, y_{max} - y_{min}, z_{max} - z_{min})$.
3. Check whether the bounding box contains P.
   – If P is outside the bounding box: determine a triangulation of the surface such that for each boundary curve, there is a side of a triangle connecting its end points. Project each triangle onto the unit sphere centered at P and add the signed areas of the geodesic triangles to W_f.
   – If P is inside the bounding box and the size of the box is less than or equal to the minimum box size: halt the process and report that P is neither definitely inside nor definitely outside the solid.
   – If P is inside the bounding box and the size of the box is larger than the miminum: split the surface into two pieces, keep track of places where the boundary curves are split, and apply Procedure R recursively for the two pieces of the surface. Then, triangulate the gap (if any) along the split and add the projected areas of these triangles to W_f.

With Procedure R in place, the whole algorithm is:

1. Set W to zero.
2. Apply Procedure R to each surface, adding each W_f to W.
3. Triangulate any gaps between surfaces, and add the projected areas of these triangles to W.
4. Divide W by 4π.

This algorithm actually works most of the time even when the solid has gaps or overlaps, but unfortunately, not always. Consider the solid in Figure 11, in which the two cylindrical faces do not quite meet precisely as shown in the exaggerated side view in Figure 12. If this solid is aligned so that the bounding box of the cylindrical surface on the right does not contain P, that surface will not be split and P will be on the wrong side of the triangulated gap between the two cylindrical surfaces.

To take care of this problem, we need a different criterion for deciding when to split a surface. One approach would be to expand the surface's bounding box in both directions of each of its dimensions by an amount greater than any potential imperfections in the solid, and to split the surface any time the point is inside this expanded bounding box. This could be a problem if one doesn't know in advance how large such imperfections might be. An alternative would be to expand the bounding box in a proportionate manner. For reasons that will become clear in Section 6, I will suggest the following criterion:

We will split the surface if the point is inside its bounding box, or if its "distance" to the bounding box is less than 0.5 times its distance to any end of a boundary curve on the surface. Rather than compute the Euclidean distance to the bounding box, we do a simpler infinity-norm type of calculation:

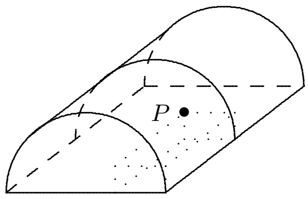


Fig. 11. Point aligned with gap between surfaces

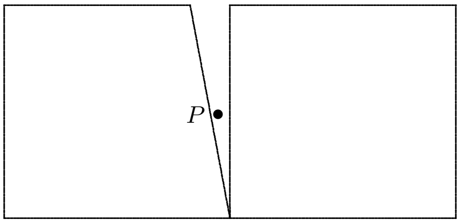


Fig. 12. Side view (exaggerated)

$max(0, P_x - x_{max}, x_{min} - P_x, P_y - y_{max}, y_{min} - P_y, P_z - z_{max}, z_{min} - P_z)$

The choice of 0.5 as the factor we multiply by the distance is somewhat arbitrary; I will discuss this issue in Section 6.

With this modification, both cylindrical surfaces in Figure 11 will be split in enough places to ensure that P is inside the polytope, provided that we are reasonably careful about the way we triangulate the gap. Details about gap triangulation are covered in Section 5.3.

5 General B-rep Solids – Algorithmic Details

The purpose of this section is to provide details about the splitting and triangulation of faces, as well as the triangulation of gaps between faces or between portions of a face. In Section 4, the examples included only faces that were simple enough that these issues were fairly obvious.

5.1 Splitting Faces

In the examples in Section 4, all the faces that needed to be split were rectangular in parameter space; I will next describe how to split faces with more complicated boundaries.

We will always split along an isoparametric line. Here is a quick way to decide at which point and in which direction to split the face that also guarantees that some boundary curve will be split. First, consider the set of all the end points of the boundary curves; if there is only one such curve, we evaluate it at its

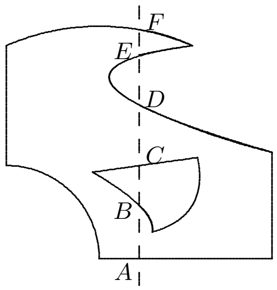


Fig. 13. Splitting boundary curves in parameter space

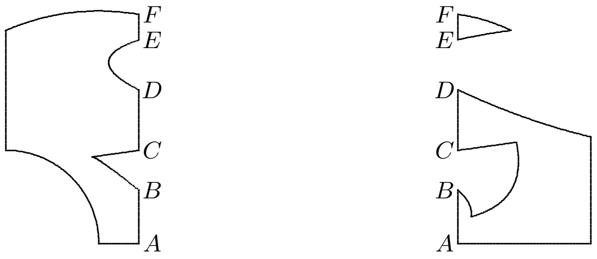


Fig. 14. New faces after splitting

parametric midpoint and add that point to the set. Now find the ranges of the u and v coefficients of this set of points. If the u range is greater, we will split along the constant u value at the center of the range, and similarly for v.

To illustrate how we split a face into two smaller faces, consider the face whose image in parameter space is shown in Figure 13. There are 9 boundary curves that form two loops, one of which is a hole inside the other. The range of u coefficients is greater than the range of v coefficients, so we wish to split along the dashed line which is constant in u. We need to find all the places where any of the boundary curves intersect the dashed line; this amounts to solving a univariate spline equation for each of the curves; see [12] for details. In this case, 4 of those equations have one solution (points A, B, C, and F) and one of the equations has two solutions (points D and E).

Figure 14 shows the new faces that result from splitting the face in Figure 13. The face on the left has two boundary curves that are identical to those in Figure 13 because they are completely to the left of the u value at which the split takes place. It also contains portions of the 5 curves that are split, as well as the line segments that connect points A to B, C to D, and E to F. The region on the right has two boundary curves identical to those in Figure 13, portions of the 5 curves that are split (although we get 2 curves from the curve that is split at D and E), and the same line segments along the split, although these 3 line segments are oriented in the opposite direction. The new face on the left has 10 boundary curves forming a single loop, while the region on the right has 11 boundary curves. The region on the right could be considered as one face with two exterior loops, or as two separate faces.

Notice that the line segments along the split that we used as new boundary curves alternate with line segments (BC and DE) that we did not need to create. This alternation is the usual case, but not if there is a tangent intersection, for example. The segments chosen are found by first sorting all the points, then checking the midpoints of potential segments to see if they are in the interior of the original face; we can use the algorithm presented in Section 2 for this purpose. Note however, that it is important that the 3 curves that form the hole in Figure 13 are oriented clockwise; then the midpoint of BC will have a winding number of 0 with respect to the complete set of boundary curves.

Finally, it should be noted that the recursive splitting of the new faces could result in the common segments AB, CD, and EF being split differently on either side, resulting in a gap that would need to be triangulated.

We have seen in this section how to split complicated faces when they need to be split. We turn next to complicated faces whose bounding boxes do not contain P, and can thus be triangulated without further splitting.

5.2 Triangulating Faces

In the examples in Section 4, each face had 4 edges, and could be triangulated simply by adding a segment connecting one pair of opposite corners. The general case is only slightly more complicated, because in this context there is no need to worry about such characteristics of the triangulation as maximum edge length or aspect ratio.

We can also get away with triangulating each loop separately, as long as we are careful about orientation. If we have an exterior boundary with a hole, the projected area that we want can be computed more easily by triangulating the region inside the exterior boundary as though the hole did not exist and then separately triangulating the hole. Again, as long as orientations of the curves are correct, the projected area will be correct.

Consider the following simple iterative method for creating the triangles we need from a given loop of consistently oriented curves. Keep in mind that we only create these triangles after we determine that the current surface does not need to be split; this means that all triangles will be separated from P by the same plane. Choose any curve and let its starting point be P_0. For each P_i, let P_{i+1} be the starting point of the curve that connects to the end of the curve starting at P_i. Then, the i_{th} triangle can consist of the line segments connecting P_0, P_{i+1}, and P_{i+2}. This is a perfectly reasonable triangulation if the region is convex in parameter space, but since all we care about is projected area, this works fine even for non-convex regions.

5.3 Triangulating Gaps

The triangulation of gaps is easily accomplished, but we can't use the same method that we use for triangulating faces. Remember that we don't triangulate a face if its bounding box contains P. However, a bounding box containing all the curves that border upon a gap could contain P, so the method used in the previous section cannot guarantee that P will be on the correct side of the triangulation of the gap.

Consider once again Figure 11. Each of the curved surfaces will be split at least twice. This will result in at least two points from each side of the gap that are higher than P, but on opposite sides of it along the gap. If those four (or more) points are triangulated, the triangles have to be above P as well. Triangulations of the remaining portions of the gap must also each be entirely on one side of P. This suggests the following general procedure:

We keep track of the points where each boundary curve is split both in model space and in the parameter space of the 4-dependent variable curve that represents the curves on either side of the gap. Suppose the points to be triangulated on one side of the gap are a_0, a_1,..., a_m and the points on the other side are b_0, b_1,..., b_n, with each list in order parametrically. Each triangle will consist of two vertices that are consecutive points on one side of the gap and a third vertex from the other side. The two vertices on the same side are shared with a triangle from the surface triangulation on that side; the orientation of the new triangle needs to be made consistent with that adjacent triangle. In order to ensure that the gap triangulation stays on the proper side of P we proceed as follows:

Start by triangulating a_0, a_1, b_0, and b_1, giving us two triangles (one or both of which could be degenerate). Now assume we have triangulated up to a_i and b_j. The next triangle has these two points as vertices; the third vertex is chosen to be either a_{i+1} or b_{j+1}, whichever has the smaller parameter value. The parametric alignment keeps each triangle on the correct side of P unless P is close to the gap. If any triangle's bounding box contains P, then P is too close to the gap to consistently determine whether it is inside the solid.

It should be noted that what I have described so far does not completely eliminate gaps between surfaces. Looking back at Figure 7, the point C could actually be three separate points, one on each of the three surfaces. Triangulating the gaps between each pair of surfaces could still leave a small triangular gap between these 3 points. Such gaps where three or more surfaces come together are not affected by surface splitting, so they don't have a significant effect on the winding number unless P is close to them. The safest course is to triangulate these small gaps anyway, and this is easily accomplished; details are left to the reader.

6 Other Applications of the Algorithm

In Section 4, two criteria were mentioned for deciding when it is necessary to split a face. The criterion I chose is not the most efficient if the only information we care about is whether a point is inside, outside or on the boundary; however, it allows us to obtain a very useful estimate of the closest distance from P to the boundary.

Let d be the closest distance from P to the boundary of a particular solid. Each time we triangulate a face or portion thereof, we can determine the distance between its bounding box and P. We can keep track of the smallest such distance L. If all faces are fully triangulated, then L is a lower bound on d; otherwise, P is inside some bounding box of a face (or portion of a face) that wasn't split, and the lower bound L is 0. We can also keep track of the closest distance of any of the points being triangulated to P. The smallest of these distances, U, is clearly an upper bound on d.

If L is 0, then U is no greater than the minimum box size specified in Procedure R; otherwise, the splitting criterion described in Section 4 ensures that $U <= 2L$. One way this could be useful is that if $L > 0$, and P lies on a second solid whose maximum diameter is less than L, then we can be certain that the boundaries of the two solids do not intersect and that the second solid is inside the first if and only if P is.

I chose 0.5 as the factor in the splitting criterion as a reasonable compromise between tighter bounds on the distance from P to the boundary versus better efficiency. Tighter bounds can be achieved by using a larger factor, but this will require more splitting of faces; as the factor apporaches 1, the amount of required splitting approaches infinity.

I will conclude this section by noting that the winding number provides more information than can be obtained by ray-casting. If a ray intersects the boundary an odd number of times, P is inside, but in this case the sign of the winding number also indicates how the boundary is oriented. Along these same lines, the method outlined in this paper could also be used to characterize points with respect to objects with self-intersecting boundaries.

7 Future Improvements

In Section 4, several restrictions were imposed on the B-rep solids for which the algorithm is valid. It should be possible to relax the restriction that adjoining curves have a common parametrization; this is a possible area of future work. Another useful extension would be to collections of faces that are not completely joined; many "solids" in use today have open edges, but it would still be useful to categorize points that aren't close to such imperfections.

The algorithm as described here is reasonably efficient, but is not optimized for speed; it would be good to have an alternate version that is much more efficient even if it lacks the feature of determining useful bounds on the distance to the boundary. One approach would be to see if the Poincaré method can be extended to three-dimensional polytopes. The extra dimension would certainly add many complications. If this is possible, then the algorithm would not need to calculate areas of projected triangles, but would still need to recursively split some of the faces, triangulate them and triangulate the gaps between them.

Another possible approach to optimizing the efficiency of the algorithm would be to find bounding boxes that are oriented so that they fit more tightly around the surfaces. It might be worth investigating whether the potential reduction in the number of face divisions would be worth the higher cost of calculating such bounds.

The essence of this paper has been the extension of a two-dimensional point membership classification method to three dimensions. An intriguing direction for future research would be to see if we can continue extending the method to four or more dimensions.

8 Conclusion

The primary contribution of this paper was to provide an algorithm for point membership classification in B-rep solids that I believe to be much more robust than standard methods.

Finally, I would like to thank Tom Grandine and Gershon Elber; my discussions with each of them were extremely helpful in developing the ideas presented here. Thanks are also due to an anonymous reviewer, whose comments resulted in several useful improvements to this paper.

References

1. Requicha, A.G.: Representations for Rigid Solids: Theory, Methods, and Systems. ACM Computing Surveys 12, 437–464 (1980)
2. Tilove, R.B.: Set Membership Classification: A Unified Approach to Geometric Intersection Problems. IEEE Transactions on Computers 10, 874–883 (1980)
3. Shapiro, V.: Well-formed set representations of solids. International Journal of Computational Geomtry and Applications 2, 125–150 (1995)
4. Shapiro, V.: Solid Modeling. Handbook of Computer Aided Geometric Design, 473–518 (2002)
5. Nishita, T., Sederberg, T.W., Kakimoto, M.: Ray Tracing Trimmed Rational Surface Patches. Computer Graphics 24, 337–345 (1990)
6. Sanchez-Reyes, J.: Quasinonparametric surfaces. Computer Aided Design 27, 263–275 (1995)
7. Segura, R., Feito, F.R., Ruiz de Miras, J., Torres, J.C., Ogáyar, C.: An Efficient Point Classification Algorithm for Triangle Meshes. Journal of Graphics Tools 10, 27–35 (2005)
8. Erdim, H., Ilies, H.T.: Classifying points for sweeping solids. Computer Aided Design 40, 987–998
9. Qi, J., Shapiro, V., Stewart, N.F.: Single-set and class-of-sets semantics for geometric models. Computer Aided Design 38, 1088–1098 (2006)
10. Grandine, T., Klein, F.W.: A new approach to the surface intersection problem. Computer Aided Geometric Design 14, 111–134 (1997)
11. Cohen, S., Elber, G., Bar-Yehuda, R.: Matching of Freeform Curves. Computer Aided Design 20, 369–379 (1997)
12. Grandine, T.: Computing zeros of spline functions. Computer Aided Geometric Design 6, 129–136 (1989)

Probabilistic Modeling and Visualization of the Flexibility in Morphable Models

M. Lüthi, T. Albrecht, and T. Vetter

Computer Science Department, University of Basel, Switzerland
{marcel.luethi,thomas.albrecht,thomas.vetter}@unibas.ch

Abstract. Statistical shape models, and in particular morphable models, have gained widespread use in computer vision, computer graphics and medical imaging. Researchers have started to build models of almost any anatomical structure in the human body. While these models provide a useful prior for many image analysis task, relatively little information about the shape represented by the morphable model is exploited. We propose a method for computing and visualizing the remaining flexibility, when a part of the shape is fixed. Our method, which is based on Probabilistic PCA, not only leads to an approach for reconstructing the full shape from partial information, but also allows us to investigate and visualize the uncertainty of a reconstruction. To show the feasibility of our approach we performed experiments on a statistical model of the human face and the femur bone. The visualization of the remaining flexibility allows for greater insight into the statistical properties of the shape.

1 Introduction

Morphable models, i.e. statistical shape models based on dense point-to-point correspondence, have become a widely used tool in computer vision, computer graphics and medical imaging. The main idea behind a morphable model is to span a space of shapes (3D surfaces) by taking linear combinations of example shapes [1]. A probability distribution is estimated from the example shapes, quantifying the probability of observing each linear combination. The most common use of morphable models is to restrict the solution-space of ill posed problems by penalizing unlikely instances of the shape. Typical examples include image segmentation [2,3,4], registration [5,6] or 2D-3D surface reconstruction [7,8,9]. In this context, the model is used to answer the following question:

– Given a shape, how likely is it that the shape belongs to the object class represented by the morphable model?

These applications exploit only the fact that the variability of the shape as a whole can be represented and quantified by the morphable model.

In this paper we are trying to get a deeper understanding of the information a morphable model represents and how one part of the model influences the rest. The central question we are trying to answer is:

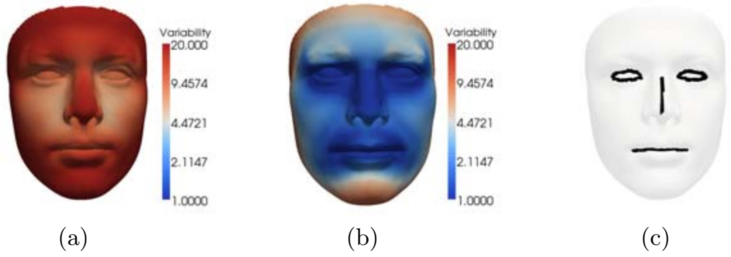


Fig. 1. Flexibility of a morphable model of the human face. The colors represent the variability (in mm) for each point. Figure (a) shows the full flexibility of the morphable model. In (b), the most likely reconstruction of the sketch depicted in (c) is shown, together with the remaining variability.

– Given only a part of a shape, what is the most likely completion of this shape and how much variance remains in the model given this partial information?

We illustrate this with an example. Assume we are given a morphable model of the human face. Let s be a random variable representing the surface, with its distribution given by the morphable face model. Figure 1(a) shows the mean face $E[s]$ and the variability represented by the model (which is, loosely speaking, the variance $\text{var}(s)$). Now suppose we are given a rough sketch of the eyes, nose and mouth in form of the black lines in Figure 1(c) and wish to reconstruct a full face from these lines. Denoting the given lines by s_b, we are interested in the distribution of the random variable $s|s_b$. Figure 1(b) shows the most likely reconstruction $s^* := \arg\max_s p(s|s_b)$ of the full shape, as well as the remaining variability $\text{var}(s|s_b)$. Naturally, the shape variability is much lower than in Figure 1(a) because the sketch s_b is observed. Hence, knowledge of the distribution of $s|s_b$ not only leads to an approach for the reconstruction of a face from the sketch, but, equally interesting, indicates how well the face is determined by the sketch. In the remainder of the paper we show how these quantities can be computed, under the assumption that the shapes follow a normal distribution.

One main assumption of morphable models is that the shapes, which are usually represented as very high dimensional vectors, lie on an embedded linear manifold (i.e. plane) within this high dimensional shape space. This manifold is found by performing Principal Component Analysis (PCA) of the sample covariance matrix. Unfortunately, standard PCA does not provide a probability model in the shape space [10]. In particular, in our “high-dimension - small sample” case, the covariance matrix becomes singular, which leads to various problems in statistical reasoning. In the approach presented in this paper, we use Probabilistic PCA (PPCA) [11], which defines a proper covariance structure in the shape space. The PPCA approach also directly implies a method for reconstructing a full shape given only partial information by using the posterior mean as the best reconstruction.

Knowledge of the best reconstruction and remaining variability of a shape is important in many application domains. We are particularly interested in the area of reconstructive medicine, where the problem arises frequently that a traumatised structure has to be reconstructed to fit the remaining parts. Being able to asses the remaining flexibility in a statistically sound way is important for the planning of reconstructive surgical interventions and the prediction of the outcome. Further, it is of general interest to know how much the different parts of an anatomical structure determine its variability. This knowledge can for example give important clues for designing implants and prostheses.

Several authors have proposed very similar methods for reconstruction of a full surface from partial information [12,13,14,15,16]. In fact, the reconstruction method resulting from the PPCA approach encompasses the one proposed by Blanz and Vetter [12], and Basso and Vetter [13] as a special case. A similar model based on factor analysis, which strongly resembles our PPCA model, was proposed by Machade et al. in [17]. However, to the best of our knowledge, the PPCA framework for model based reconstruction and the use of the full posterior distribution for modeling and visualizing the remaining flexibility has not been considered before. Indeed, the only work we are aware of that explicitly tries to model the remaining flexibility is the one by Albrecht et al. [18]. In contrast to our work it does not admit a probabilistic interpretation and requires a separate algorithm for shape reconstruction.

2 Background

Before we present our model, we briefly review the mathematical concepts we will use in the remainder of the paper. In order to apply statistical methods to shapes we need to be able to represent them as random variables. A particularly simple approach, is to sample the shape and organize the sampled points as a vector in Euclidean space. Such a vector is usually referred to as a shape vector.

Two types of statistical shape models are distinguished in the literature. In the *Active Shape Models* [19], the shape is given as a 2D contour and is relatively sparsely sampled. In contrast, in the *Morphable Model* [1], the shape is given as a 3D surface and the sampling is dense. From a mathematical point of view, the concepts are the same. An important difference is, however, that in the case of the Morphable Model the dimensionality of the shape vectors is much larger than the number of observations. It is this property that motivates our work, and we will therefore focus only on this case in the remainder of this paper.

2.1 Shape Vectors and Registration

Let $\{\Gamma_i \subset \mathbb{R}^3 | i = 1, \ldots, n\}$ be n surfaces, given in some suitable representation. Define an arbitrary surface, say Γ_1, as a reference surface. We assume that each surface Γ_i was obtained by warping the reference surface Γ_1 with a smooth vector field $\phi_i : \Gamma_1 \to \mathbb{R}^3$. That is

$$\Gamma_i = \{x + \phi_i(x) | x \in \Gamma_1\}.$$

Let $\hat{\Gamma}_1$ be a suitable discretization of Γ_1 of N points (e.g. $\hat{\Gamma}_i$ is represented as a triangle mesh with N vertices). Note, that the same discretization is induced by the mapping ϕ for each surface Γ_i. We define the shape vector $s_i \in \mathbb{R}^{3N}$ as

$$s_i = (v_x^{i,1}, v_y^{i,1}, v_z^{i,1}, \ldots, v_x^{i,N}, v_y^{i,N}, v_z^{i,N})^T,$$

where the vector $v^{i,j} = (v_x^{i,j}, v_y^{i,j}, v_z^{i,j})$ represents the x, y, z coordinates of the j-th vertex of $\hat{\Gamma}_i$.

Usually we are given the surfaces $\Gamma_1, \ldots, \Gamma_n$ rather than a reference surface Γ_1 and the corresponding vector fields $\{\phi_i\}_{i=2}^n$. Finding a vector field ϕ that maps between a given pair of surfaces is a central problem in medical imaging and computer vision and is referred to as the *registration* or *correspondence* problem. Many algorithms for surface registration have been described in the literature (see e.g. [20] for an overview).

2.2 Principal Component Analysis and Statistical Shape Models

PCA is a well known and widely used method for dimensionality reduction and data visualization. From n data sets, represented by vectors $s_i \in \mathbb{R}^m$ the mean $\mu = \frac{1}{n}\sum_{i=1}^n s_i$ and covariance matrix $\Sigma \in \mathbb{R}^{m\times m}$ with $\Sigma = \frac{1}{n}\sum_{i=1}^n (s_i - \mu)(s_i - \mu)^T$ can be estimated. PCA consists of an eigenvalue decomposition or *principal component* transformation of Σ:

$$\Sigma = UD^2U^T, \tag{1}$$

where $U \in \mathbb{R}^{m\times m}$ is the orthonormal matrix of the eigenvectors of Σ, ordered according to the size of the corresponding eigenvalues, which make up the diagonal of the matrix $D^2 = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_m^2)$. Note that if $n < m$, we have $\sigma_i = 0$ for all $i > n$.

When building a PCA-based shape model, it is assumed that the training datasets s_i and linear combinations thereof form a linear space of shapes that can be modelled by a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$. With the help of a coefficient vector α, each shape can be represented as:

$$s = s(\alpha) = UD\,\alpha + \mu. \tag{2}$$

Thanks to this representation the probability density function $\mathcal{N}(\mu, \Sigma)$ evaluated at $s(\alpha)$ takes the form:

$$p(s(\alpha)) = \tfrac{1}{z}\exp(-\|\alpha\|_2^2), \tag{3}$$

where $z = \sqrt{(2\pi)^m \det(D)}$ is a normalization factor [12].

2.3 Linear Gaussian Models

The PPCA model considered in this paper is a linear Gaussian model of the form

$$y = Ax + b + \varepsilon, \tag{4}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are parameters, $x \sim \mathcal{N}(\mu, \Lambda)$ and $\varepsilon \sim \mathcal{N}(0, L)$ are normally distributed random variables. For this model, the predictive distribution $p(y)$ and posterior distribution $p(x|y)$ have a simple analytic form, as summarized in the following theorem [21]:

Theorem 1 (Theorem for Linear Gaussian Models). *Given a marginal Gaussian distribution for x and a conditional distribution for $y|x$ in the form*

$$p(x) = \mathcal{N}(\mu, \Lambda) \tag{5}$$

$$p(y|x) = \mathcal{N}(Ax + b, L) \tag{6}$$

the marginal distribution of y and the conditional distribution of x given y are given by

$$p(y) = \mathcal{N}(A\mu + b, L + A\Lambda A^T) \tag{7}$$

$$p(x|y) = \mathcal{N}(\Sigma A^T L^{-1}(y - b) + \Lambda^{-1}\mu, \Sigma) \tag{8}$$

where

$$\Sigma = (\Lambda + A^T L A)^{-1}. \tag{9}$$

It is the fact that we have an analytic form of the posterior distribution that allows us in the following to model and visualize the remaining variability.

3 Shape Modeling Using Probabilistic PCA

In this section we present our PPCA based method for modeling the shape variability and show how it leads to a natural approach for shape reconstruction. Further, we show how the resulting posterior distribution can be used to visualize effectively the remaining flexibility in the model.

3.1 Mathematical Model

The mathematical model we use for shape modeling is obtained by applying standard Probabilistic Principal Component Analysis, as proposed by Tipping and Bishop [11], to a set of surfaces represented as shape vectors.

Let $\{s_i \in \mathbb{R}^{3N}\}_{i=1}^n$ be n shape vectors as defined in section 2.1. The main assumption in PPCA is that the high dimensional observations can be explained by a mapping from a low dimensional latent space plus some additional Gaussian noise. Let α be a d-dimensional latent variable

$$p(\alpha) = \mathcal{N}(0, \mathcal{I}_d). \tag{10}$$

We model the conditional distribution of observing the shape vector s as

$$p(s|\alpha) = \mathcal{N}(W\alpha + \mu, \sigma^2 \mathcal{I}_{3N}) \tag{11}$$

where $W \in \mathbb{R}^{3N \times d}$ is a linear mapping and $\mu \in \mathbb{R}^{3N}$ a shape vector. We can interpret this as a generative model, where the shape s is given by the mapping

W of the latent variable α plus some additive Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma \mathcal{I}_{3N})$. That is

$$s = W\alpha + \mu + \varepsilon. \tag{12}$$

According to Theorem 1 the predictive distribution

$$p(s) = \int p(s|\alpha)p(\alpha)\, d\alpha. \tag{13}$$

is again Gaussian with mean μ and covariance matrix $WW^T + \sigma^2\mathcal{I}_{3N}$. In summary, we obtain the distribution

$$p(s) = \mathcal{N}(\mu, WW^T + \sigma^2\mathcal{I}_{3N}). \tag{14}$$

Theorem 1 also provides us with an expression for the posterior distribution:

$$p(\alpha|s) = \mathcal{N}(M^{-1}W^T\sigma^{-2}(s-\mu), M^{-1}), \tag{15}$$

where

$$M = \sigma^{-2}W^TW + \mathcal{I}_d.$$

Tipping and Bishop [11] showed that the maximum likelihood solution for the parameters μ, W, σ^2 is

$$\mu_{\text{ML}} = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{16}$$

$$W_{\text{ML}} = U_d(D_d^2 - \sigma^2\mathcal{I}_d)^{\frac{1}{2}} \tag{17}$$

$$\sigma_{\text{ML}} = \frac{1}{3N-d}\sum_{i=d+1}^{3N} D_{ii}^2. \tag{18}$$

Here, $U_d \in \mathbb{R}^{3N\times d}$ and $D_d \in \mathbb{R}^{d\times d}$ are the sub-matrices corresponding to the d largest eigenvalues of the covariance matrix decomposition in (1). Using these maximum likelihood estimates in the generative model (12) yields a striking similarity with the PCA model (2), which gives the method its name. However, in contrast to the standard PCA, PPCA provides a fully probabilistic model. This allows for the computation of the full posterior distribution and to deal with missing data in a principled way.

3.2 Missing Data

Assume now that a part of the model is given. Without loss of generality, the model can be partitioned as $s = (s_a, s_b)$ with s_b given and s_a unknown. We would like to reconstruct the full shape $s \in \mathbb{R}^{3N}$ from the partial shape $s_b \in \mathbb{R}^{3\tilde{N}}$. Equation 14 can be written as

$$p(s) = p(s_a, s_b) = \mathcal{N}(\begin{bmatrix}\mu_a \\ \mu_b\end{bmatrix}, \begin{bmatrix}W_aW_a^T & W_aW_b^T \\ W_bW_a^T & W_bW_b^T\end{bmatrix} + \sigma\mathcal{I}_{3\tilde{N}}). \tag{19}$$

Using the well known formula for the conditional distribution of a multivariate normal distribution, we have

$$p(s_a|s_b) = \mathcal{N}(\mu_{s_a|s_b}, \Sigma_{s_a|s_b}) \tag{20}$$

with

$$\mu_{s_a|s_b} = \mu_a + W_a W_b^T (W_b W_b^T + \sigma \mathcal{I}_{3\tilde{N}})^{-1}(s_b - \mu_b) \tag{21}$$

and

$$\Sigma_{s_a|s_b} = (W_b W_b^T + \sigma \mathcal{I}_{3\tilde{N}}) - W_a W_b^T (W_b W_b^T + \sigma \mathcal{I}_{3\tilde{N}})^{-1} W_b W_a^T). \tag{22}$$

The above equations give us a simple recipe for shape reconstruction. Unfortunately, it requires the inversion of the matrix $(W_b W_b^T + \sigma \mathcal{I})$, which is potentially huge. Using the fact that the shape is determined by the latent variables, we instead try to find an expression for $p(\alpha|s_b)$:

$$p(s_b|\alpha) = \mathcal{N}(W_b\alpha + \mu_b, \sigma^2 \mathcal{I}_{3\tilde{N}}) \tag{23}$$

$$p(\alpha) = \mathcal{N}(0, \mathcal{I}_d) \tag{24}$$

Again, we are in the position to apply Theorem 1:

$$p(\alpha|s_b) = \mathcal{N}(M^{-1} W_b^T \sigma^{-2}(s_b - \mu_b), M^{-1}), \tag{25}$$

with

$$M = \sigma^{-2} W_b^T W_b + \mathcal{I}_d. \tag{26}$$

In all practical cases, $W_b^T W_b$ will be small and can easily be computed. Once α has been determined, the most likely shape s^* is given by

$$s^* = \arg\max_s p(s|\alpha) = W\alpha + \mu. \tag{27}$$

This reconstruction is sufficient for the majority of shape modeling applications. Hence, we hardly ever need to compute the full covariance matrix $\Sigma_{s_a|s_b}$. It is, however, interesting to write down the distribution $p(s_a|s_b)$ in terms of the latent variables:

$$p(s|s_b) = p(s_a|s_b) = \int p(s_a|\alpha, s_b) p(\alpha|s_b)\, d\alpha = \int p(s_a|\alpha) p(\alpha|s_b)\, d\alpha \tag{28}$$

where we used the fact that s_a and s_b are conditionally independent given α. This can be interpreted as a projection of the observation onto the latent space, followed by the reconstruction of the full shape for the given α.

3.3 Reconstruction of Partial Shapes

We show now how the results from Section 3.2 can be used to reconstruct missing parts of any shape that can be modeled by a morphable model. In order to model the partial shape s_b as a part of a given complete morphable model, it has to be in correspondence with the reference shape (cf. Section 2.1).

The latent variable $\alpha|s_b$ is distributed according to Equation (25). The most probable reconstruction is obtained by reconstructing the full shape from the mean according to Equation (27). In addition to providing us with the most probable reconstruction, $p(\alpha|s_b)$ describes the distribution for all possible reconstructions. By considering how strongly this distribution is concentrated around its mean, we see exactly how reliable the reconstruction with the mean is. In effect, $p(\alpha|s_b)$ models the remaining flexibility of the morphable model given the fixed part s_b. In the next section, we will show how this flexibility can be explored visually.

3.4 Visualizing the Remaining Variability

We reconstruct the shapes from the latent variable α using Equation (27), i.e. $s^* := W\alpha + \mu$. According to the standard formula the covariance matrix of s^* under this affine transformation becomes

$$WM^{-1}W^T. \tag{29}$$

Note that this is a simpler distribution than the $p(s|s_b)$ given in Equation (20), as we can ignore the Gaussian noise ε of the original model (12) here.

In order to visualize the main modes of variation of this distribution, we perform an additional PCA, i.e. an eigenvalue decomposition of the covariance matrix $WM^{-1}W^T$. When we choose the maximum likelihood estimator $W = W_{\mathrm{ML}}$ from Equation (17), the covariance matrix decomposes as follows:

$$W_{\mathrm{ML}} M^{-1} W_{\mathrm{ML}}^T = U_d \, (D_d^2 - \sigma^2 \mathcal{I}_d)^{\frac{1}{2}} M^{-1} (D_d^2 - \sigma^2 \mathcal{I}_d)^{\frac{1}{2}} \, U_d^T \,. \tag{30}$$

By computing a $(d \times d)$-dimensional eigenvalue decomposition of the inner part

$$(D_d^2 - \sigma^2 \mathcal{I}_d)^{\frac{1}{2}} M^{-1} (D_d^2 - \sigma^2 \mathcal{I}_d)^{\frac{1}{2}} =: \tilde{U} \tilde{D}^2 \tilde{U}^T, \tag{31}$$

we get the eigenvalue decomposition

$$W_{\mathrm{ML}} M^{-1} W_{\mathrm{ML}}^T = (U_d \tilde{U}) \, \tilde{D}^2 \, (U_d \tilde{U})^T, \tag{32}$$

without having to calculate a prohibitively large $(3N \times 3N)$-dimensional PCA.

The main modes of variation of the shapes s^* are the eigenvectors corresponding to the largest eigenvalues. They model those deformations of the shapes causing a maximum deformation of the full shape, while keeping the given part s_b virtually fixed.

By visualizing these modes of variation, we can see how much flexibility remains in the model after fixing s_b and thus how well s_b determines the rest of the shape. For instance, the eigenvector v_1 corresponding to the largest eigenvalue σ_1^2 is the unique deformation with unit norm that changes the full model s as much as possible, while keeping s_b fixed within the limits of the noise modeled by ε. By visualizing $\mu \pm 3\lambda_1 v_1$ we can observe 99 % of the variation along this first mode of variation.

The Parameter σ was introduced in Equation (11) as the variance of additive noise assumed to be present in the model. A maximum likelihood estimator was given in Equation (18). When reconstructing partial shapes and examining the remaining flexibility, σ controls how strictly the given part of the model s_b is matched.

The larger σ is chosen, the more the shape is allowed to deviate from s_b. Of course this results in a larger remaining flexibility as even the parts of the shape constrained by s_b are allowed to move slightly.

The maximum likelihood estimator for σ given in Equation (18) is:

$$\sigma_{\mathrm{ML}} = \frac{1}{3N-d} \sum_{i=d+1}^{3N} D_{ii}^2 .$$

The number of non-zero eigenvalues is usually small compared to the dimensionality. Therefore, the maximum likelihood solution σ_{ML} becomes small or even zero, which will lead to the covariance matrix M in equation (26) being (close to) singular. This results in little or no remaining flexibility as well as possible overfitting in the reconstructed shapes.

Letting $\sigma \to 0$ in Equation (17) and (14), leads to $W = U_d D_d$ and thus the use of the sample covariance matrix $U_d D_d^2 U_d^T$ as our covariance estimator. It is well known in statistics that in the "small sample, large dimension" case, the maximum likelihood estimator of the covariance matrix provides a poor estimate of the real covariance matrix. Letting $\sigma^2 > 0$ be a parameter corresponds to the usual shrinkage approach for covariance estimation and can be shown to improve the estimate in such cases (see e.g. Schäfer and Strimmer [22]).

In a practical setting, σ is a very sensible parameter and has the natural interpretation as controlling the balance between matching accuracy and flexibility. So instead of the very small σ_{ML}, it can for instance be chosen to match the measurement error of the capturing device.

4 Results and Medical Applications

We conducted a number of experiments using statistical models of the human face and the femur bone. The experiments show how the model can be used to answer typical questions that arise in clinical practice.

4.1 Experimental Setup

For the femur experiments we used a statistical model built from 50 surfaces of normal femurs. To establish correspondence among the surfaces, we used the non-rigid registration algorithm proposed by Albrecht et al. [5]. For the face experiments we used the data from the Basel Face Model[1], which consists of 200 registered faces, acquired with a structured light scanner. All the experiments have been performed with the parameter $\sigma = 10$. This value has deliberately been chosen relatively large, to make the variations clearly visible in the paper.

[1] Basel face model: *http://faces.cs.unibas.ch*

4.2 Results

For our first experiment we considered the case where the nose is missing in a face and has to be reconstructed. This is a case that actually occurs in clinical practice, for example where a large tumor has to be removed. With our method the reconstruction can be efficiently computed, requiring only a surface scan of the patient. Figure 4.2 shows the reconstruction results as well as the variability represented by the first mode of variation. It can be seen that the reconstruction closely resembles the original nose. This is an indication that the shape of the nose is rather well constrained, given the remaining facial surface. Figure 2(f) shows an extremely unlikely reconstruction (with probability less than 10^{-13}). However, even such an unlikely reconstruction still looks natural.

Our second experiment shows that a valid reconstruction is also possible when only a small part of the face is fixed. Figure 3 shows the reconstruction and the variation captured by the first two principal components. In Figure 3(b) the variability that remains for each point is color coded. The variability σ_{v^i} for the point v^i is defined as

$$\sigma_{v^i} = \sqrt{\operatorname{var}\left(v_x^i\right)^2 + \operatorname{var}\left(v_y^i\right)^2 + \operatorname{var}\left(v_z^i\right)^2},$$

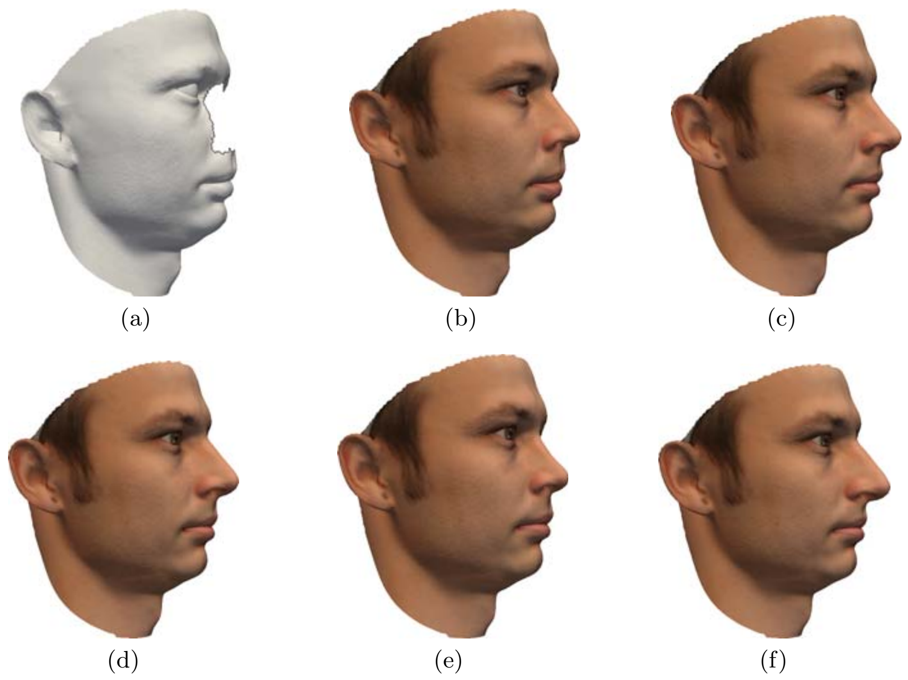


Fig. 2. Reconstruction of a nose: (a) shows the surface with the nose removed. (b) shows the real face while (c) shows the reconstructed nose. In (d) and (e) we see the $\pm 3\sigma$ of the main mode of variation. (f) shows a nose where the first 7 components are 3σ from the mean.

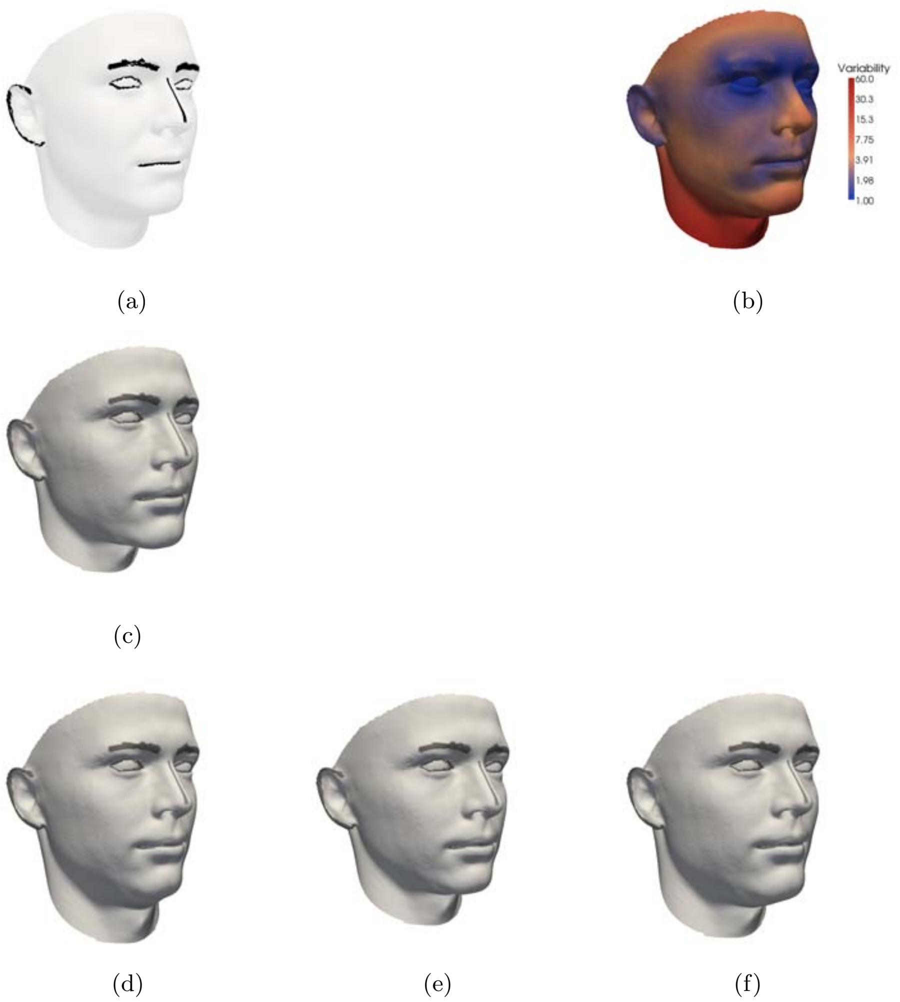


Fig. 3. Reconstruction of the face where only a sketch (a) is given. (b) shows the best reconstruction. The colors encode the variability (in mm) at the given point. (d), (c) show $\pm 3\sigma_1$ in the first mode of variation. (f), (e) show $\pm 3\sigma_2$ in the second mode of variation.

i.e. it is the norm of the variance measured in each direction. Of course, the reconstruction from only a sketch shows much more variability than what can be observed in the nose example. The last experiment shows how the model can be used to investigate how well the femur bone is determined by the joint surfaces. This variability can be helpful in prosthesis design. Figure 4 shows the variability in the direction of the first two principal components.

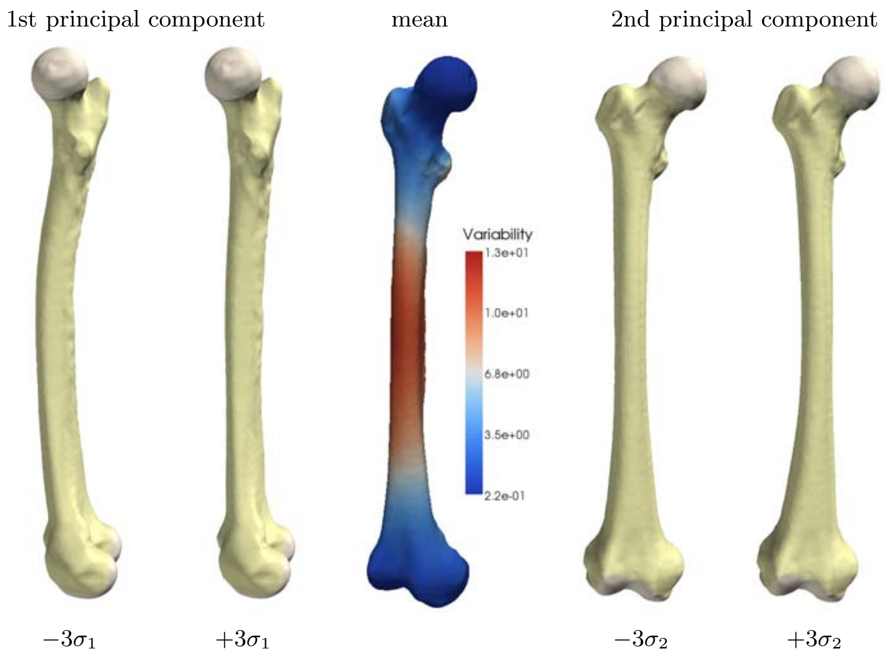


Fig. 4. When a statistical model of the human femur bone is fitted to given joint surfaces (grey), considerable flexibility remains, visualized here by the first two principal components with standard deviation $\sigma_{1,2}$. The joint surfaces are taken from the mean, seen in the middle, colored according to the remaining variability (in mm).

4.3 Reconstruction in Practice

For all reconstruction examples that we presented in this section the surfaces were already in correspondence with the model. In practice, establishing the correspondence is a pivotal step that both influences the reconstruction error and the remaining variability. As this paper's main focus is on modeling the remaining flexibility and not the reconstruction of missing parts, we refer the reader to the article by Basso and Vetter [13] for a more thorough evaluation of the actual reconstruction using an equivalent method.

5 Conclusion

We presented a method for computing and visualizing the remaining flexibility in statistical shape models, when part of the shape is known. To model the shape variability we use a probabilistic model based on PPCA. The flexibility remaining in the model can be computed from the posterior distribution arising from the PPCA model. We proposed to compute the remaining flexibility by evaluating the principal components of this posterior distribution and showing the effect that changing the corresponding coefficients has on the shape. We presented experiments that illustrate typical applications of our model. Our results for shape reconstruction are similar to those achieved in comparable reconstruction approaches. However, in contrast to other methods, we also showed in our experiments the full range of possible reconstructions that complete the given partial shape. Furthermore, our method allows us to assign a probability to every reconstruction.

The main application of our method is to gain more insight into the information represented by a morphable model, and learn more about the statistical properties of the surfaces. The model allows us to investigate how strictly a given part determines a shape. This is of particular interest in the medical domain, where such questions frequently arise in the planning of reconstructive surgeries as well as the designing of prosthesis.

In future work we will investigate the question whether it is possible to automatically find the parts of the surface that best determine its shape. A particular application we have in mind is to use this information in face modelling, for investigating which parts of the face determine the identity of a person and which parts constitute the "remaining flexibility".

Acknowledgements

This work was funded by the Swiss National Science Foundation in the scope of the NCCR CO-ME project 5005-66380 and the Hasler Foundation in scope of the HOVISSE project. The face data is courtesy of Pascal Paysan (Figure 2(b)), University of Basel.

References

1. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D faces. In: SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 187–194. ACM Press, New York (1999)
2. Leventon, M.E., Grimson, W.E.L., Faugeras, O.: Statistical shape influence in geodesic active contours. In: CVPR, vol. 1, p. 1316 (2000)
3. Tsai, A., Yezzi, A., Wells, J.W., Tempany, C., Tucker, D., Fan, A., Grimson, W.E., Willsky, A.: A shape-based approach to the segmentation of medical imagery using level sets. IEEE Transactions on Medical Imaging 22(2), 137–154 (2003)
4. Cremers, D., Rousson, M., Deriche, R.: A Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape. International Journal of Computer Vision 72(2), 195–215 (2007)
5. Albrecht, T., Lüthi, M., Vetter, T.: A statistical deformation prior for non-rigid image and shape registration. In: IEEE Conference on CVPR, pp. 1–8 (2008)
6. Wang, Y., Staib, L.: Physical model-based non-rigid registration incorporating statistical shape information. Medical Image Analysis 4(1), 7–20 (2000)
7. Benameur, S., Mignotte, M., Destrempes, F., De Guise, J.: Three-dimensional biplanar reconstruction of scoliotic rib cage using the estimation of a mixture of probabilistic prior models. IEEE Transactions on Biomedical Engineering 52(10), 1713–1728 (2005)

8. Lamecker, H., Wenckebach, T., Hege, H.: Atlas-based 3D-shape reconstruction from X-ray images. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), vol. 1, pp. 371–374 (2006)
9. Fleute, M., Lavallee, S.: Nonrigid 3-D/2-D Registration of Images Using Statistical Models. In: Taylor, C., Colchester, A. (eds.) MICCAI 1999. LNCS, vol. 1679, pp. 138–147. Springer, Heidelberg (1999)
10. Roweis, S.: EM Algorithms for PCA and SPCA. In: NIPS, pp. 626–632 (1998)
11. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. Journal of the Royal Statistical Society 61, 611–622 (1999)
12. Blanz, V., Mehl, A., Vetter, T., Seidel, H.P.: A statistical method for robust 3d surface reconstruction from sparse data. In: International Symposium on 3D Data Processing, Visualization and Transmission (2004)
13. Basso, C., Vetter, T.: Statistically Motivated 3D Faces Reconstruction. In: Proceedings of the 2nd International Conference on Reconstruction of Soft Facial Parts, p. 71 (2005)
14. Hwang, B., Lee, S.: Reconstructing a Whole Face Image from a Partially Damaged or Occluded Image by Multiple Matching. In: Lee, S.-W., Li, S.Z. (eds.) ICB 2007. LNCS, vol. 4642, pp. 692–701. Springer, Heidelberg (2007)
15. Jensen, K., Sporring, J.: Reconstructing Teeth with Bite Information. In: Ersbøll, B.K., Pedersen, K.S. (eds.) SCIA 2007. LNCS, vol. 4522, pp. 102–111. Springer, Heidelberg (2007)
16. Gong, X., Wang, G.: A Novel Deformation Framework for Face Modeling from a Few Control Points. In: Wang, G., Li, T., Grzymala-Busse, J.W., Miao, D., Skowron, A., Yao, Y. (eds.) RSKT 2008. LNCS, vol. 5009, pp. 434–441. Springer, Heidelberg (2008)
17. Machado, A., Gee, J., Campos, M.: Visual data mining for modeling prior distributions in morphometry. IEEE Signal Processing Magazine 21(3), 20–27 (2004)
18. Albrecht, T., Knothe, R., Vetter, T.: Modeling the Remaining Flexibility of Partially Fixed Statistical Shape Models. In: Workshop on the Mathematical Foundations of Computational Anatomy, MFCA 2008, New York, USA (2008)
19. Cootes, T., Taylor, C., Cooper, D., Graham, J., et al.: Active shape models-their training and application. Computer Vision and Image Understanding 61(1), 38–59 (1995)
20. Audette, M.A., Ferrie, F.P., Peters, T.M.: An algorithmic overview of surface registration techniques for medical imaging. Medical Image Analysis 4, 201–217 (2000)
21. Bishop, C.: Pattern recognition and machine learning. Springer, Heidelberg (2006)
22. Schäfer, J., Strimmer, K.: A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. Statistical Applications in Genetics and Molecular Biology 4(1), 32 (2005)

Parameterizing Singularities of Positive Integral Index

Matthias Nieser and Konrad Polthier

Freie Universität Berlin
{matthias.nieser,konrad.polthier}@fu-berlin.de

Abstract. Classical surface parameterization algorithms often place singularities in order to enhance the quality of the resulting parameter map. Unfortunately, singularities of positive integral index (as the north pole of a sphere) were not handled since they cannot be described with piecewise linear parameter functions on a triangle mesh. Preprocessing is needed to adapt the mesh connectivity. We present an extension to the QuadCover parameterization algorithm [1], which allows to handle those singularities.

A singularity of positive integral index can be resolved using bilinear parameter functions on quadrilateral elements. This generalization of piecewise linear functions for quadrilaterals enriches the space of parameterizations. The resulting parameter map can be visualized by textures using a rendering system which supports quadrilateral elements, or it can be used for remeshing into a quad mesh.

1 Introduction

Classical algorithms for surface parameterization often use a global map which flattens a given surface and transfers it to a 2d parameter domain. During recent years, approaches have been given which allow the placement of singularities (or cone points). Singularities are necessary in order to minimize overall distortion of the parameterization.

For a given parameterization method, the amount of distortion is mainly determined by the location and type of singularities. Typical algorithms first place some singularities and hold them fixed during the subsequent optimization. An accurate singularity placement belongs to the main problems for parameterization.

The QuadCover algorithm also works in this manner. The singularities are taken from a given input frame field (e.g. from principle curvatures).

A special case arises for singularities of index $+1$. They cannot be resolved by a piecewise linear parameter function, since this would require its gradient to increase to infinity. All existing parameterization algorithms using PL triangle functions, face the same problem.

1.1 Previous Work

Surface parameterization is an active research area. We will shortly discuss early and recent work which are closely related. More complete lists can be found in [2,3].

Early global parameterization methods were introduced by Haker, Gu and Yau [4,5] and others. They studied conformal parameterizations which preserve angles at the cost of possibly large length distortions. Angles and lengths can not be preserved at the same time on general surfaces.

Other methods like Tong et al. [6] or Lai at al. [7] allow singular points which enlarge the space of harmonic functions used for parameterization. A good placement of singular points is an ongoing problem. [8] and [9] present two approaches for an automatic placement of singularities which are suited for conformal parameterizations.

With the QuadCover algorithm [1] we built upon an idea from Ray et al. [10], which use two orthogonal input fields as guiding directions for the parameter lines. The singularities are then taken from the guiding field. The idea of QuadCover is to find a parameterization whose gradient matches this field as well as possible.

1.2 Contributions

Typical parameterization algorithms use piecewise linear parameter functions on triangle meshes. Unfortunately, this function space is too rigid for describing vortex-like singularities. It is impossible to represent a vortex with piecewise linear texture maps on a triangle grid.

We propose a procedure for handling singularities of positive integral index, i.e. the indices are all integer numbers rather than fractional values. We therefore introduce bilinear texture maps on quadrilateral elements and use them for the construction of such singularities. The resulting parameterization can be visualized by rendering systems which support bilinear textures on quads. Independent from rendering, the parameterization can still be used for quad remeshing.

As in the classical QuadCover algorithm [1], we assume that the placement of singularities is given by the original input field. The construction of a good input field is still an unsolved problem.

The paper is structured as follows: The overview and the main ideas of QuadCover are outlined in Sect. 2. The algorithm intensively uses branched covering spaces. A short introduction into coverings is given in Sect. 3. Sect. 4 describes QuadCover in detail.

The main contribution of this paper is provided in Sect. 5. We show the relation between the index of a frame field singularity and a branch point of the covering. We introduce the problem which occurs with singularities of positive integral index. We then describe the extension of the QuadCover algorithm which solves the problem. Finally, Sect. 6 shows some results of the extended algorithm.

2 Overview

Given a smooth manifold M with an atlas of charts $\{U_i\}$. A global parameterization $f_i : U_i \to \mathbb{R}^2$ maps all charts into a flat (u,v)-domain. The parameter lines are the preimages of the unit grid $\mathbb{Z} \times \mathbb{R}$ (u_i lines), $\mathbb{R} \times \mathbb{Z}$ (v_i lines). They induce a quadrilateral structure in each chart.

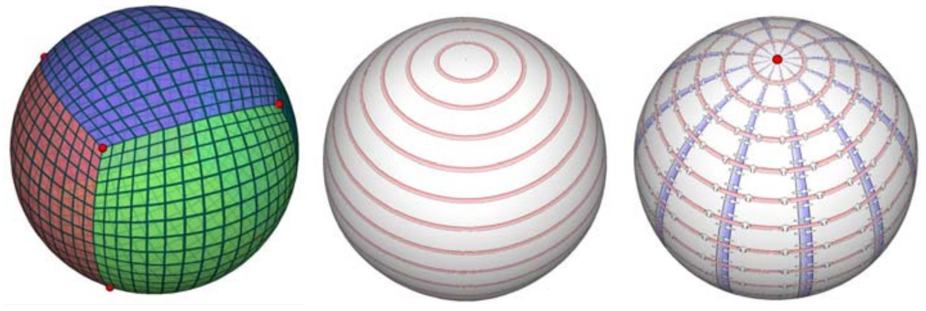


Fig. 1. Different parameterizations of a sphere. **Left:** Classical algorithms may produce 8 singularities which are equally distributed. **Middle:** Use a radial guidance frame field in QuadCover. Since index 1 singularities cannot be handled in the classic way, the resulting u component nearly vanishes everywhere. Only the v component behaves well. **Right:** The two singularities of index 1 are correctly resolved with the presented method.

Whenever charts overlap, the parameter functions are related by transition functions ξ_{ij}. In order to ensure, that the quad structure is globally continuous, we restrict all transition functions to leave the unit grid invariant. It is constrained to be of the form

$$\xi_{ij}(u,v) := f_j(f_i^{-1}(u,v)) = J^{r_{ij}} \begin{pmatrix} u \\ v \end{pmatrix} + d_{ij}, \quad r_{ij} \in \mathbb{Z}, d_{ij} \in \mathbb{Z}^2, \qquad (1)$$

where J is the rotation by 90 degrees in counter-clockwise direction. We call these maps *grid automorphisms*. The numbers r_{ij} decide, whether the u lines in chart U_i correspond to u, v, $-u$ or $-v$-lines in U_j. They are called *matchings* between the charts. The vectors d_{ij} encode a translational offset and are called *gaps* (see Fig. 2).

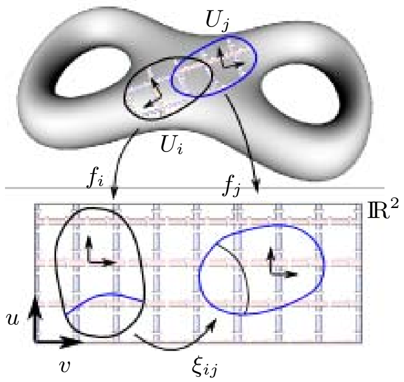


Fig. 2. Smooth manifold with two charts and matching $r_{ij} = 1$

Frame fields. The parameterization is guided by a so-called *frame field*. In each chart, it is just a collection of two vector fields. The goal is to find a parameter map, whose gradients matches up with the given frames as well as possible.

The gradients of the parameter map in different charts are related by:

$$\begin{pmatrix} \nabla u_j \\ \nabla v_j \end{pmatrix} = J^{r_{ij}} \begin{pmatrix} \nabla u_i \\ \nabla v_i \end{pmatrix} \tag{2}$$

Thus, the ordered list $(\nabla u, \nabla v, -\nabla u, -\nabla v)$ in U_j is the same as in U_i, but cyclically shifted $-r_{ij}$ times. In general, the gradients therefore do not describe global continuous vector fields on M. Whenever u- and v-gradients are flipped in different charts, the corresponding vectors must be identified. It turns out, that there is an elegant way to describe this setting using vector fields on covering spaces (see Sect. 3).

The algorithm. The QuadCover algorithm takes any frame field as input and generates a parameter function whose gradients matches up with the input field as well as possible. In practice, the principle curvature directions are often used as starting frame field, but the user can start with any field. Details about the algorithm are given in Sect. 4.

The resulting coordinates u and v can be used as texture map. Because of the coupling of u and v, the texture pattern has to be symmetric due to a rotation by 90 degrees. E.g. if one uses an image containing a quadrangular grid, then the surface gets divided into quadrangles.

Special issues arise, when there are singularities. When tracing a small cycle around a singularity p, the frame vectors turn by either a whole number of revolutions (singularity of integral index) or just by a multiple of 90 degrees (singularity of fractional index $k/4$, $k \in \mathbb{Z}$). The singularities with fractional index can be naturally described with branch points in a covering space as described in Sect. 4.

Singularities of index $k \in \mathbb{N}$ in general cannot be represented with standard piecewise linear functions. The nature of these singularities is that the parameter function in its vicinity tends to infinity (e.g. the gradients of the longitudinal part of the polar parameterization on a sphere). It turns out, that the space of PL functions is not flexible enough to represent such singularities. A key idea is to integrate additional quadrilateral elements and therefore extend the space of PL functions to functions which are linear on each triangle and bilinear on each quad. As for PL functions, this is a fully consistent space of continuous functions, even if triangles and quads are mixed in the same geometry. The idea and the algorithm will be presented in Sect. 5.

3 Frame Fields

This section gives a formal definition of frame fields and describes their relation to covering spaces.

Definition 1. *Given a manifold M with charts U_i and matchings r_{ij}. A* frame field *on M is a collection of two vector fields $K_{i,0}$, $K_{i,1}$ in each chart U_i. Let $K_{i,2} := -K_{i,0}$, $K_{i,3} := -K_{i,1}$. All overlapping charts $U_i \cap U_j \neq \emptyset$ must hold: $K_{j,m} = K_{i,(m-r_{ij})\ mod\ 4}$, $\forall m \in \{0, 1, 2, 3\}$.*

The algorithm uses the notion of branched covering surfaces for an equivalent description of frame fields as explained below.

3.1 Branched Covering Spaces

A frame field on the input surface can be seen as two vector fields on a covering surface. The advantage of this notion is, that standard vector field calculus can now be applied to frame fields.

Coverings. First, recall some definitions about Riemann surfaces, see [11], [12]. We first give an abstract definition of a covering and explain below how we actually construct one.

Definition 2. *Let M be a Riemann surface. A 4-sheeted covering M' of M is a Riemann surface with a local homeomorphism $\pi : M' \to M$ and the property: For each point $p \in M$, there exists a neighborhood U_p whose preimage $\pi^{-1}(U_p)$ is the union of exactly four pairwise disjoint topological disks. Fig. 3, left shows a 4-sheeted covering.*

In our setting, we allow some exceptional points p (branch points), where the preimage of a neighborhood of p is the union of less than four topological disks (e.g. as in Fig. 3, right)).

Construction. We construct a covering of M as follows: From each chart U_i, make four copies (*layers*) and name them $U'_{i,k}$, $k \in \{0, 1, 2, 3\}$. Let $\pi_i : \bigcup_k U'_{i,k} \to U_i$ be the operator which projects the copies back to U_i and $\tau_{i,k} : U_i \to U'_{i,k}$ the inverse maps. The four layers $U'_{i,k}$ together with π_i is called the *4-sheeted trivial covering* of the chart U_i (Fig. 3, left).

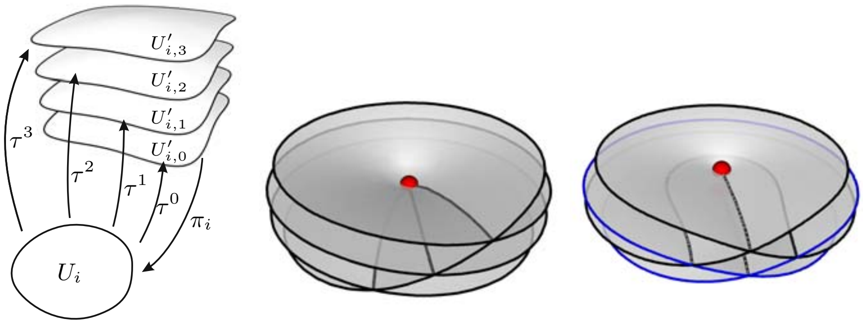


Fig. 3. Left: Trivial 4-sheeted covering. **Right:** Different branch points.

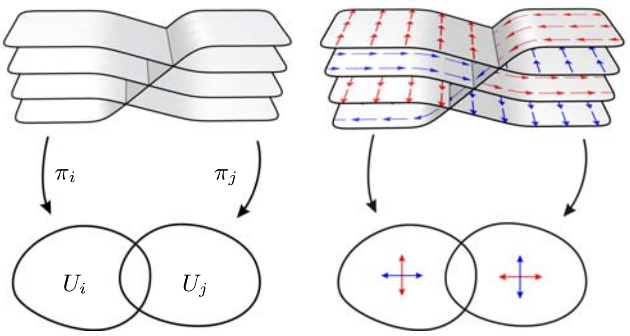


Fig. 4. Left: Patching two coverings together with matching $r_{ij} = 1$. **Right:** A frame field lifted to a vector field on the covering.

In the next step, we glue these layers at the overlaps of the adjacent charts together. For each pair of charts the layers may be glued in four different ways, which is defined by the corresponding matching r_{ij}.

Definition 3. *A* covering surface induced by given matchings r_{ij} *is uniquely defined by the following construction:*

Let (U_i', π_i) *be 4-sheeted trivial coverings of all charts* U_i*. The covering surface is given as the union of all* U_i' *where the following points are identified: For all overlapping charts* U_i*,* U_j*, identify the points* $\tau_i^k(p)$ *with* $\tau_j^{(k-r_{ij}) \text{ mod } 4}(p)$*,* $p \in U_i \cap U_j$*,* $k \in \{0, 1, 2, 3\}$ *(see Fig. 4, left).*

Since the trivial coverings of charts have no branch points and the charts cover the surface, we cannot get any branch point by this construction. Instead, they can be invented by removing single points from the surface. Depending on the matchings of charts in its vicinity, the covering could be extended to a branched covering there.

3.2 Vector Fields on Covering Spaces

In this section, we show how frame fields can be described as vector fields on a covering surface. It allows us to apply the classical theory for vector fields to frame fields.

A frame field $(K_{i,0}, K_{i,1})$ on M with matchings r_{ij} canonically lifts to two vector fields (K_0', K_1') on the covering which is induced by r_{ij}. In each chart, define the vectors on its trivial covering as follows: For all $p \in U_{i,k}'$ set $K_0'(p) := K_{i,k}(\pi_i(p))$ and $K_1'(p) := K_{i,(k+1)\text{mod } 4}(\pi_i(p))$, $k \in \{0, 1, 2, 3\}$.

The result are two globally well defined vector fields K_0', K_1' on M', since the layers of the covering are connected in the same way as the vector fields permute when another chart is chosen (compare Def. 1 and 3, Fig. 4, right).

Moreover, the vector fields are symmetric on the layers. I.e. if $K_0'(p)$ and $K_1'(p)$ are the vectors in a given point in layer 0, then the vectors in all other layers are: $(K_1'(p), -K_0'(p))$, $(-K_0'(p), -K_1'(p))$ and $(-K_1'(p), K_0'(p))$.

Discretization. Each triangle of the mesh is considered as a chart. The transition function between two adjacent triangles is fully determined by the matching and the translation vector associated with their common edge (see Eqn. (1)). There is no need to really compute the covering surface. It is implicitly given by the matchings.

Branch points are located at vertices. Let p be a vertex and all elements in its star are enumerated clockwise from 0 to $n-1$. The matchings at incident edges to p are given by $r_{i,(i+1) \bmod n}$. Let $t_p := ((\sum_i r_i) \bmod 4)$ be the *type* of the vertex. Branch points are characterized by $t_p \neq 0$. This means, starting somewhere in the neighborhood of p and walking around the vertex ends on a different layer in the covering. Fig. 3 shows branch points of type 1 and 2.

Discrete frame fields are piecewise constant. In each triangle, two vectors are stored. Together with the matching numbers, a discrete frame field is thereby uniquely described.

4 QuadCover Parameterization Algorithm

This section reviews the basic principles of the QuadCover algorithm. For further details, refer to [1].

Compute potential function. Given a surface M together with a frame field $(K_{i,0}, K_{i,1})$. Equivalently, given a covering surface M' with two vector fields (K'_0, K'_1). The parameterization $(u', v') : M' \to \mathbb{R}^2$ can be projected back to $(u, v) : M \to \mathbb{R}^2$ by taking the values in one of the layers. It does not matter which one, because the parameter lines in all layers will be congruent.

First pass. QuadCover uses a variational approach in order to find a parameterization which fits best possible to the given frame field. More precisely, the energy $E(u', v') = \int_{M'} \left(\|\nabla u' - K'_0\|^2 + \|\nabla v' - K'_1\|^2 \right) dA$ gets minimized. The functions u', v' live in a space of PL functions, which may be discontinuous at the edges (because of Eqn. (1)). The difference of function values at both sides of an edge must be a constant (called *gap* at an edge).

In practice, u' (and equivalently v') is found using a discrete Hodge-Helmholtz decomposition of the vector fields K'_0 (resp. K'_1). They can uniquely be written as $K'_k = P_k + C_k + H_k$, $k \in \{0, 1\}$ with a potential P_k, a copotential C_k and a harmonic vector field H_k. Any pair of functions u', v' satisfying $\nabla u' = P_0 + H_0$, $\nabla v' = P_1 + H_1$ minimizes the energy.

QuadCover integrates the vector fields $P_k + H_k$ in each chart (triangle) of the covering separately. The exact translation constant is left open at this stage. The resulting map (u', v') is in general not injective, the images of triangles may overlap. The common edge between adjacent triangles (of the covering) is by construction parallel and of same length in texture space.

Second pass. For getting a valid global parameterization, it remains to ensure that all gaps are $\in \mathbb{Z}^2$ (Eqn. (1)). QuadCover decreases the degrees of freedom by translation, such that all triangles are connected. Thus, all gaps become 0, except at few edges which form a cut graph of the mesh. A cut graph is a set of paths γ_i which cut the surface to a topological disk.

The second pass is based on the following observation: along each path γ_i of this cut graph, the gap is always a constant d_i. This is because the derivative of the function is locally integrable. Note, that there is an exception if two paths γ_i, γ_j merge and run on top of each other. In this case, the gap turns into $d_i + d_j$. For further details, see [1].

The algorithm first computes the gaps $d_i \in \mathbb{R}^2$ for all cut paths. Then, a special parameter function (h', k') is computed, which is harmonic and whose gaps at cut paths are exactly $[d_i] - d_i$. The maps h' and k' are uniquely described by this property up to global constant summand. The final parameterization is given as $(u' + h', v' + k')$ and satisfies all needed conditions.

5 Singularities

5.1 Singularities in QuadCover

A characteristic of each parameterization is the location and type of its singularities. The placement of singularities is important for low metric distortion.

Fractional index. For vector fields, the type of a singularity can be measured by its index. Take a closed path, which runs counterclockwise around a singularity p. The index $ind_p(X)$ of a vector field X at p is defined as the number of whole revolutions of the vectors along the path. The index is always an integer number.

When dealing with frame fields, the index is not constrained to be integer since tracking a vector along a closed path may not necessarily end up in the same frame vector. Therefore, the index can be multiples of $1/4$, (Fig. 5).

In QuadCover, we detect singularities from the input frame field (in most cases the principle curvature field). There is a difference about the handling of singularities with integer index and those with fractional index. Integer indices just appear as vector field singularities on the covering. Singularity of fractional index are resolved using a branch point. Remember from Sect. 3.1 that the type t_p of a branch point can be seen as the layer shift when walking around the branch point once. During the construction of the covering, a branch point of type $t_p = 4(i \bmod 1)$ is placed at each point, where the frame field has a non-integer singularity of index i. If you track one vector around the branch point, you end up on a different layer, i.e. the parameter lines exchange or flip the sign.

Figure 5 shows the branch points of different singularities. For index $1/4$ or $-1/4$, the 4 layers of the covering are connected at the branch point (of type 1 or 3) forming a cyclic spiral. For index $1/2$ or $-1/2$, two spirals (with two layers each) are formed (branch point of type 2).

Branch points increase the topologically complexity of the cover. They can be described by cutting an infinitesimally small hole into the surface. This enlarges the fundamental group of the surface and therefore enriches the wealth of parameter functions.

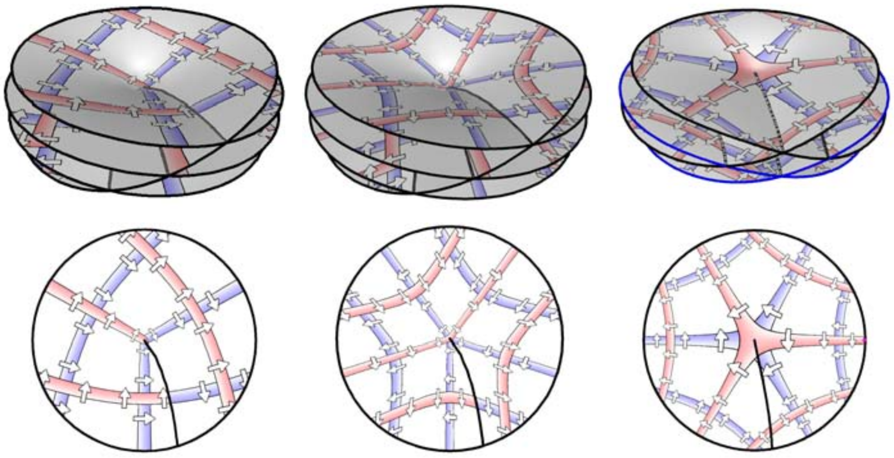


Fig. 5. Parameterizations with different singularities and their coverings. Left: Index = 1/4, Middle: Index = −1/4, Right: Index = −1/2.

The location of a branch point on the surface is fix during the whole optimization. The reason is that only the parameter function on the covering gets optimized, not the covering surface itself.

Integer index. Singularities of integer index do not have any affect on the topology of the covering. The north pole of a sphere for example (Fig. 1, right) does not lead to a branch point since walking around the pole ends up on the same layer. The same is true for a saddle of index -1.

The nature of singularities with integer index is completely different. Since they are not represented by the covering itself, they just arise as vector field singularities of the gradients. This may occur automatically during the optimization.

Positive integer index. There is a special case for singularities with positive integer index. Take a u function which has a local maximum at point p, thus ∇u has index 1 there. In a good parameterization, the gradients of u and v are approximately perpendicular to each other. In this case, ∇v would be a vortex around p.

Unfortunately, vortices cannot be described with a standard PL parameter map on triangles because of the following reason: If γ is a curve which runs around p, then the path integral $\int_\gamma \nabla v ds$ is a constant number, which stays the same as the curve gets contracted. Therefore, in the near vicinity of p, the gradients must tend to infinity (like an irrotational vortex). It is not possible to represent such a function as a piecewise linear function on a triangle mesh.

Hence, if the input frame field has a singularity of index 1, the parameterization algorithm ignores it and produces a parameter function which is far from the guiding field (see Fig. 1, middle). We now explain a method which handles this case and therefore provides better parameterizations.

5.2 Handle Singularities of Positive Integral Index

Use quads. A regular triangle mesh is too rigid for a piecewise linear vortex-like parameter function. Thus, we locally remesh the surface and invent some quadrilateral elements. The shape of the surface stays the same, but the quads allow a more complex structure of the parameter function.

Let p be a vertex, where the frame field has an index of $k \in \mathbb{N}$. Remesh the vertex star as displayed in Fig. 6. All triangles in the star become degenerated quads, whereas all newly inserted vertices are geometrically in the same location as p.

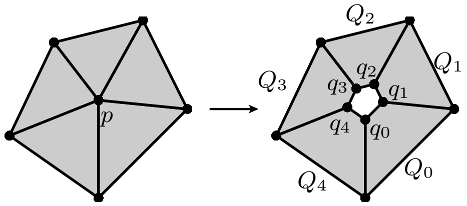


Fig. 6. Left: Vertex star of a singularity p. **Right:** Combinatoric of the remeshed vertex star. It consists of 5 quads Q_i. The inner vertices q_i are geometrically located in the same point p, thus the quads are degenerated.

Scalar functions on a mesh with quads are not longer forced to be piecewise linear. We extend the space to functions, which are linear on each triangle and bilinear on each quadrangle. Given function values at the vertices of a quad, the function itself is then given by the unique bilinear function, which interpolates these values. The resulting function is continuous, even if triangles and quads are mixed in the same mesh.

Parameter functions on meshes with triangles and quads can be used as texture map. If the rendering system supports bilinear textures, one can easily display singularities of integral index, see Fig. 7.

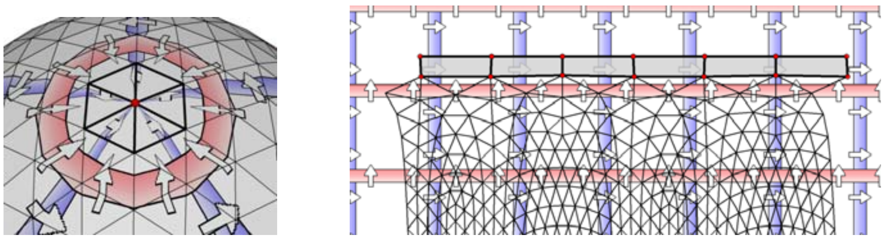


Fig. 7. Left: Bilinear texture on a coarse mesh with triangles and quads. **Right:** Image of the elements in texture space. The quads in the vertex star of the singularity are marked in grey.

Using quadrilaterals, one can now map a polygon in texture space to a single point on the surface. All the radial parameter lines which cross one of the quadrangles in Fig. 7 run into the singularity.

Approximation. The original QuadCover algorithm works on triangles. It would be straight forward to generalize it to quadrangles. The only difference is that the function space for energy minimization changes. Although, an adaption of the optimization algorithm to work with bilinear functions turned out to be complicated, since the computation of the derivatives of the energy requires to solve a non-linear integral. Instead, we simplified the problem and approximated the optimal solution. This approximation replaces the quads by triangles again, but with an altered connectivity. Thus the standard QuadCover algorithm can be used. The quads are then used afterwards for the final description and visualization of the result. The outline of the extended QuadCover is listed in Algorithm 1.

Algorithm 1. Modified QuadCover algorithm. Input: Guidance frame field

```
1: for all vertices $p$ do
2:     Measure index of frame field at $p$
3:     if (index mod 1 == 0) and (index > 0) then
4:         Store vertex $p$ in array specialSingularities
5:         Cut all outgoing edges from $p$ open.
6:     end if
7: end for
8: Run original QuadCover algorithm
9: for all $p$ in specialSingularities do
10:     Replace all adjacent triangles to $p$ by a quadrilateral
11:     Compute texture coordinates for the quads
12: end for
```

Lines 1–7 do a local remeshing at each vertex p with positive integer index. All adjacent edges to p are cut open generating a hole in the surface, see Fig. 8. Then, the QuadCover algorithm is applied to the triangle surface. Fig. 9 shows the texture domain of the example from Fig. 7.

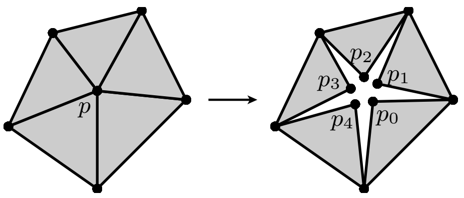


Fig. 8. Each vertex star of a singularity p with positive integer index will be cut open. The right mesh shows the new combinatoric, the inner vertices p_i are geometrically located at p.

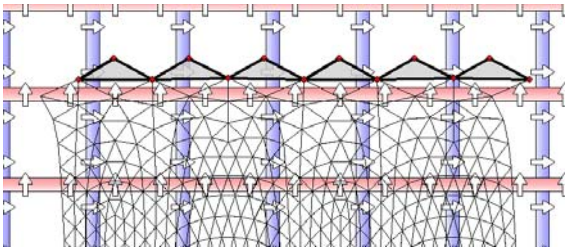


Fig. 9. Texture domain after parameterization with modified mesh connectivity. The surface from Fig. 7 left is taken as input.

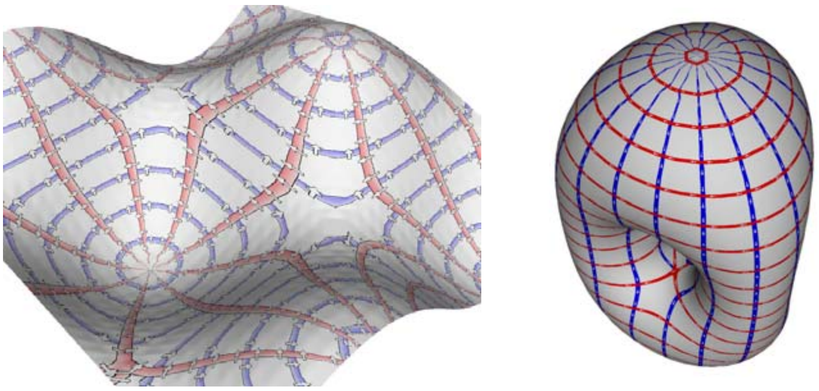


Fig. 10. Left: The parameterization is guided by the gradient and cogradient of the height function. Singularities of index 1 naturally appear at all local minima and maxima. They are correctly detected and resolved by the method. The singularities of negative index at saddle points occur automatically, even if there is no parameter line running directly into the saddle. **Right:** Parameterization on a 3-fold Lawson surface. The height function was used for this parameterization, too.

In lines 9–12, the singularities get remeshed again. Each triangle of the vertex star (which was previously cut open) is now replaced by a quadrangle. All quads are connected as in the situation of Fig. 6, right.

It remains to compute the texture coordinates for the created vertices q_i. They are obtained by just averaging the texture coordinates of the old vertices p_i (from Fig. 8, right). In quadrangle Q_j, compute the texture coordinates as:

$$f_j(q_j) := 1/2\,(f_{j-1}(p_{i-1}) + d_{i-1,i} + f_j(p_j)) \tag{3}$$
$$f_j(q_{j+1}) := 1/2\,(f_j(p_j) + f_{j+1}(p_{i+1}) + d_{i+1,i})$$

where $d_{i-1,i}$ is the translational part of the transition between chart Q_{i-1} and Q_i, see Eqn. (1).

6 Results

With this extension, QuadCover produces very stable parameterizations, even when singularities of integral index are present. All what we need is a frame field, which contains singularities at reasonable locations.

A good placement of singularities is still an open problem. The singularities from principle curvature fields are mostly nice, but their location is not very stable in nearly umbilic areas. Particularly, singularities of index 1 will mostly split up into 4 singularities of index 1/4 each.

We tested the algorithm on some user generated frame fields. Fig. 10, left shows the graph of the function $f(u, v) = \sin(u) \cos(v)$. A frame field was produced using the gradient field of the height function and its 90 degrees rotated field. The parameterization has two singularities of index 1 and two saddles of index -1.

Fig. 10, right shows a Lawson surface. It is a constant mean curvature in $\mathbb{R}^3$ and is made out of 30k triangles. The parameterization took several seconds.

References

1. Kälberer, F., Nieser, M., Polthier, K.: Quadcover - surface parameterization using branched coverings. Comp. Graph. Forum 26 (2007)
2. Hormann, K., Polthier, K., Sheffer, A.: Mesh parameterization: Theory and practice. In: SIGGRAPH Asia, Course Notes (2008)
3. Floater, M., Hormann, K.: Surface parameterization: a tutorial and survey (2005)
4. Haker, S., Angenent, S., Tannenbaum, A., Kikinis, R., Sapiro, G., Halle, M.: Conformal surface parameterization for texture mapping. IEEE Trans. on Vis. and Comp. Graph. 6 (2000)
5. Gu, X., Yau, S.T.: Global conformal surface parameterization. In: Symp. on Geom. Proc., pp. 127–137 (2003)
6. Tong, Y., Alliez, P., Cohen-Steiner, D., Desbrun, M.: Designing quadrangulations with discrete harmonic forms. In: Proc. Eurographics (2006)
7. Lai, Y.K., Jin, M., Xie, X., He, Y., Palacios, J., Zhang, E., Hu, S.M., Gu, X.D.: Metric-driven rosy fields design. Technical report, Tsinghua Univ., Beijing, China (2008)
8. Ben-Chen, M., Gotsman, C., Bunin, G.: Conformal flattening by curvature prescription and metric scaling. Computer Graphics Forum (2008)
9. Springborn, B., Schröder, P., Pinkall, U.: Conformal equivalence of triangle meshes. In: Siggraph, Proceedings (2008)
10. Ray, N., Li, W.C., Lévy, B., Sheffer, A., Alliez, P.: Periodic global parameterization. ACM Trans. Graph. 25(4) (2006)
11. Farkas, H.M., Kra, I.: Riemann Surfaces. Springer, Heidelberg (1980)
12. Jost, J.: Compact Riemann Surfaces. Springer, Heidelberg (2002)

Two Step Time Discretization of Willmore Flow

N. Olischläger and M. Rumpf

Institut für Numerische Simulation,
Rheinische Friedrich-Wilhelms-Universität Bonn, Germany
{nadine.olischlaeger,martin.rumpf}@ins.uni-bonn.de

Abstract. Based on a natural approach for the time discretization of gradient flows a new time discretization for discrete Willmore flow of polygonal curves and triangulated surfaces is proposed. The approach is variational and takes into account an approximation of the L^2-distance between the surface at the current time step and the unknown surface at the new time step as well as a fully implicity approximation of the Willmore functional at the new time step. To evaluate the Willmore energy on the unknown surface of the next time step, we first ask for the solution of a inner, secondary variational problem describing a time step of mean curvature motion. The time discrete velocity deduced from the solution of the latter problem is regarded as an approximation of the mean curvature vector and enters the approximation of the actual Willmore functional. To solve the resulting nested variational problem in each time step numerically relaxation theory from PDE constraint optimization are taken into account. The approach is applied to polygonal curves and triangular surfaces and is independent of the co-dimension. Various numerical examples underline the stability of the new scheme, which enables time steps of the order of the spatial grid size.

1 Introduction

In this paper a new scheme for the time and space discretization of parametric Willmore flow is presented. Willmore flow is the L^2 gradient flow of surfaces for the Willmore energy, which measures the squared mean curvature on the surface. Let $\mathcal{M}$ be a closed d-dimensional surface embedded in $\mathbb{R}^m$ with $m \geq d+1$ and denote by x the identity map on $\mathcal{M} = \mathcal{M}[x]$. Then the Willmore energy is defined as

$$w[x] := \frac{1}{2}\int_{\mathcal{M}} \mathbf{h}^2 \,\mathrm{d}a$$

where $\mathbf{h}$ is the mean curvature on $\mathcal{M}$, i. e., $\mathbf{h}$ is the sum of the principle curvatures on $\mathcal{M}$. Furthermore, the L^2-metric $\int_{\mathcal{M}} v_1 v_2 \,\mathrm{d}a$ measures variations $x + v_i n$ of the surface $\mathcal{M}$ in direction of the surface normal n. Given energy and metric the corresponding gradient flow – the Willmore flow – in the hypersurface case (cf. Figure 1) ($m = d+1$) is given by the following fourth order parabolic evolution problem

$$\partial_t x(t) = \Delta_{\mathcal{M}(t)}\mathbf{h}(t)\, n(t) + \mathbf{h}(t)\,(|S(t)|_2^2 - \frac{1}{2}\mathbf{h}(t)^2)\, n(t)\,,$$

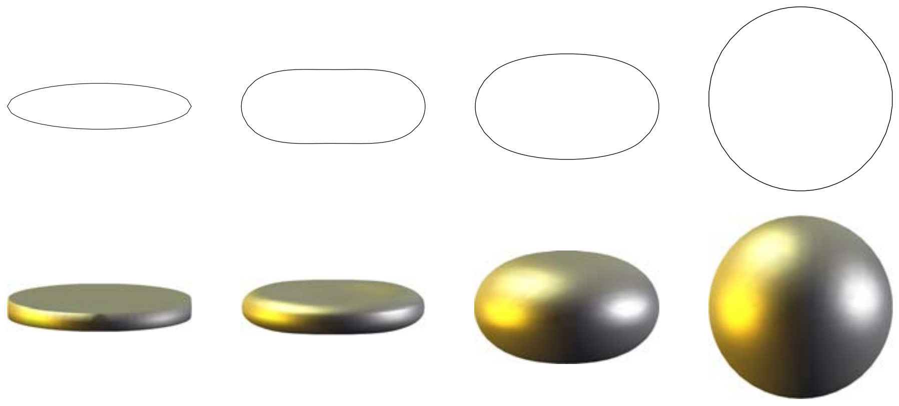


Fig. 1. Different time steps of the Willmore flow of an original ellipsoid curve with 100 vertices is shown (top row). The time step size was chosen of the order of the spatial grid size $h = \tau = 0.0632847$. Willmore flow of a deformed sphere towards a round sphere is depicted in the bottom row. We show the surface at times $t = 0$, $t = \tau$, $t = 50\tau$, and $t = 150\tau$, where $\tau = h = 0.02325548045$.

which defines for a given initial surface $\mathcal{M}_0$ a family of surfaces $\mathcal{M}(t)$ for $t \geq 0$ with $\mathcal{M}(0) = \mathcal{M}_0$ problem [1,2,3]. Here, $\Delta_{\mathcal{M}(t)}$ is the Laplace Beltrami operator on $\mathcal{M}(t)$, $S(t)$ denotes the shape operator on $\mathcal{M}(t)$, $n(t)$ the normal field on $\mathcal{M}(t)$, and $|\cdot|_2$ the Frobenius norm on the space of endomorphisms on the tangent bundle $\mathcal{TM}(t)$. The Willmore functional is used e.g. for the modeling of elastic surfaces [4,5,6,7]. The analytic treatment of the Willmore flow has been considered by Polden [8,9] and sharp results on long time existence and regularity were obtained by by Kuwert and Schätzle [3,10,11]. Willmore flow for curves is called elastic flow of curves (cf. Figure 1) and has been considered by Dziuk, Kuwert and Schätzle in [12]. A level set formulation has been developed by [13]. We refer to Deckelnick and Dziuk [14] for the convergence analysis in the graph case and to Barrett, Garcke and Nürnberg [15], Bobenko and Schröder [16] and Dziuk [17] for alternative numerical methods for Willmore flow on triangular surfaces. An early variational approach for surface modeling is described in [18]. Nowadays fourth order problem are very popular in the context of image inpainting and surface restoration [19,20,21,22,23]. Apart from fully explicit time discretizations these numerical approach are characterized by a semi-implicit time discretization, which requires the solution of a linear system of equations in each time step. One observes significant restrictions on the time step size. Effectively, one usually has to enforce time steps $\tau = O(h^2)$, where h was the spatial grid size.

This shortcoming motivated the development of a new concept for the time discretization of Willmore flow picking up the variational time discretization of general gradient flows. Given an energy $e[\cdot]$ on a manifold the gradient flow $\dot{x} = -\mathrm{grad}_g e[x]$ with initial data x^0 one defines a sequence of time discrete solutions $(x^k)_{k=0,\ldots}$, where $x_k \approx x(k\tau)$ for the time step size τ via a variational problem, to be solved in each time step, i.e.

$$x^{k+1} = \arg\min_x \operatorname{dist}(x, x^k)^2 + 2\tau\, e[x]$$

where $\operatorname{dist}(x, x^k) = \inf_{\gamma \in \Gamma} \int_0^1 \sqrt{g_{\gamma(s)}(\dot\gamma(s), \dot\gamma(s))}\,\mathrm{d}s$ is the shortest path length on the manifold, given the metric $g(\cdot,\cdot)$. Here Γ denotes the set of smooth curves γ with $\gamma(0) = x^k$ and $\gamma(1) = x$. As an immediate consequence, one obtains the energy estimate $e[x^{k+1}] + \frac{1}{2\tau}\operatorname{dist}(x^{k+1}, x^k)^2 \leq 0 + e[x^k]$. For geometric problems, this approach has already been considered by Luckhaus and Sturzenhecker [24] in case of mean curvature motion, which is the L^2 gradient flow of the surface area. They proposed a corresponding fully implicit time discretization based on a variational problem in BV to be solved in each time step. In fact, in each time step the symmetric distance between two consecutive shapes corresponding to the current and the next time step is balanced by the time step τ times the perimeter of the shape at the next time step. Chambolle [25] investigated a reformulation of this approach in terms of a level set method. A related method for anisotropic mean curvature motion is discussed in [26,27].

In case of Willmore flow, we will proceed as follows. We aim at balancing the squared distance of the unknown surface at time $t_{k+1} = t_k + \tau$ from the current surface at time t_k and a suitable approximation of the Willmore energy at time t_{k+1} scaled by twice the time step size. Solving a fully implicit time discrete problem for mean curvature motion for the unknown surface at time t_{k+1}, we can regard the corresponding difference quotient in time as a time discrete, fully implicit approximation of the mean curvature vector. Based on this mean curvature vector, the Willmore functional can be approximated. Thus, we are led to a nested minimization problem in each time step. In the inner problem on the new time step an implicit mean curvature vector is identified. Then, the outer problem is the actual implicit, variational formulation of Willmore flow. Indeed, the resulting two step time discretization experimentally turns out to be unconditionally stable and effectively allows for time steps of the order of the spatial grid size.

The paper is organized as follows. In Section 2 we derive the time discrete scheme for Willmore flow, still continuous in space. Based on piecewise affine finite elements on simplicial surfaces we derive a fully discrete numerical approach in Section 3. In Section 4 the duality technique from PDE constraint optimization is revisited to derive a minimization algorithm for the optimization problem. Finally, in Section 5 various examples for the Willmore of curves and surfaces are investigated.

2 Derivation of the Two Step Time Discretization

Before we consider the actual time discretization of Willmore flow, let us briefly review the time discretization of mean curvature motion. Following the above abstract approach the variational time discretization of mean curvature motion for a given surface $\mathcal{M} = \mathcal{M}[x]$ defines the mapping $y = y[x]$ of the next time step surface $\mathcal{M}[y]$ as the minimizer of the functional $\mathrm{dist}(\mathcal{M}[y], \mathcal{M}[x])^2 + 2\tilde{\tau} \int_{\mathcal{M}[y]} \mathrm{d}a$, where $\tilde{\tau}$ is the considered time step, $\mathrm{dist}(\cdot,\cdot)$ is the L^2 distance between surfaces, and $\int_{\mathcal{M}[y]} \mathrm{d}a$ the surface area of $\mathcal{M}[y]$ as the underlying energy. Now, for y close to x, we can consider a first order expansion in time and obtain the following variational problem: Given a surface $\mathcal{M}[x]$ parameterized by a mapping x we ask for a mapping $y = y[x]$, which minimizes the functional

$$e[x,y] = \int_{\mathcal{M}[x]} (y-x)^2 + \tilde{\tau} |\nabla_{\mathcal{M}[x]} y|^2 \,\mathrm{d}a$$

for given x. In what follows the time step size $\tilde{\tau}$ is chosen independent of the time step size for the actual time discrete Willmore flow. In our later spatially discrete model we consider a $\tilde{\tau}$ equal to the square of the spatial grid size. The resulting weak form of the corresponding Euler-Lagrange equations is

$$0 = \int_{\mathcal{M}[x]} (y-x)\cdot\theta + \tilde{\tau} \nabla_{\mathcal{M}[x]} y : \nabla_{\mathcal{M}[x]} \theta \,\mathrm{d}a$$

for some test function θ, where $A : B = \mathrm{tr}(A^T B)$. This equation coincides with the nowadays classical scheme for a single semi–implicit time step of mean curvature motion already proposed by Dziuk [28].

Now, we deduce from the time continuous evolution equation $\partial_t x = \mathbf{h} n$ that the difference quotient $\frac{y[x]-x}{\tilde{\tau}}$ can be considered as a regularized approximation of the mean curvature vector $\mathbf{h} n$ on $\mathcal{M}[x]$. Thus, the functional $\frac{1}{2}\int_{\mathcal{M}[x]} \frac{(y[x]-x)^2}{\tilde{\tau}^2} \,\mathrm{d}a$ approximates the Willmore functional on $\mathcal{M}[x]$.

This enables us to define a time discretization of Willmore flow, which does not require the explicit evaluation of the mean curvature on the unknown surface of the next time step. Indeed, in the abstract variational problem

$$\mathrm{dist}(\mathcal{M}[x], \mathcal{M}[x^k])^2 + \tau \int_{\mathcal{M}[x]} \mathbf{h}^2 \,\mathrm{d}a \to \min$$

we consider the same linearization of the L^2 distance as for mean curvature motion and use the above approximation of the Willmore energy. Finally, we obtain the following scheme:

Given an initial surface $\mathcal{M}[x^0]$ we define a sequence of surfaces $\mathcal{M}[x^k]$ with $k = 1, \cdots$, where x^{k+1} minimizes the functional

$$w[x^k, x, y[x]] = \int_{\mathcal{M}[x^k]} (x - x^k)^2 \,\mathrm{d}a + \frac{\tau}{\tilde{\tau}^2} \int_{\mathcal{M}[x]} (y[x]-x)^2 \,\mathrm{d}a$$

for given x_k. Hence, x^k is assumed to approximate $x(t_k)$ with $t_k = k\tau$ for the given time step τ.

Thus, in each time step we have to solve the nested variational problem

$$\begin{aligned} x^{k+1} &= \arg\min_x w[x^k, x, y[x]] \quad \text{with} \\ y[x] &= \arg\min_y e[x,y]\,. \end{aligned} \tag{1}$$

The inner problem is quadratic, hence the Euler–Lagrange equation is a linear elliptic PDE and we end up with a PDE constrained optimization problem for each time step.

To be more explicit, let us examine circles in the plane. Under Willmore flow circles expand according the ODE $\dot{R}(t) = \frac{1}{2}R(t)^{-3}$ for the radius. In comparison to this the radius R^{k+1} in the above time discrete scheme turns out to be a solution of the nonlinear equation $\frac{R-R_k}{\tau} = \frac{1}{2}\frac{R^4-3R^2\tau}{(R^2+\tau)^3\, R_k}$, which is an implicit first order scheme for the above ODE (cf. Figure 3).

3 Finite Element Space Discretization

In this section we introduce a suitable space discretization based on piecewise affine finite elements. Here, we follow the guideline for finite elements on surfaces introduced by [29]. Thus, we consider simplicial meshes $\mathcal{M}[X]$ - polygonal curves for $d = 1$ and triangular surfaces for $d = 2$ - as approximations of the d dimensional surfaces $\mathcal{M}[x]$. Here, X is the identity on the simplicial mesh $\mathcal{M}[X]$ which is described by a vector $\bar{X}$ of vertex positions of the mesh. To clarify the notation we will always denote discrete quantities with upper case letters to distinguish them from the corresponding continuous quantities in lower case letters. Furthermore, a bar on top of a discrete function indicates the corresponding nodal vector, i.e. $\bar{X} = (\bar{X}_i)_{i\in I}$, where $\bar{X}_i = (X_i^1, \cdots, X_i^m)$ is the coordinate vector of the ith vertex of the mesh and I denotes the index set of vertices.

Hence, given some initial surface $\mathcal{M}[X^0]$ we seek a sequence of discrete surfaces $(\mathcal{M}[X^k])_{k=1,\cdots}$ of discrete surfaces. Locally, using also local indices each element T of a polygonal curve is a line segment with nodes X_1 and X_2 and elements T of a triangulation are planar triangles with vertices X_0, X_1, and X_2 and face vectors $F_0 = X_2 - X_1$, $F_1 = X_0 - X_2$, and $F_2 = X_1 - X_0$. Given a simplicial surface $\mathcal{M}[X]$ we denote by

$$\mathcal{V}(\mathcal{M}[X]) := \left\{U \in C^0(\mathcal{M}[X]) \,|\, \Phi|_T \in \mathcal{P}_1 \,\forall T \in \mathcal{M}[X]\right\}.$$

the corresponding piecewise affine Finite Element space consisting of those functions being affine on each element T of $\mathcal{M}[X]$. With a slight misuse of notation the mapping X itself is considered as an element in $\mathcal{V}(\mathcal{M}[X])^m$. Let $\{\Phi_i\}_{i\in I}$ be the nodal basis of $\mathcal{V}(\mathcal{M}[X])$. Thus, for $U \in \mathcal{V}(\mathcal{M}[X])$ we obtain $U = \sum_{i\in I} U(X_i)\Phi_i$ and $\bar{U} = (U(X_i))_{i\in I}$, in particular in accordance to our above definition we recover $\bar{X} = (X_i)_{i\in I}$.

Next, let us introduce the mass matrix $M[X]$ and the stiffness matrix $L[X]$ on the discrete surface $\mathcal{M}[X]$, whose entries are given by

$$M_{ij}[X] = \int_{\mathcal{M}[X]} \Phi_i \Phi_j \,\mathrm{d}a, \qquad L_{ij}[X] = \int_{\mathcal{M}[X]} \nabla_{\mathcal{M}[X]}\Phi_i \cdot \nabla_{\mathcal{M}[X]}\Phi_j \,\mathrm{d}a\,.$$

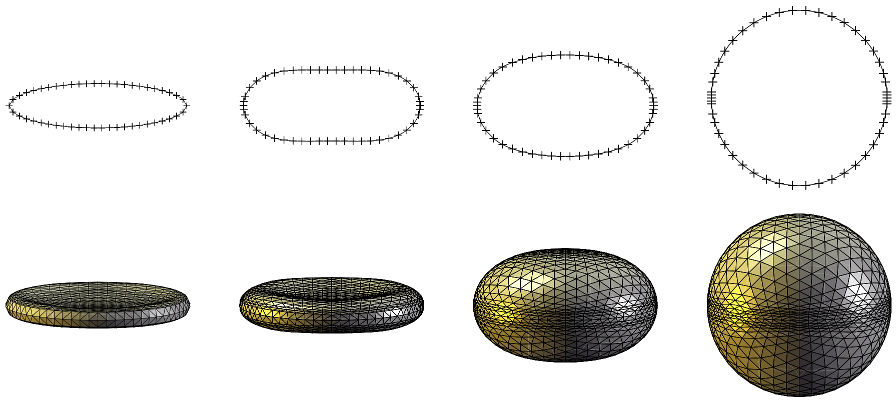


Fig. 2. The grids of the evolution under Willmore flow of the initial ellipsoid curve and deformed sphere of Figure 1 are shown at the same times. We did not reparametrize the curve since our scheme does not suffer from undesired tangential motions.

To apply mass and stiffness matrices to discrete maps from $\mathcal{M}[X]$ to $\mathbb{R}^m$, we need corresponding block matrices $\mathbf{M}[X]$ and $\mathbf{L}[X]$ in $\mathbb{R}^{m\sharp I\times m\sharp I}$:

$$\mathbf{M}[X] = \begin{pmatrix} M[X] & & \\ & M[X] & \\ & & M[X] \end{pmatrix}, \qquad \mathbf{L}[X] = \begin{pmatrix} L[X] & & \\ & L[X] & \\ & & L[X] \end{pmatrix}.$$

Both the mass and the stiffness matrices M and L can be assembled from corresponding local mass and stiffness matrices $m(T)$ and $l(T)$ for all simplices T on $\mathcal{M}[X]$.

Now, we have all the ingredients at hand to derive the fully discrete two step time discretization of Willmore flow (cf. Figure 2), which can be regarded as the discrete counterpart of (1). Given a discrete surface $\mathcal{M}(X^k)$ in time step k we define $X^{k+1} \in \mathcal{V}(\mathcal{M}[X^k])^m$ as the minimizer of the following spatially discrete, nested variational problem

$$\begin{aligned} X^{k+1} &= \arg\min\nolimits_{X\in\mathcal{V}(\mathcal{M}[X^k])^m} W[X^k, X, Y[X]] \quad \text{with} \\ Y[X] &= \arg\min\nolimits_{Y\in\mathcal{V}(\mathcal{M}[X])^m} E[X,Y]\,, \end{aligned} \tag{2}$$

where

$$\begin{aligned} E[X,Y] &:= \int_{\mathcal{M}[X]} (Y-X)^2 + \tilde{\tau}|\nabla_{\mathcal{M}[X]} Y|^2 \,\mathrm{d}a \\ &= \mathbf{M}[X](\bar{Y}-\bar{X})\cdot(\bar{Y}-\bar{X}) + \tilde{\tau}\mathbf{L}[X]\bar{Y}\cdot\bar{Y}\,, \\ W[X^k,X,Y] &:= \int_{\mathcal{M}[X^k]} (X-X^k)^2 \,\mathrm{d}a + \frac{\tau}{\tilde{\tau}^2}\int_{\mathcal{M}[X]} (Y-X)^2 \,\mathrm{d}a \\ &= \mathbf{M}[X^k](\bar{X}-\bar{X}^k)\cdot(\bar{X}-\bar{X}^k) + \frac{\tau}{\tilde{\tau}^2}\mathbf{M}[X](\bar{Y}-\bar{X})\cdot(\bar{Y}-\bar{X}) \end{aligned}$$

are the straightforward spatially discrete counterpart of the functionals $e[x, y]$ and $w[x^k, x, y]$, respectively. In analogy to the continuous case for given X the nodal vector $\bar{Y}[X]$ solves the linear system of equations

$$(\mathbf{M}[X] + \tilde{\tau}\mathbf{L}[X])\,\bar{Y}[X] = \mathbf{M}[X]\,\bar{X}\,. \tag{3}$$

For the sake of completeness let us finally give explicit formulas for the entries of the mass and stiffness matrices. Later in Section 4 we will have to compute variations of these entries as well.

Polygonal curve. In the case of curves we consider a lumped mass matrix (cf. [30]) and obtain directly for the global matrices

$$M[X] = \operatorname{diag}\left(\frac{1}{2}(Q_i + Q_{i+1})\right),\; L[X] = \operatorname{tridiag}\left(-\frac{1}{Q_i}, \frac{1}{Q_i} + \frac{1}{Q_{i+1}}, -\frac{1}{Q_{i+1}}\right)$$

where $Q_i = |X_i - X_{i-1}|$ is the length of the jth line segment and diag() and tridiag() denote diagonal or tridiagonal matrices with the corresponding entries in each row. Here, we assume a cyclic indexing, i.e. we identify the indices $i = 1$ and $i = \sharp I + 1$ for closed curves with $X_0 = X_{\sharp I}$.

Triangular surfaces. Due to the greater variability of triangular surfaces compared to polygonal curves, let us consider the local matrices on triangles separately. Denoting the local basis function on a triangle T by Φ_0, Φ_1, Φ_3, where $\Phi_i(X_j) = \delta_{ij}$ (with δ_{ij} being the usual Kronecker symbol) we verify by a simple straightforward computation (cf. [31]) that

$$m(T) = \left(\int_T \Phi_i \Phi_j \,\mathrm{d}a\right)_{i,j=0,1,2} = \frac{|T|}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix},$$

with $|T| = \frac{1}{2}|F_2 \wedge F_1|$ being the area of the triangle T, and

$$l(T)_{ij} = \int_T \nabla_T \Phi_i \cdot \nabla_T \Phi_j \,\mathrm{d}a = \frac{F_i \cdot F_j}{4|T|}\,,$$

where ∇_T the gradient on planar T.

4 Numerical Solution of the Optimization Problem

In this section, we discuss how to solve the nonlinear optimization problem (2) in each time step numerically. Here, we will confine ourselves to a gradient descent approach and take into account a suitable duality technique to effectively compute the gradient of the energy functional $\hat{W}[X] = W[X^k, X, Y[X]]$ given the fact that the argument $Y[X]$ is a solution of the inner minimization problem and as such solves the linear system of equations (3). Indeed, we obtain for the variation of $\hat{W}$ in a direction $\Theta \in \mathcal{V}(\mathcal{M}[X^k])^m$

$$\partial_X \hat{W}[X](\Theta) = \partial_X W[X^k, X, Y[X]](\Theta) + \partial_Y W[X^k, X, Y[X]]\ (\partial_X Y[X](\Theta))\,.$$

A direct computation of $\partial_X Y[X](\Theta)$ would require the solution of the inner minimization problem and thus specifically a linear system (cf. (3)) would have to be solved for every test function Θ. This can be avoided applying the following duality argument:

From the optimality of $Y[X]$ in the inner problem, we deduce the equation $0 = \partial_Y E[Y[X], X](\Psi)$ for any test function $\Psi \in \mathcal{V}(\mathcal{M}[X])^m$. Now, differentiating with respect to X we obtain

$$\begin{aligned} 0 &= \partial_X \left(\partial_Y E[Y[X], X](\Psi)\right)(\Theta) \\ &= \partial_X \partial_Y E[Y, X](\Psi, \Theta) + \partial_Y^2 E[Y[X], X](\Psi, \partial_X Y[X](\Theta)) \end{aligned}$$

for any test function Ψ. Let us now define $P \in \mathcal{V}(\mathcal{M}[X^k])^m$ as the solution of the dual problem

$$\partial_Y^2 E[Y[X], X](P, \Psi) = \partial_Y W[X^k, X, Y[X]](\Psi)\,. \tag{4}$$

for all test functions $\Psi \in \mathcal{V}(\mathcal{M}[X^k])^m$. Now, choosing $\Psi = \partial_X Y[X](\Theta)$ one obtains

$$(\partial_Y W)\,[X^k, X, Y[X]]\ (\partial_X Y[X](\Theta)) = -\partial_X \partial_Y E[Y, X](P, \Theta)\,.$$

Thus, we can finally rewrite the variation of $\hat{W}$ with respect to X in a direction Θ as

$$\partial_X \hat{W}[X](\Theta) = \partial_X W[X^k, X, Y[X]](\Theta) - \partial_X \partial_Y E[Y, X](P, \Theta). \tag{5}$$

The solution P of the dual problem (4) requires the solution of

$$\int_{\mathcal{M}[X]} P \cdot \Psi + \tilde{\tau} \nabla_{\mathcal{M}[X]} P : \nabla_{\mathcal{M}[X]} \Psi \, \mathrm{d}a = \int_{\mathcal{M}[X]} \frac{\tau}{\tilde{\tau}^2}\,(Y - X) \cdot \Psi \, \mathrm{d}a$$

for all test functions Ψ. In matrix vector notation, this can be written as the linear system of equations

$$(\mathbf{M}[X] + \tilde{\tau} \mathbf{L}[X])\,\bar{P} = \frac{\tau}{\tilde{\tau}^2} \mathbf{M}[X](\bar{Y} - \bar{X})\,.$$

The terms on the right hand side of (5) are to be evaluated as follows

$$\begin{aligned} (\partial_X W)\,[X^k, X, Y](\Theta) &= 2\,\mathbf{M}[X^k](\bar{X} - \bar{X}^k) \cdot \bar{\Theta} + 2 \frac{\tau}{\tilde{\tau}^2} \mathbf{M}[X](\bar{X} - \bar{Y}) \cdot \bar{\Theta} \\ &\quad + \frac{\tau}{\tilde{\tau}^2}\,(\partial_X \mathbf{M}[X](\Theta))(\bar{Y} - \bar{X}) \cdot (\bar{Y} - \bar{X})\,, \\ \partial_X \partial_Y E[Y, X](P, \Theta) &= \partial_X \left(2\mathbf{M}[X](\bar{Y} - \bar{X}) \cdot \bar{P} + 2\tilde{\tau} \mathbf{L}[X]\bar{Y} \cdot \bar{P}\right)(\Theta) \\ &= 2(\partial_X \mathbf{M}[X](\Theta))(\bar{Y} - \bar{X}) \cdot \bar{P} - 2\mathbf{M}[X]\bar{\Theta} \cdot \bar{P} \\ &\quad + 2\tilde{\tau}(\partial_X \mathbf{L}[X](\Theta))\bar{Y} \cdot \bar{P}\,. \end{aligned}$$

The computation remains of the variation of the mass and stiffness matrix with respect to a variation θ of the simplicial grid,

$$\partial_X \mathbf{M}[X](\Theta) = \begin{pmatrix} \partial_X M[X](\Theta) & & \\ & \partial_X M[X](\Theta) & \\ & & \partial_X M[X](\Theta) \end{pmatrix},$$
$$\partial_X \mathbf{L}[X](\Theta) = \begin{pmatrix} \partial_X L[X](\Theta) & & \\ & \partial_X L[X](\Theta) & \\ & & \partial_X L[X](\Theta) \end{pmatrix},$$

where $\partial_X M[X](\Theta) = \frac{\mathrm{d}}{\mathrm{d}\epsilon} M[X + \epsilon\,\Theta]|_{\epsilon=0}$ and $\partial_X L[X](\Theta) = \frac{\mathrm{d}}{\mathrm{d}\epsilon} L[X + \epsilon\,\Theta]|_{\epsilon=0}$. Finally, we can compute the descent direction in $\mathbb{R}^{m\sharp I}$ of the energy $\hat{W}$ at a given simplicial mesh $\mathcal{M}[X]$ described by the nodal vector $\bar{X}$ and obtain

$$\overline{\mathrm{grad}_X \hat{W}[X]} = \left(\partial_X \hat{W}[X](\Phi_r e_s)\right)_{r\in I,\, s=1,\cdots,m},$$

where e_s denotes the sth coordinate direction in $\mathbb{R}^m$.
In the concrete numerical algorithm we now perform a gradient descent method with the Amijo step size control starting from the initial position given by the previous time step.

Polygonal curve. We obtain for the derivatives of the mass matrix (using again the usual Kronecker symbol δ_{ir}) with respect to a variation of node r in direction s

$$\partial_X M[X](\Phi_r e_s) = \mathrm{diag}\left(\frac{(X^s_{i-1}-X^s_i)(\delta_{(i-1)r}-\delta_{ir})}{2Q_i} + \frac{(X^s_i-X^s_{i+1})(\delta_{ir}-\delta_{(i+1)r})}{2Q_{i+1}}\right),$$

where as above $Q_i = |X_i - X_{i-1}|$. Furthermore, we get for the derivatives for the stiffness matrix in the same direction

$$\partial_X L[X](\Phi_r e_s) = \mathrm{tridiag}(\partial_X L_{i-1}, \partial_X L_i, \partial_X L_{i+1}),$$

where

$$\begin{aligned} \partial_X L_{i-1} &:= \frac{(X^s_{i-1} - X^s_i)(\delta_{(i-1)r} - \delta_{ir})}{Q_i^3}, \\ \partial_X L_i &:= -\frac{(X^s_{i-1} - X^s_i)(\delta_{(i-1)r} - \delta_{ir})}{Q_i^3} - \frac{(X^s_i - X^s_{i+1})(\delta_{ir} - \delta_{(i+1)r})}{Q_{i+1}^3}, \\ \partial_X L_{i+1} &:= \frac{(X^s_i - X^s_{i+1})(\delta_{ir} - \delta_{(i+1)r})}{Q_{i+1}^3}. \end{aligned}$$

Triangular surfaces. The first variation of $|T|$ with respect to a variation of node r in direction s is given by

$$\partial_X |T|(\Phi_r e_s) = \frac{1}{2}\frac{F_1 \wedge F_2}{|F_1 \wedge F_2|} D_s^{90} P_s F_r\,,$$

where P_s is a projection onto the plane spanned by the vectors e_{s-1} and e_{s+1} and D_s^{90} a counter-clockwise rotation of 90 degree in this plane. It is suitable to assume vertex indices to be in $\{0, 1, 2\}$ and take them modulo 2 if this is not the case. Now, we obtain for the derivative of the local mass matrix with respect to a variation of node r in direction s

$$\partial_X m(T)(\Phi_r e_s) = \frac{\partial_X |T|(\Phi_r e_s)}{12} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} .$$

The corresponding derivative of the local stiffness matrix is given by

$$\begin{aligned} \partial_X l(T)_{ij}(\Phi_r e_s) = \frac{1}{4|T|} &((\delta_{r(i-1)} - \delta_{r(i+1)}) F_j^s + (\delta_{r(j-1)} - \delta_{r(j+1)}) F_i^s) \\ &- \frac{\partial_X |T|(\Phi_r e_s)}{4|T|^2} F_i \cdot F_j . \end{aligned}$$

5 Numerical Results

We have applied the developed numerical algorithm to the evolution of curves in $\mathbb{R}^2$ and $\mathbb{R}^3$ and of two dimensional surfaces in $\mathbb{R}^3$. Here, we present first results which in particular demonstrate the robustness of the proposed method. In fact, the applications underline that time steps up to the order the spatial grid size h are feasable.

Willmore flow for curves. At first we have studied the evolution of circles in $\mathbb{R}^2$. Figure 3 shows that the numerical solution will approximate the known exact solution of expanding circles. In the next computations we consider a slight generalization of the above Willmore flow model. In fact, we add $\lambda\, a[x]$ to the Willmore energy, where λ is a fixed constant and $a[x]$ denotes the length of the

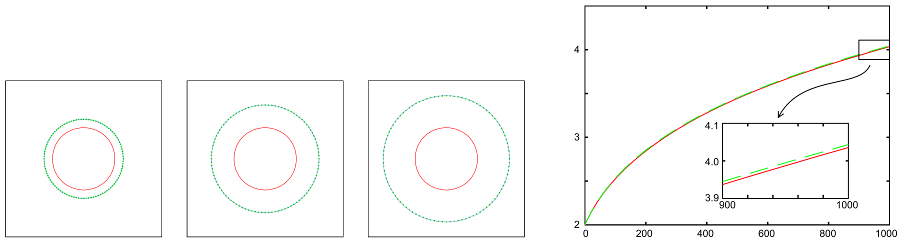


Fig. 3. A circle of radius $R_0 = 2$ expands in two dimension due to its propagation via Willmore flow (left). The exact solution (grey dashed line) and the corresponding discrete solution computed by the two step time discretization for 200 polygon vertices and a time step size which equals the grid size (green crosses) are plotted for different times $t = 100\,h$, $500\,h$, $1000\,h$. The radius of the growing circle under Willmore flow is plotted for the known continuous solution (green) and the discrete solution (red) (right).

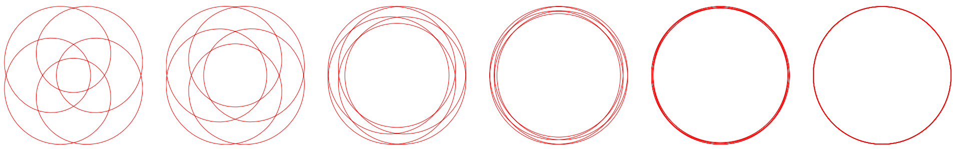


Fig. 4. The evolution of a planar hypocycloid towards a fivefold covering of a circle is shown at times $t = 0.0$, $t = 685.7$, $t = 2987.4$, $t = 4850.1$, $t = 7965.8$, $t = 10630.6$. The curves are graphically rescaled to have similar size. Here the computational parameters were $\lambda = 0.025$, $N = 200$ and $\tau = h = 0.5493$.

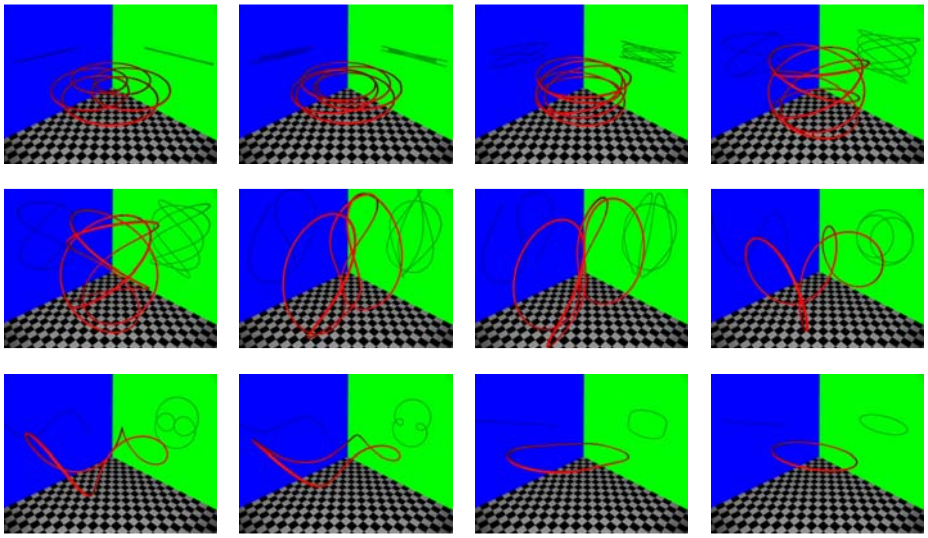


Fig. 5. Evolution of a vertically perturbed hypocycloid towards a circle under Willmore flow is shown at times $t = 0.0$, $t = 1348.9$, $t = 4467.1$, $t = 5511.4$, $t = 6555.7$, $t = 7406.6$, $t = 8257.2$, $t = 9108.4$, $t = 9297.0$, $t = 9361.3$, $t = 9426.8$, $t = 9489.1$. The computational data were $\delta = 0.1$, $\lambda = 0.025$, $N = 200$ and $\tau = h = 0.5$.

curve. Here, λ can be regarded as a Lagrangian multiplier with respect to a length constraint. Hence, for proper choices of λ the generalized model avoids expansion.

If X represents a discrete closed curve as above, we obtain for the discrete length functional $A[X] = \sum_{i \in I} Q_i$. Furthermore, its gradient vector in $\mathbb{R}^{m\sharp I}$ is given by $\overline{\text{grad}_X A[X]} = L[X]\bar{X}$.

As a first example for the resulting flow we consider the evolution of an ellipse towards a circle under the elastic flow (cf. the first rows in Figure 1 and 2).

The initial parametrization is given as

$$x_0(t) = (\sin(t), 4\ \cos(t), 0) \text{ for } t \in [0, 2\pi].$$

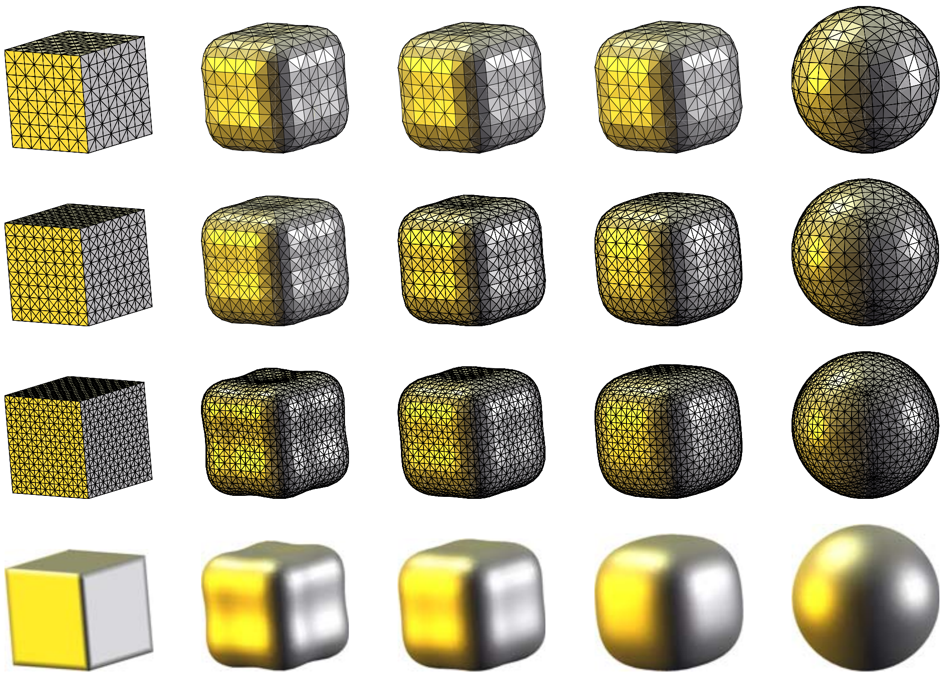


Fig. 6. Willmore flow for an initial cubical surface with 768 (1st row), 1536 elements (2nd row), and 3072 elements (3rd, 4th rows) are shown at times $t = 0.0$, $t = 0.0366$, $t = 0.0732$, $t = 0.1464$, and $t = 0.366$, where $h = 0.07322$, $h = 0.0517766$, and $h = 0.0366$, respectively.

The computational parameters are $h = 0.0632847$, $\tau = h$ and $\lambda = 0.025$. On observes that the ellipse evolves to a circle and the polygonal vertices stay well-distributed on the evolving curve. In the next application we pick up an example already discussed by Dziuk and Deckelnick in [32], where a hypocycloid is considered as initial data. Here, the parametrization of the initial curve is given by

$$X_0(t) = \left(-\frac{5}{2}\cos(t) + 4\ \cos(5t), -\frac{5}{2}\sin(t) + 4\ \sin(5t), \delta\ \sin(3t)\right)$$

with $\delta = 0$. In $\mathbb{R}^2$ the initial curve evolves to a fivefold covering of a circle (cf. Figure 4) since multiple coverings of a circle are stable stationary solutions in the codimension one case [9]. This is not true for higher codimension with $m \geq 3$. If we start with an initial curve slightly perturbed in vertical direction, we have chosen $\delta = 0.1$, the curve begins to unfold and evolves to a single circle (cf. Figure 5).

Willmore flow for surfaces. Spheres are minima of the Willmore functional with energy 8π. In our first example for two dimensional surfaces in $\mathbb{R}^3$ we show

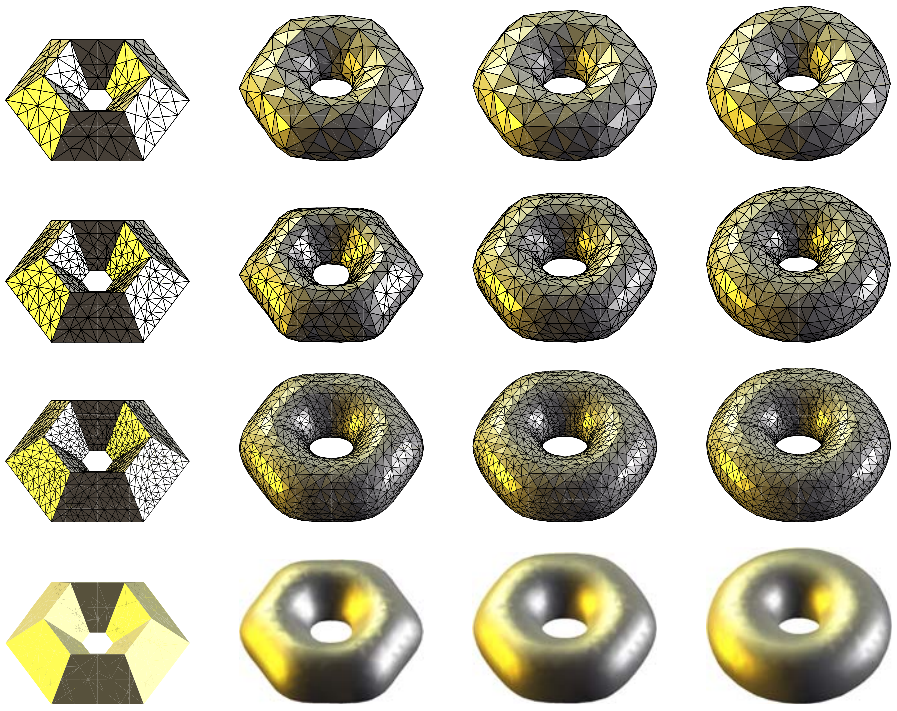


Fig. 7. Different time steps of Willmore flow towards the Clifford torus for an initial macro torus with 522 (1st row), 1224 (2nd row), and 2736 elements (3rd and 4th row). We render the surfaces at times $t = 0.0$, $t = 0.09$, $t = 0.15$, and $t = 0.97$, where $h = 0.0977$, $h = 0.0745$, and $h = 0.0089$, respectively.

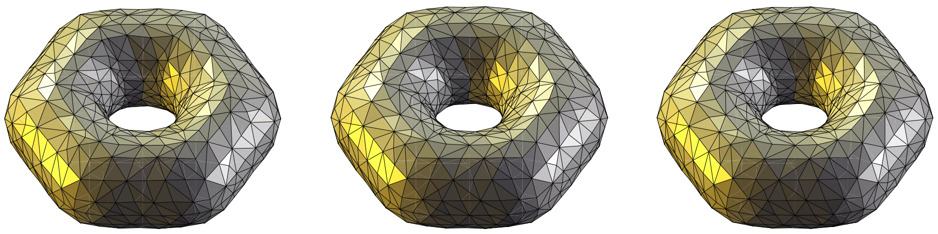


Fig. 8. From the evolution towards the Clifford torus (cf. Figure 7) discrete surfaces at time $t = 0.3735$ are shown based on a computation with time step sizes towards a sphere for an initial macro torus with 1224 elements for different time steps sizes (from left to right) $\tau = h^4$, $\tau = h^2$, and $\tau = h$, where $h = 0.0745$.

the evolution of a cubical surface into a round sphere (cf. Figure 6). In Figure 7 we depict the evolution of a coarse polygonal approximation of a torus towards the Clifford torus $\mathcal{M} = \{x \in \mathbb{R}^3 | (1 - \sqrt{x_1^2 + x_2^2})^2 + x_3^2 = \frac{1}{2}\}$. Finally, in Figure

8 we compare the discrete evolution at a fixed time for different choices of the time step τ used in the computation.

Acknowledgement

Nadine Olischläger has been supported by the DFG collaborative research center SFB 611 at Bonn University and the Hausdorff Center for Mathematics. Furthermore, the authors are grateful to Orestis Vantzos for helping with the rendering in Figure 5.

References

1. Willmore, T.: Riemannian Geometry. Claredon Press, Oxford (1993)
2. Simonett, G.: The Willmore Flow near spheres. Diff. and Integral Eq. 14(8), 1005–1014 (2001)
3. Kuwert, E., Schätzle, R.: The Willmore flow with small initial energy. J. Differential Geom. 57(3), 409–441 (2001)
4. Nitzberg, M., Mumford, D., Shiota, T.: Filtering, Segmentation and Depth. LNCS, vol. 662. Springer, Heidelberg (1993)
5. Mumford, D.: Elastica and computer vision. In: Bajaj, C. (ed.) Algebraic Geometry and Its Applications, pp. 491–506. Springer, New York (1994)
6. Yoshizawa, S., Belyaev, A.G.: Fair triangle mesh generation with discrete elastica. In: Proceedings of the Geometric Modeling and Processing; Theory and Applications (GMP 2002), Washington, DC, USA, pp. 119–123. IEEE Computer Society, Los Alamitos (2002)
7. Chan, T.F., Kang, S.H., Shen, J.: Euler's elastica and curvature-based inpainting. SIAM Appl. Math. 63(2), 564–592 (2002)
8. Polden, A.: Closed Curves of Least Total Curvature. SFB 382 Tübingen, Preprint 13 (1995)
9. Polden, A.: Curves and Surfaces of Least Total Curvature and Fourth-Order Flows. Dissertation, Universität Tübingen (1996)
10. Kuwert, E., Schätzle, R.: Removability of Point Singularities of Willmore Surfaces. Preprint SFB 611, Bonn (2002)
11. Kuwert, E., Schätzle, R.: Gradient flow for the Willmore functional. Comm. Anal. Geom. 10(5), 1228–1245 (2002) (electronic)
12. Dziuk, G., Kuwert, E., Schätzle, R.: Evolution of elastic curves in $\mathbb{R}^n$: existence and computation. SIAM J. Math. Anal. 33(5), 1228–1245 (2002) (electronic)
13. Droske, M., Rumpf, M.: A level set formulation for Willmore flow. Interfaces and Free Boundaries 6(3), 361–378 (2004)
14. Deckelnick, K., Dziuk, G.: Error analysis of a finite element method for the Willmore flow of graphs. Interfaces and Free Boundaries 8, 21–46 (2006)
15. Barrett, J.W., Garcke, H., Nürnberg, R.: A parametric finite element method for fourth order geometric evolution equations. J. Comp. Phys. 222, 441–467 (2007)
16. Bobenko, A., Schröder, P.: Discrete Willmore flow. In: SIGGRAPH (Courses). ACM Press, New York (2005)
17. Dziuk, G.: Computational parametric Willmore flow. Preprint Fakultät für Mathematik und Physik, Universität Freiburg 13-07 (2007)

18. Welch, W., Witkin, A.: Variational surface modeling. In: SIGGFRAPH Computer Graphics, vol. 26, pp. 157–166 (1992)
19. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proc. of SIGGRAPH 2000, New Orleans, USA, pp. 417–424 (2000)
20. Bertalmio, M., Bertozzi, A., Sapiro, G.: Navier-stokes, fluid dynamics, and image and video inpainting. In: IEEE Proceedings of the International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 355–362 (2001)
21. Clarenz, U., Diewald, U., Dziuk, G., Rumpf, M., Rusu, R.: A finite element method for surface restoration with smooth boundary conditions. Computer Aided Geometric Design 21(5), 427–445 (2004)
22. Rane, S.D., Remus, J., Sapiro, G.: Wavelet-domain reconstruction of lost blocks in wireless image transmission and packet-switched networks. In: 2002 International Conference on Image Processing. Proceedings, September 22-25, vol. 1 (2002)
23. Xu, G., Pan, Q.: G^1 surface modelling using fourth order geometric flows. Computer-Aided Design 38(4), 392–403 (2006)
24. Luckhaus, S., Sturzenhecker, T.: Implicit time discretization for the mean curvature flow equation. Calc. Var. 3, 253–271 (1995)
25. Chambolle, A.: An algorithm for mean curvature motion. Interfaces and free Boundaries 6, 195–218 (2004)
26. Bellettini, G., Caselles, V., Chambolle, A., Novaga, M.: Crystalline mean curvature flow of convex sets. Technical Report 7641, UMR CNRS (2004)
27. Chambolle, A., Novaga, M.: Convergence of an algorithm for anisotropic mean curvature motion. SIAM J. Math. Anal. 37, 1978–1987 (2006)
28. Dziuk, G.: An algorithm for evolutionary surfaces. Numer. Math. 58, 603–611 (1991)
29. Dziuk, G.: Finite elements for the Beltrami operator on arbitrary surfaces. In: Hildebrandt, S., Leis, R. (eds.) Partial Differential Equations and Calculus of Variations. Lecture Notes in Mathematics, vol. 1357, pp. 142–155. Springer, Heidelberg (1988)
30. Thomée, V.: Galerkin finite element methods for parabolic problems, 2nd edn. Springer Series in Computational Mathematics, vol. 25. Springer, Berlin (2006)
31. Diewald, U., Morigi, S., Rumpf, M.: A cascadic geometric filtering approach to subdivision. Computer Aided Geometric Design 19, 675–694 (2002)
32. Deckelnick, K., Dziuk, G.: Error analysis for the elastic flow of parametrized curves. Preprint Fakultät für Mathematik und Physik, Universität Freiburg 14-07 (2007) (to appear in Math. Comp.)

Surface Triangulation and the Downstream Effects on Flattening

S.S. Parwana and R.J. Cripps

School of Mechanical Engineering, University of Birmingham, UK
r.cripps@bham.ac.uk

Abstract. Many applications within computer aided engineering require the support of a triangulation to gain geometrical and structural insight into the original surface being approximated. Therefore, downstream applications such as computer numerically controlled (CNC) machine path generation, finite element analysis (FEA) and flattening are triangulation dependant processes. However, despite the importance of triangulation in these applications, very little work has been focused on the downstream effects of triangulation. This paper investigates these effects on the application of surface flattening and demonstrates how subtle, but important changes in the triangulation can have a major impact on the flattening results.

1 Introduction

The triangulation of trimmed and untrimmed parametric surfaces is a necessary process needed for many computer aided design (CAD) and computer aided manufacture (CAM) methods. By generating a computer based geometric approximation of the surface shape using a set of triangles, downstream engineering applications such as CNC machine path generation, FEA, visualisation and flattening can be carried out.

Downstream CAD/CAM applications are dependant on a triangulation since they operate on its topological data. To generate the initial surface approximation, many different methods for creating a triangulation of trimmed and untrimmed surfaces exist [2]–[4], [7]–[10], each using different success criteria to evaluate the resulting approximation, such as the operating speed of the algorithm, or the triangle shapes generated. For example, Rockwood et al. [9] use a parameter space method that subdivides the 2D trimming region into monotonic sub-regions, each sub-region is then sampled uniformly and the points are then joined to create triangles using a tiling and coving method. The completed triangulation is finally mapped into 3D. The emphasis of this algorithm is computational speed enabling real-time rendering of trimmed surfaces. Piegl and Tiller [8], on the other hand, present a method which is computationally more expensive but produces a better geometrical representation of the surface without oversampling points. Their method uses a 3D geometrical surface subdivision technique to sample 3D points on a given trimmed surface using a flatness criteria. These points are then joined by Delaunay triangulation [[6], [11]]. Despite the diversity of methods available, users must still select a triangulation for a downstream application based on personal preference or experience. This is because little literature is available on the required characteristics of a surface triangulation to optimise the results for downstream engineering applications.

Within the field of FEA, it has been established that the quality of a triangulation can affect the final results of a simulation. In particular, it has been shown that triangles with high aspect ratios, i.e. 'long thin' triangles, are undesirable as they are less numerically stable and can reduce the convergence rate of an FEA simulation [12]. This type of triangulation criteria has been adopted by many other areas of CAD/CAM without justification. Piegl and Tiller [8] and Kumar et al. [2] attempt to eliminate the occurrence of 'long thin' triangles from their triangulation schemes, but they give no indication as to why this would be needed.

Triangle shapes can also become a source of debate when choosing a triangulation scheme to use for a downstream application. When triangulating a trimmed surface, the boundary edge triangles generally give less control, and often many non-uniform triangle shapes are generated within these regions. Excluding the boundary edges, the bulk of the surface approximation is commonly expressed in terms of right angled triangles (RATs) [[2]-[4], [7]-[10]]. However, unlike equilateral triangles, RATs are not unique in terms of connectivity which may affect the performance of a downstream application.

This paper focuses on the engineering application of flattening and demonstrates the downstream effects of triangulation. To remove any other possible outside influences that could affect the performance of the downstream application, any non-uniform triangles caused by boundary edge triangles will be avoided by only producing triangulations of untrimmed surfaces. This does not restrict the method, as boundaries can be recovered by projecting onto the resulting flattened surface. As RATs are commonly used in the triangulation of trimmed surfaces throughout industry, this paper will focus on this type of triangle. It will also introduce the concept of right angle triangle configurations (RATCs) within a triangulation and will show how different configurations of RATs, generated from the same point sampling, produce different flattening results.

2 Surface Flattening

The method of flattening is used to map a 3D surface into a two dimensional (2D) plane, and is applicable to industries such a shoe making, textile and ceramic decorations. These fields of work all involve making an item from a 2D material, which will ultimately be placed onto a 3D model, such as a tea pot or human body part. In conventional practice, a 2D pattern is designed by a field expert, which is then used to create the item from the 2D material. Once completed, it is placed onto the 3D model after which alterations can be made to produce a better fit. Designing an initial 2D pattern is a specialised craft and can be a complicated task. However, use of a flattening simulation allows a designer to create the final 3D model, which can then be flattened to produce a 2D pattern. It has been shown that this approach can produce 2D patterns that have a better initial fit as well as decreased production lead-time [[1], [14]]. There are many methods available for flattening a 3D surface. However, the method chosen for this paper is a modification of McCartney et al.'s [5] algorithm, which is a purely geometrical approach, presented by Cader et al. [1].

2.1 McCartney et al.'s Flattening Method

McCartney et al.'s original method operates by producing an initial flattening, which keeps the edge lengths of each triangle in 2D the same as the corresponding 3D triangle. The first stage of the algorithm is to triangulate a surface, producing a set of triangles $\mathbf{T}$. The first triangle to be flattened is chosen from the triangulation and is known as the seed triangle, $\mathbf{t}_s$,that has the 3D vertices, $(\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3)$, (Fig. 1a) and edge lengths

$$l_1 = \|\mathbf{V}_2 - \mathbf{V}_1\|, l_2 = \|\mathbf{V}_2 - \mathbf{V}_3\|, l_3 = \|\mathbf{V}_3 - \mathbf{V}_1\|$$

The first two vertices are placed into 2D using the coordinates $\mathbf{V}_1^{'} = (0,0)$ and $\mathbf{V}_2^{'} = (l_1, 0)$. The next stage is to create two 2D circles $\mathbf{C}_1$ and $\mathbf{C}_2$, which have the centre points $\mathbf{V}_1^{'}$, and $\mathbf{V}_2^{'}$, as well as radii l_1, and l_2 respectively. The position of the third vertex $\mathbf{V}_3^{'}$ in 2D is determined by the intersection point between $\mathbf{C}_1$ and $\mathbf{C}_2$ (Fig. 1b). Once the seed is flattened, any triangle in $\mathbf{T}$ that shares an edge with $\mathbf{t}_s$ may be flattened next. As the next triangle will share an edge with $\mathbf{t}_s$, two of its points will already have been flattened, therefore, only the third vertex needs to be calculated and this is done by using the circle intersection method. Instances where all three vertices of a triangle have already been flattened will occur during the process, which will mean that one vertex in 3D may have more than one position in 2D.

If the initial surface has zero curvature along one or both of its principal curvature axes, then the surface is defined as a developable surface. Examples of developable surfaces are planes, cylinders or cones. If a surface is developable it

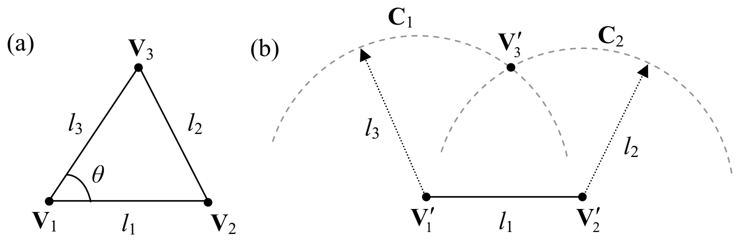


Fig. 1. (a) 3D triangle. (b) 2D flattening of 3D triangle.

will flatten with no deformation to the original shape, however, it will be shown in §3 that having a developable surface does not guarantee that the approximating triangulation will not distort. If the surface is non-developable, then tearing and overlapping will occur at the points where one 3D vertex has multiple positions in 2D. If the surface is developable, generally, the multiple 2D positions will all be the same and therefore no distortion should occur.

To remove any distortion which can arise from this flattening algorithm, McCartney et al., use an energy release method, which extracts strain energy from the flattened triangulation and reconnects the topology. Due to the circle intersection method runtime modifications of the flattened topology are not possible and have to be postponed until the end. This is because the vertices that have been flattened have the potential to act as centre points for circles, and moving a vertex will essentially move a possible centre of a circle. Over time, this movement can accumulate so that the distance between the centre points is large enough to force the circles not to intersect one another.

2.2 Modified McCartney's Flattening

The modified algorithm utilises a cosine method to determine the position of the triangle vertices in 2D rather than the intersection of circles. In the case of the seed triangle, the first two vertices would be flattened using the same method as McCartney et al. [5]. The edge lengths are then used to calculate the angle θ incident to $\mathbf{V}_1$ (Fig. 1a) using the cosine rule:

$$\theta = \cos^{-1}\left(\frac{l_1^2 + l_3^2 - l_2^2}{2l_1 l_3}\right)$$

Therefore, the coordinates of the three flattened vertices $(\mathbf{V}_1^{'}, \mathbf{V}_2^{'}, \mathbf{V}_3^{'})$, which preserve the 3D edge lengths of the original triangle, are given as:

$$\mathbf{V}_1^{'} = (0, 0), \mathbf{V}_2^{'} = (l_1, 0), \mathbf{V}_3^{'} = (l_3 \cos(\theta), l_3 \sin(\theta))$$

This produces a result which is equivalent to the circle intersection method, but has the advantage that it allows for runtime modifications of the flattened triangulation, which eliminates any tearing or overlapping as it occurs. Therefore, in the situation where a duplicate 2D position for an existing flattened vertex has to be calculated, an unbiased average of the two different positions is taken to create a single position. It is again worth noting that if the surface is developable, the position of the two vertices will be the same.

Cader et al. [1] have noted that the choice of seed triangle can effect the final flattening. As the flattened seed triangle geometry is constrained, any distortion spreads outwards from this seed triangle. Another aspect that can affect the final output is the mapping order, i.e. the order in which the triangles are sequentially flattened. Therefore, this method also provides a mapping order which is used to decide which triangle will be flattened next. The seed triangle is denoted level 0 and will be the first triangle to be flattened. Any triangle which shares an edge with the seed triangle will be denoted level 1, and any triangle which shares an edge will a triangle from level 1 will be denoted as level 2, etc. This method is continued so that a 'circular' mapping order is created.

2.3 Flattening Success Criteria

This modified version of the flattening method does have one limitation in that it was created for surfaces that are developable or 'almost' developable. Consequently, the averaging process used to create a single position for a vertex can cause the orientation of the triangle vertices to change from the convention of anti-clockwise to clockwise. This causes a topological violation, as inconsistent orientations of triangle vertices are introduced, causing the flattening to produce an unacceptable results with large changes in area and shape. Therefore, if at any instance, during the averaging process, the orientation of the flattened triangle vertices flip from anti-clockwise to clockwise, the algorithm terminates and the flattening is said to have failed.

2.4 Accuracy Measurements

A successful unwrapping does not guarantee an acceptable flattening. The accuracy of the flattening can be assessed using differences in area and shape as proposed by Wang et al. [14]. The area difference measures the change in surface area pre- and post- flattening. The relative area difference is given by:

$$\delta A = \frac{A^o - A^f}{A^o}$$

where A^o is the original area of the 3D surface and A^f is the area of the flattened surface. Since analytical forms of the flattened surface cannot be determined, the areas are approximated by summing triangle areas within each triangulation, i.e.

$$A^o = \sum_{i=0}^{n-1} \Delta_i^o \quad \text{and} \quad A^f = \sum_{i=0}^{n-1} \Delta_i^f$$

where Δ_i^o is the area of the ith triangle from the 3D triangulation consisting of n triangles and Δ_i^f is the area of the ith flattened triangle.

The shape difference measures the change in the surface curve lengths pre- and post- flattening, where the curve length is defined as the length of an arbitrary iso-parameter line on the surface. The final relative edge length difference is given by:

$$\delta S = \frac{L^o - L^f}{L^o}$$

where L^o is the actual curve length of the curve segment on the original surface and L^f is the corresponding curve length in the flattened model. Again, analytical forms of the curve lengths cannot be determined for the flattened model, so

L^o and L^f are approximated by summing the length of each triangle edge in the corresponding triangulation, i.e.

$$L^o = \sum_{j=0}^{m-1} E_j^o \quad \text{and} \quad L^f = \sum_{j=0}^{m-1} E_j^f$$

where E_j^o is the edge length of the jth edge in the 3D triangulation consisting of m edges and E_j^f is the edge length of the jth edge in the flattened triangles.

3 Developable Surface Flattening Test

If a 3D surface is developable, then it will flatten into 2D without any distortion to the original shape.

However, this is not true of a developable surface's triangulation. In order to flatten a developable surface without distortion, the triangulation must inherit the property of developability from the original surface.

In particular, the topological information contained within the surface approximation, such as the vertices, edges and faces, must also contain an approximation of zero curvature along the principal axes. One way to achieve this is to ensure that the strings of points that are sampled on a developable surface for triangulation are parallel with any axis that has zero curvature. Consider a developable surface defining part of a cylinder which has zero curvature along constant v values as shown in Fig. 2.

A triangulation consisting of RATs, was constructed from a grid of 20×20 points lying on the surface evaluated at equally spaced parametric points (u_i, v_i). The grid size was used since it produced a good surface approximation, without being computationally expensive. This triangulation consisted of 772 triangles that was flattened using the modified flattening algorithm and produced no deformation of the original triangulation. Fig. 3, shows the original 3D triangulation of the surface and the flattened 2D result. The line of zero curvature,

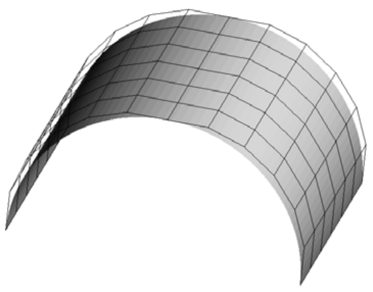


Fig. 2. Cylinder part surface

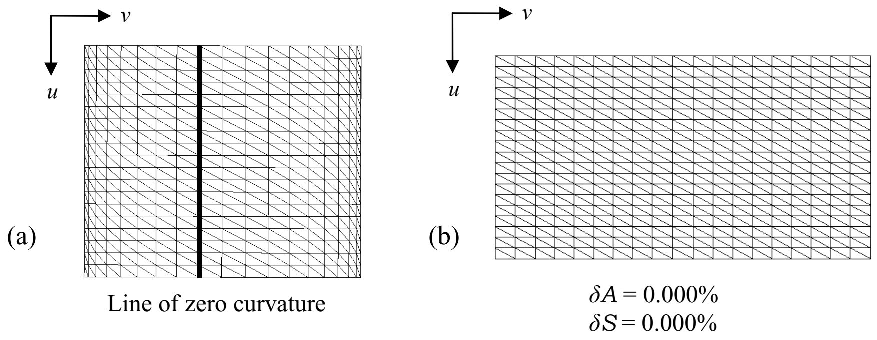


Fig. 3. (a) 3D triangulation of part cylinder. (b) Flattened triangulation.

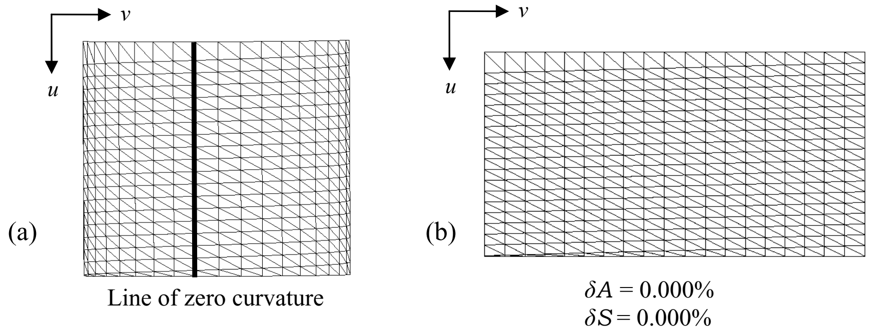


Fig. 4. (a) 3D triangulation of part cylinder. (b) Flattened triangulation.

which is parallel to the principal axis of curvature, is also shown on the 3D triangulation. It is noted that the sampling points all lie parallel to the principal axis of curvature.

The surface was sampled a second time using the same grid size. Points in the u direction were skewed by distorting the parameter spacing along one boundary and blending to the opposite fixed boundary. Points in the v direction remained fixed, following constant values of v (Fig. 4a). The strings of points continued to follow the lines of zero curvature and the triangulation topology inherited enough information from the original surface to approximate a developable surface which flattened without distortion (Fig. 4b).

A third grid of points was sampled with constant u values but skewed v values, using the same approach as previously described, so that they no longer followed the line of zero curvature along the principal axis of curvature (Fig. 5a). The topological information of the triangulation no longer represents a developable surface, which resulted in a distorted flattening (Fig. 5b).

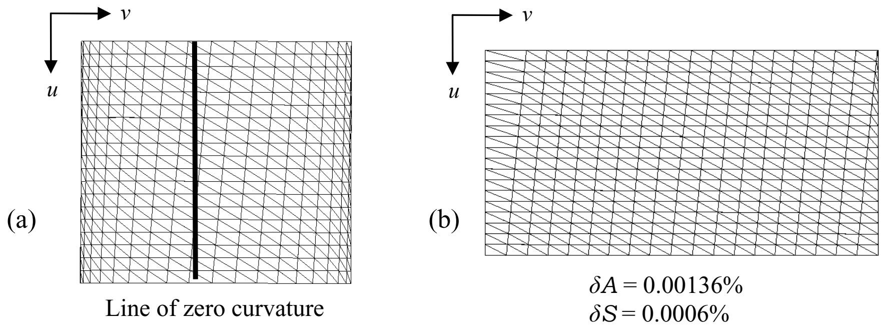


Fig. 5. (a) 3D triangulation of part cylinder. (b) Flattened triangulation.

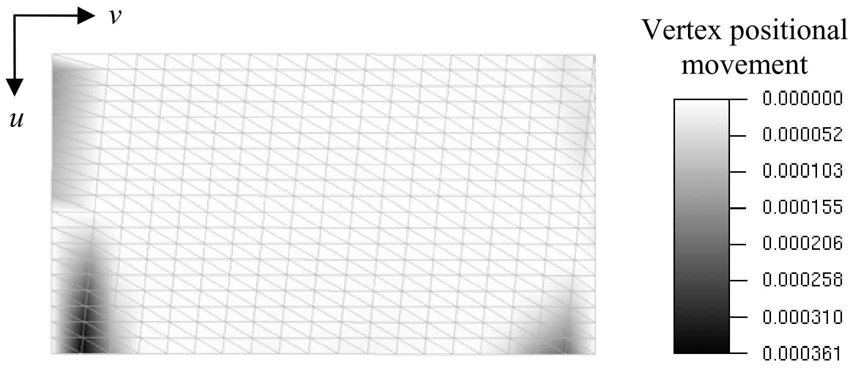


Fig. 6. Flattened triangulation showing vertex positional movement

Fig. 6 shows the magnitude of positional movement for each vertex, i.e. the difference between its original flattened position and its final position after all averaging.

Fig. 7 shows the area difference between the 3D triangles and their corresponding flattened triangles. It can be seen that the regions of the flattened model that have vertex positional movement are the same regions that have an area change. The distortion caused by this triangulation is an unexpected result for a developable surface, as it should flatten without inducing any vertex positional movement, or triangle area change. This suggests that the sampling points should always follow along the line of zero curvature to approximate a developable surface. Furthermore, a developability measure of a triangulation, rather than the original surface, may prove to be more appropriate when estimating the performance of a flattening. In this test the area and shape differences are small; however, increasing the magnitude of the skewing in the v parameter direction will eventually cause the flattening to completely fail.

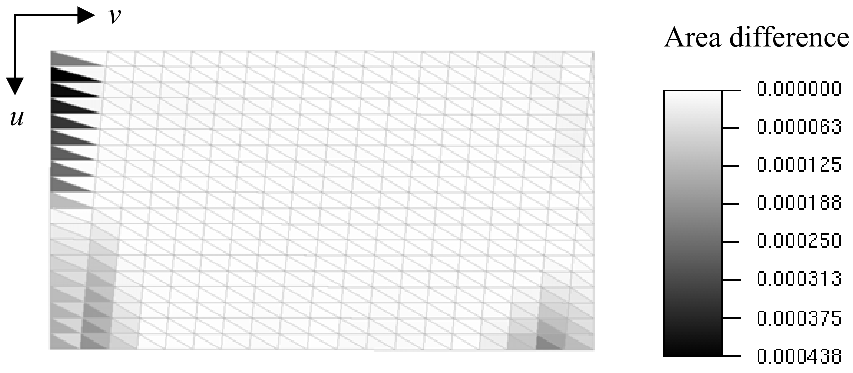


Fig. 7. Flattened triangulation showing triangle area difference

4 RAT Configurations

Given a set of regularly sampled 2D points in a plane, the triangulation is not unique. A unique triangulation can be created by sampling a given domain so that the points can be joined to create a desired triangle shape. For example, the domain could be sampled at points that can be connected to form a triangulation consisting of equilateral triangles. In this case, there is a unique configuration of equilateral triangles.

Randomly sampled points are more complex and a different approach is needed. The Delaunay triangulation [[6], [11]] provides a possible solution. Thus given a set of points, P, Delaunay will generally produce a unique triangulation solution, independent of the starting point. It does this by ensuring that for every triangle within the triangulation, no other point within the set P, other than the points which make up the triangle vertices, are contained within the circumcircle of that triangle. It also ensures that the minimum angle of every triangle is maximised, which is essentially trying to make the triangles as close to equilateral as possible.

However, a unique solution is not always guaranteed for random points, as has been shown by Sugihara and Inagaki [13]. If four points all lie on the same circumcircle, the points can be joined to make two triangles in two different ways, by performing an interior edge flip, and still produce a legal Delaunay triangulation. If more than four points lie on a common circumcircle, many more legal solutions are possible. Consider if the whole 2D plane was sampled for uniformly spaced points, which could be joined to create RATs, then using a Delaunay triangulation to achieve a unique solution would not be possible. This is because four points would always be on a circumcircle of a triangle. Therefore, the final triangulation produced would be dependant on the starting point. As no unique solution exists, the user must decide how best to join the vertices to create RATs. In this type of situation, many different RATC can

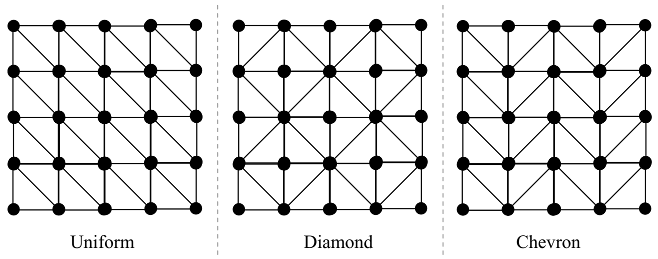


Fig. 8. Three types of global right angle triangle configurations

form a legal triangulation consisting of RATs. With RATs being used by many surface triangulation methods, and hence for downstream applications such as flattening, the choice of RATC can be viewed as a variable and can have a dramatic impact on the downstream application of flattening.

4.1 Global Triangulation Configurations

For a 2D domain which has been sampled to form RATs, there can exist thousands of permutations for the triangulation that will still form a legal triangulation. Three different RATCs will be used to investigate the effect of the configuration on the flattening process (Fig. 8).

These are called uniform, diamond and chevron. The uniform and diamond global RATCs were chosen as they are commonly used in many surface triangulation schemes. The chevron configuration was introduced as an alternative RATC. As can be seen, all three configurations are legal RAT triangulations, generated from the same point sampling, and could result in an application producing different results.

4.2 RATC Test

To consider the effects of RATC on the flattening process, an undevelopable NURBS test surface, Fig. 9, was sampled uniformly using a 60×60 grid of points and triangulated in each of the three different global configurations. Each of these triangulations consisted of 6962 triangles and was flattened using the same seed triangle. This seed triangle was chosen as it was defined by the same vertices and edges in each of the three configurations, thus giving an unbiased starting point to each flattening simulation. The flattening results for the three triangulations are given in Table 1 and Figs. 10-12. The seed triangle is shaded in black to show its location.

As can been seen from Tab. 1, the flattening process is dependent on the global configuration of the triangulation. In the diamond configuration, only 82% of the triangles were processed before the flattening failed. However, the chevron configuration resulted in the triangulation flattening successfully with an area change of -0.27%, where a negative value indicates a reduction in area.

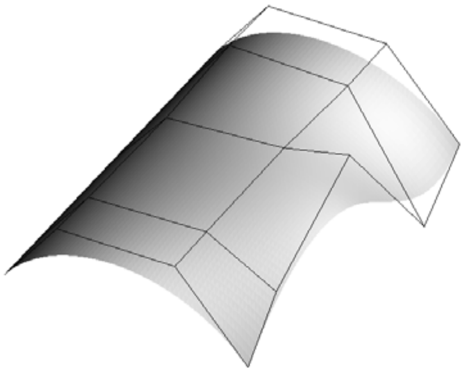


Fig. 9. Non-developable NURBS surface

Table 1. Flattening Results

Number of triangles in each triangulation 6962				
Global RATC	No of triangles flattened	Area Change (%)	Shape Change (%)	Successful
Diamond	5903	-15.23	-15.74	NO
Chevron	6962	- 0.27	- 0.05	YES
Uniform	6962	- 0.18	- 0.08	YES

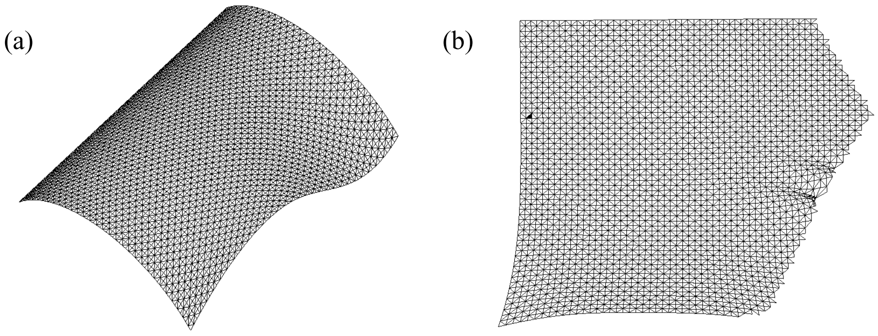


Fig. 10. (a) Diamond RAT 3D triangulation. (b) Flattening.

An unmodified version of this flattening would cause the triangulation to have overlapping triangles or gaps between triangles. However, the averaging process removes overlaps and tears, thus triangle areas can either increase or

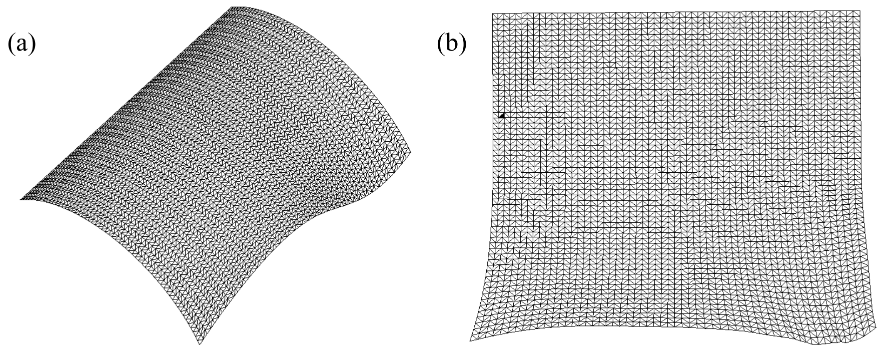


Fig. 11. (a) Half-Diamond RAT 3D triangulation. (b) Flattening.

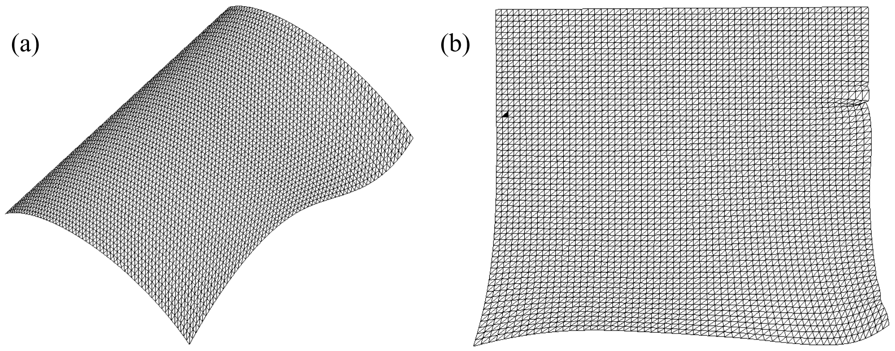


Fig. 12. (a) Uniform RAT 3D triangulation. (b) Flattening.

decrease in these regions. The flattening was further improved by using the uniform configuration, which reduced the area change by 33% compared to the chevron configuration.

Further, it can be seen from the Fig. 12, that whilst the uniform configuration induced less change in area, it produced some noticeable creasing in the top right corner of the final flattening. In contrast, the chevron triangulation resulted in visually less creasing (Fig. 11). This implies that for this surface, the chevron configuration preserves the original shape better, and this is reflected in the resulting shape change coefficients.

It can be seen from Figs. 11, 12 that both the chevron and uniform RATCs produce similar results for the left and top edges, but different outcomes for the right and bottom edges. The chevron configuration (Fig. 11) produces a better flattening for the right edge, and the uniform configuration (Fig. 12) creates a better approximation for the bottom edge. This could suggest that some RATCs may be optimised for certain surface characteristics, such as curvature. Therefore, the optimum RATC for a surface triangulation may not be a global one, but a collection of configurations, which will be placed strategically in different areas of a surface triangulation depending on the local surface characteristics.

Currently, the failure of a surface flattening would be attributed either to the type of triangulation, i.e. triangle shape, triangle size, or the flattening algorithm. However, this test has shown that the performance of the flattening algorithm is also dependent on the configuration of the RATs.

5 Discussion

It has been shown that the triangulation of a surface definition can impact on the performance of the triangulation dependant downstream application of flattening. Since the surface triangulation is the only means by which the flattening process can get information about the original surface definition, it is important that the triangulation approximates the original shape characteristics. It was shown in §3 that if the points sampled on a developable surface are not aligned to the lines of principal curvature, then the triangles generated from these points will not approximate a developable surface. Depending on the amount of deviation from these axes of principal curvature, the final flattening can induce distortion or fail altogether. Therefore, a measure for the developability of a surface triangulation, rather than the original surface would be a better gauge of determining the success of a surface flattening. Further research is required to identify connections between the surface characteristics and the RATC.

Since RATs are commonly used for the triangulation of both trimmed and untrimmed surfaces, the subject of RATCs was introduced and it was shown that this also has an effect on the downstream application. Although the test results for the test surface in §4 suggested that the uniform configuration would be the best overall choice, experimental evidence has shown this not always to be true in general. In fact, the diamond configuration can produce a better flattening result for some surface definitions.

Experimental evidence also suggests that chevron and uniform configuration flattenings have more in common than the diamond configuration flattenings. Since the only difference between each configuration is the connectivity of the points, the similarities may be due to the point valency. Both the chevron and uniform configurations have a valency of 6, whereas the diamond configuration has a mix of points with valency of either 8 or 4. This difference in valency could explain the variation in results between the configurations.

Changing the RATC of a triangulation can change the point valences and edges created, possibly altering the sequence of triangles being processed, even if the seed triangle used is the same (c.f. §4). Thus the triangulation clearly effects the flattening, as noted by Cader at al. [1].

It was also suggested in §4 that an optimum RATC could be one that consisted of many different RATCs. However, further work needs to be done to establish which RATC perform better on certain surface characteristics.

Acknowledgments. The authors are pleased to acknowledge the financial support from EPSRC (EP/P502322/1) and Delcam International.

References

1. Cader, R.A., Ball, A.A., Cripps, R.J.: Pattern shape design for ceramic ware: an integrated solution. International Journal of Computer Integrated Manufacturing 19(3), 287–293 (2005)
2. Kumar, G.V.V.R., Srinivasan, P., Shastry, K.G., Prakash, B.G.: Geometry based triangulation of multiple trimmed NURBS surfaces. Computer-Aided Design 33, 439–454 (2001)
3. Kumar, S., Manocha, D.: Interactive Display of Large Scale NURBS models. IEEE Transactions on Visualisation and Computer Graphics 2(4), 323–336 (1996)
4. Luken, W.L.: Tessellation of trimmed NURB surfaces. Computer-Aided Geometric Design 13, 163–177 (1996)
5. McCartney, J., Hinds, B.K., Seow, B.L.: The flattening of triangulated surfaces incorporating darts and gussets. Computer-Aided Design 31, 249–260 (1999)
6. O'Rourke, J.: Computational Geometry in C, 2nd edn. Cambridge University, Cambridge (2001)
7. Peterson, J.W.: Tessellation of NURBS surfaces. In: Graphics Gems IV, pp. 286–320. Academic Press, London (1994)
8. Piegl, L.A., Tiller, W.: Geometry based triangulation of trimmed NURBS surfaces. Computer-Aided Design 30(1), 11–18 (1998)
9. Rockwood, A., Heaton, K., Davis, T.: Real-time rendering of trimmed surfaces. ACM Computer Graphics 23(3), 107–116 (1989)
10. Sadoyan, H., Zakarian, A., Avagyan, V., Mohanty, P.: Robust uniform triangulation algorithm for computer aided design. Computer-Aided Design 38, 1134–1144 (2006)
11. Schneider, P.J., Eberly, D.H.: Geometric Tools for Computer Graphics. Morgan Kaufmann, San Francisco (2003)
12. Shimada, K., Gossard, D.C.: Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis. Computer Aided Geometric Design 15, 199–222 (1998)
13. Sugihara, K., Inagaki, H.: Why is the 3D Delaunay triangulation difficult to construct? Information Processing Letters 54, 275–280 (1995)
14. Wang, C.C.L., Smith, S.S.-F., Yuen, M.M.F.: Surface flattening based on energy model. Computer-Aided Design 34, 823–833 (2002)

On Mesh Editing, Manifold Learning, and Diffusion Wavelets

R.M. Rustamov

Drew University, USA
rrustamov@drew.edu

Abstract. We spell out a formal equivalence between the naive Laplacian editing and semi-supervised learning by bi-Laplacian Regularized Least Squares. This allows us to write the solution to Laplacian mesh editing in a 'closed' form, based on which we introduce the Generalized Linear Editing (GLE). GLE has both naive Laplacian editing and gradient based editing as special cases. GLE allows using diffusion wavelets for mesh editing. We present preliminary experiments, and shortly discuss connections to segmentation.

1 Introduction

A remarkable similarity exists between semi-supervised manifold learning and mesh editing: both seek to extrapolate data attached at some points to the whole manifold.

Given a set of labeled samples, extrapolating labels throughout the entire sample space is the task of semi-supervised learning. The qualification "manifold" is added if the samples are assumed to belong to a manifold embedded into a high-dimensional space – attaching labels is equivalent to defining a function on this manifold.

Editing a mesh involves determining the new locations of vertices given new locations of some of the vertices – the handles. The displacement vectors – the differences between new and old vertex positions – can be considered to define a function on the mesh. Thus, given the values of this function at the handles we are trying to extrapolate to the whole mesh – a task that would otherwise qualify as semi-supervised learning. If rotations at handles are also given, propagating them throughout the mesh is again an instance of semi-supervised learning.

Does this similarity of the two fields extend beyond the objectives sought? Laplacian based approaches to mesh editing start by extracting the surface's *differential coordinates*, and then reconstruct the surface by imposing the handle constraints and requiring that the differential coordinates are preserved as much as possible. The differential coordinates capture the local detail, so the more they are preserved, the more the shape is preserved. When viewed from this angle, Laplacian mesh editing seems to bear no resemblance to the methods of semi-supervised learning.

We show, however, that there exists a formal equivalence between the naive (linear) Laplacian editing and semi-supervised learning by bi-Laplacian Regularized Least Squares (Section 4). This allows us to write the solution to Laplacian mesh editing in a "closed" form (Section 5). Based on this closed form we introduce the **G**eneralized **L**inear **E**diting (GLE) which has both naive Laplacian editing and gradient based editing as special cases. GLE allows the use of diffusion wavelets for mesh editing (Section 6). Preliminary experiments are presented (Section 7), and connections to segmentation are discussed (Section 8).

Our contributions are threefold: clearly spelling out the formal relationship between manifold learning and mesh editing, introducing GLE, and using diffusion wavelets for mesh editing. Although naive Laplacian editing is not practically useful, it becomes so when embedded into nonlinear schemes such as [1]. Therefore, understanding the properties of the naive Laplacian editing better, and investigating its potential improvements may have considerable consequences on the state of art.

2 Related Work

Immense attention has been received by direct surface manipulation methods based on differential representations. The first examples were Poisson mesh editing [2] and Laplacian coordinates [3,4]. As we learn from [5], it was gradient domain image manipulation that inspired these approaches: specifically, the Poisson surface editing of [6] motivated Poisson mesh editing. These initial methods are now customarily referred as naive or linear – they optimize each of x, y, z coordinates separately, and so do not allow dealing with handle rotations. An excellent survey of these and other linear techniques is [5].

Remarkably, differential representations come with an elegant interpretation of being local surface descriptors. For example, Laplacian coordinates are the components of the mean-curvature normal. This interpretation offers a strong intuition about how to deal with rotations, and led to new techniques a few examples of which are [7,8,9]; a great survey is [10]. The interpretation also inspired the development of other intrinsic coordinates: pyramid coordinates [11], and rotation invariant coordinates [12].

In the light of all these developments, one might legitimately the need to study any further the linear methods, generalize or alter them. The answer is best given by an example: in a recent paper [1], the naive Laplacian editing is embedded into an iterative scheme to obtain a non-linear method. The algorithm is guaranteed to converge and is remarkably easy to implement. The method compares very favorably with PriMo [13], a state-of-the-art non-linear technique. This makes us believe that understanding the properties of the naive methods better, and investigating their potential improvements will have considerable consequences on the state of the art.

To the best of our knowledge, the connections to the semi-supervised learning have never emerged, perhaps due to the sheer beauty of the interpretation that differential coordinates came with. Yet we must mention the inspiring study of Lévy [14], where among other things he explains how Laplace-Beltrami eigenfunctions can be used to manipulate shapes. He proposes pose transfer by exchanging expansion coefficients of the coordinate functions in terms of Laplace-Beltrami eigenfunctions. In addition, in [15] geometry filtering is performed by modifying the expansion coefficients. Another work that uses the Laplace-Beltrami expansion coefficients is [16], where the size of linear systems associated with Laplacian editing are significantly reduced to achieve interactive computational speed for manipulating large meshes. All of these works can be interpreted as an approach to editing where the control is vested into these expansion coefficients, yet they discuss the connections neither to manifold learning nor to Laplacian editing. In a different context, let us also mention the paper [17] which studies mesh smoothing in the light of Laplacian eigenbasis and regularization.

Using different function bases to manipulate meshes would not be surprising to the space deformation community. Just to mention a few, in [18,19,20,21] Radial Basis Functions (RBF) are used to infer the space deformation that would satisfy given constraints. There are many bases to choose from and, thus, flexibility to decide based on time/quality constraints. However, to the best of our knowledge, the idea to use different bases to generalize/modify a direct manipulation method such as Laplacian editing is novel. Let us add that diffusion wavelets have found only very limited use in digital geometry processing – the only work that we are aware of is [22] where diffusion wavelets are used for mesh compression.

As for manifold learning, we can only mention some key papers that are directly relevant. Laplacian based regularization was introduced in [23], where semi-supervised learning is reduced to minimization of an expression which contains a penalty term to enforce the labels of labeled samples, and terms to ensure "continuity" of labeling – "close" points get similar labels. The latter are called the *regularization* terms; they ensure that learning results in a sufficiently smooth function. As shown in [23], the Laplacian – the manifold's Laplace-Beltrami operator – makes a good regularization term for a variety of learning applications. Diffusion maps, diffusion wavelets, diffusion wavelet packets and other related concepts were introduced by R. Coifman and coworkers, and the definitive reference is the special issue [24].

3 Manifold Learning

We will avoid describing the most general setting for manifold learning, rather our exposition will be geared towards surfaces in space – we will make assumptions and introduce notations most appropriate for our purposes. We will concentrate on versions of Laplacian regularized least squares learning, a concept introduced in [23], heavily borrowing from it and from paper [25] with some notational modifications.

We start with a connected surface $\mathcal{S}$. Some of the points on the surface, say $\{\mathbf{p}_i\}_{i=1}^{l}$ have been labeled – real numbers $\{d_i\}_{i=1}^{l}$ have been assigned to them. Semi-supervised learning seeks a function f^* that minimizes

$$\sum_{i=1}^{l}(f(\mathbf{p}_i)-d_i)^2+\beta\|f\|_E^2+\gamma\|f\|_I^2. \tag{1}$$

The first term tries to enforce the function to take values prescribed at labeled vertices. The last two terms are called regularization terms, and they aim to make the function smooth in *extrinsic* and *intrinsic* senses. To clarify, extrinsic smoothness could mean that function takes close values at points that are close in Euclidean space; intrinsic smoothness would mean close function values for points that are geodetically close. We will be mostly interested in the first and last terms, dropping the middle terms of (1) in what follows.

Belkin et. al. [23] propose to use the surface integral of function's gradient as the intrinsic regularization term, which can be rewritten using Green's theorem as

$$\|f\|_I^2=\int_{\mathcal{S}} f\Delta f,$$

where Δ is the Laplace-Beltrami operator of the surface. They also note that iterated Laplacians Δ^k and linear combinations of these can be used in this expression to yield other examples of intrinsic regularization terms; thus for a self-adjoint linear differential operator L on the surface, we have the regularization term

$$\|f\|_I^2=\int_{\mathcal{S}} fLf.$$

The natural realm for discussing the minimization problem (1) is an appropriately chosen Reproducing Kernel Hilbert Space. We avoid these details, and only point out to a few consequences. Let us denote by μ_i and $e_i:\mathcal{S}\to R$ the eigenvalues and the eigenfunctions of the linear operator L. These satisfy $Le_i=\mu_i e_i$. Since L is self-adjoint, the eigenfunctions constitute an orthogonal basis for $L_2(\mathcal{S})$. For a function $f=\sum_i c_i e_i$, the intrinsic regularizator easily evaluates to

$$\|f\|_I^2=\sum_i \mu_i c_i^2.$$

Now, the *kernel function*

$$K(\mathbf{p},\mathbf{q})=\sum_i \frac{e_i(\mathbf{p})e_i(\mathbf{q})}{\mu_i}$$

can be defined (note the similarity with the formula for matrix pseudoinverses); the important point is that the solution of the semi-supervised learning problem (1) can be written as

$$f^*(\mathbf{q})=\sum_{i=1}^{l} a_i K(\mathbf{p}_i,\mathbf{q}).$$

Notice that the summation is over the labeled points' indices. Numbers a_i are the entries of the vector $\boldsymbol{a}$ which solves the equation

$$(\gamma I_l+K')\boldsymbol{a}=\boldsymbol{d}. \tag{2}$$

Here I_l is the $l \times l$ identity matrix, K' is an $l \times l$ matrix with $K'_{ij} = K(\mathbf{p}_i, \mathbf{p}_j)$, and $\boldsymbol{d}$ is the vector whose i-th entry is the label d_i.

4 Laplacian Editing Reinterpreted

Given a surface mesh, mesh editing allows a user to modify it by specifying new positions for some surface points; these user constrained points are called *handles.* Laplacian mesh editing is based on extracting the surface's *differential coordinates*, and then reconstructing the surface by imposing the user constraints and requiring that the differential coordinates are preserved as much as possible; for a recent survey we refer the reader to [5]. We will show that the naive or linear version of Laplacian editing, the version where handle rotations are not propogated, is equivalent to semi-supervised learning with the bi-Laplacian as the regularizator.

Consider projection onto x-coordinate$\pi_x : \mathcal{S} \to R$, i.e. the function whose value $\pi_x(\mathbf{p})$ is the x-coordinate of the surface point $\mathbf{p}$. This function, will be assumed to satisfy $\int_{\mathcal{S}} \pi_x = 0$, which in all cases can be achieved by translating the origin to the center of mass of the surface. The differential coordinate $\delta_x(\mathbf{p})$ is defined by

$$\delta_x(\mathbf{p}) = \Delta \pi_x(\mathbf{p}),$$

where Δ is the Laplace-Beltrami operator of the surface. In a similar way one defines $\delta_y(\mathbf{p})$ and $\delta_z(\mathbf{p})$. As an aside, the differential coordinates are precisely the components of the mean curvature normal.

The edited surface $\mathcal{S}'$ is in one to one correspondence with the original surface, so we will use the same letter to denote both a point on $\mathcal{S}$ and $\mathcal{S}'$. Also, $\mathcal{S}'$ has its own function π'_x. Suppose that the handles are the points $\{\mathbf{p}_i\}_{i=1}^l$, and the user has specified new x-coordinates for these points as $\{x'_i\}_{i=1}^l$. Soft constraint Laplacian mesh editing constructs the new surface by minimizing the expression

$$\sum_{i=1}^{l} (\pi'_x(\mathbf{p}_i) - x'_i)^2 + \gamma \int_{\mathcal{S}} (\Delta \pi'_x - \delta_x)^2.$$

We are able to treat the x-coordinate separately, since in Laplacian editing mesh coordinates are treated independently.

Now we will rewrite the problem. Let us introduce the function $f = \pi'_x - \pi_x$, the difference between the old and new x-coordinates. Along with the corresponding functions for the other two coordinates, this function completely determines the new surface in terms of the old. At the handle points $\mathbf{p}_i$ the user has specified the sought values of f, which we denote by $d_i = x'_i - \pi_x(\mathbf{p}_i)$. Note that

$$\Delta \pi'_x - \delta_x = \Delta \pi'_x - \Delta \pi_x = \Delta f.$$

The equality

$$\int_{\mathcal{S}} (\Delta f)^2 = \int_{\mathcal{S}} f \Delta^2 f$$

follows from Green's theorem; this changes our optimization problem into

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \gamma \int_{\mathcal{S}} f \Delta^2 f.$$

This is precisely equation (1) of semi-supervised learning with $L = \Delta^2$. Notice that, the same derivation is valid for surfaces with boundary, if f assumed to satisfy the Neumann boundary condition. With little thought the reader can see that this assumption is implicit in Laplacian editing.

In a similar vein, let us show that gradient based editing of [2] is equivalent to semi-supervised learning with Laplacian as regularizator; this is precisely Laplacian Regularized Least Squares presented and studied in [23]. In fact, gradient based editing seeks to minimize the expression

$$\sum_{i=1}^{l} (\pi'_x(\mathbf{p}_i) - x'_i)^2 + \gamma \int_{\mathcal{S}} (\nabla \pi'_x - \nabla \pi_x)^2,$$

where ∇ is the gradient. In other words, the reconstruction aims to preserve the gradients of mesh coordinate functions – "gradient coordinates" $\nabla \pi_x$. Using our notation and applying Green's theorem this can be rewritten as

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \gamma \int_{\mathcal{S}} f \Delta f.$$

Clearly, this is the semi-supervised learning objective with $L = \Delta$.

This proves our claim that *formally*, Laplacian mesh editing is an instance of semi-supervised manifold learning. The formulas make clear that the preservation of differential coordinates is equivalent to intrinsic regularization. This equivalence has an intuitive appeal: regularization forces the displacement function f to be "smooth", this in turn allows to protect the local surface detail – the original goal behind the preservation of differential coordinates.

5 Consequences

After establishing the relationship to manifold learning, we can start importing knowledge from the field. Let us discuss two such examples – they will motivate a generalization of Laplacian and gradient editing.

First, we can write an explicit formula for the solution of the Laplacian and gradient editing. In fact, consider the eigenvalues λ_i and eigenfunctions ϕ_i of the Laplace-Beltrami operator. The eigenvalues are non-negative and constitute a discrete set; we put them into non-decreasing order

$$\lambda_0 = 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_i < \ldots$$

Note that $\lambda = 0$ is always a simple eigenvalue because the surface is assumed to be connected. The appropriately normalized eigenfunction corresponding to λ_i will be denoted by ϕ_i. Notice that the bi-Laplacian has the same eigenfunctions ϕ_i, with corresponding eigenvalues λ_i^2.

Now the kernel function corresponding to Laplacian editing is given by

$$K(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{\infty} \frac{\phi_i(\mathbf{p})\phi_i(\mathbf{q})}{\lambda_i^2}$$

and the solution to the mesh editing problem is

$$f^*(\mathbf{q}) = \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q}),$$

where a_i are found by solving the system (2). A similar formula is valid for gradient based editing – one replaces λ_i^2 by λ_i in the definition of the kernel function.

Second, there is an interpretation of regularized manifold learning schemes which translates into a remarkable reading of Laplacian editing. To explain, we follow [25] and write the minimization of bi-Laplacian regularized least squares in terms of eigenvalues and eigenfunctions of the Laplace-Beltrami operator.

Remembering that the eigenfunctions ϕ_i constitute a basis for $L_2(\mathcal{S})$, we can expand any function as $f = \sum_i c_i \phi_i$. When this expansion is plugged into the bi-Laplacian regularizator in the objective function, the expression to minimize reduces to

$$\sum_{i=1}^{l} (f(\mathbf{p}_i) - d_i)^2 + \gamma \sum_i \lambda_i^2 c_i^2, \tag{3}$$

while a similar formula for Laplacian based regularization would contain λ_i instead of λ_i^2. Thereby, we are trying to device a function by combining ϕ_i so that this combination takes prescribed values in the least squares sense, but we harshly penalize the use of high frequency eigenfunctions – eigenfunctions corresponding to larger eigenvalues.

Thus, in some sense, Laplacian based semi-supervised learning assumes that Laplacian eigenfunctions constitute the preferred "learning basis". Preferred, because they provide the smoothest basis for $L_2(\mathcal{S})$ in the sense explained in [25]. Higher penalty for high frequency eigenfunctions makes sure that the solution is smooth enough. This fits very well with Occam's razor – lower frequency eigenfunctions are "simpler". For example, under general conditions, the first eigenfunction ϕ_1 changes its sign only once, and has only one minimum and one maximum.

In the context of mesh editing, we should perhaps talk about the "motion basis" of an object. The motion basis would contain (linearly independent) displacement functions for natural articulations of the object, ordered from the simplest to the most complex. One could device a measure of complexity for linear combinations of motion basis functions. The objective of mesh editing would be to find the simplest such combination that satisfies the user constraints.

From this perspective, the fundamental assumption of Laplacian editing is that Laplacian eigenfunctions constitute a good motion basis. Clearly, the difference between Laplacian and gradient based editing is in the penalty for using high frequency eigenfunctions – Laplacian editing is harsher in this respect. Of course, using higher iterated Laplacians in the regularizator further increases the penalty for using high frequencies. An important question emerges – do we have to base our regularization on Laplacian or its iterates? *Can we set the penalties manually for each eigenfunction?*

Clearly such a modification of Laplacian editing, namely choosing the penalties manually, can be obtained by simply changing the formula for the kernel into

$$K(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{\infty} \frac{\phi_i(\mathbf{p})\phi_i(\mathbf{q})}{\mu_i}.$$

We speculate that if the user agrees to set $\mu_i = \infty$ for all $i > r$, then this formula can be efficiently implemented to allow the user to adjust the remaining penalty weights μ_i interactively. Setting infinite penalties is equivalent to truncating the sum in the formula for the kernel function – effectively keeping only r low frequency eigenfunctions.

Remember that in discrete setting, the Laplacian and the kernel function are $n \times n$ matrices, where n is the number of mesh vertices in the region of interest. We will assume that one computes r eigenvectors in the preprocessing step. Of course, computing the eigenvectors of a matrix is a considerable burden; yet if r is in the range of a few hundreds, that many eigenvectors of the Laplacian for a mesh as large as 250K vertices can be evaluated in about 15-20 minutes [15]. During the interactive session, editing will involve solving an $l \times l$ linear system ([2]), where l is the number of handles, and evaluating the linear combination of l columns of the kernel matrix – the columns associated with the handle vertices. Of course, we should not pre-compute and store the whole $n \times n$ kernel matrix in the main memory. Instead, storing only the r eigenvectors will reduce the space requirement from $O(n^2)$ to $O(nr)$. Now notice that the time complexity of the interactive part – obtaining the new vertex coordinates after the user modifies the penalty weights – is linear in the number of vertices, $O(nl + nr)$. Also, the new coordinates can be evaluated in parallel. Let us emphasize that we have not implemented this approach, and we do not know how the omission of high frequency eigenvectors will influence the result of editing.

6 GLE and Diffusion Wavelets

Once again, do we have to base our regularization on Laplacian or its iterates? *Can we use a different basis?* These questions lead us to propose the following Generalized Linear Editing, GLE for short. Similar to naive Laplacian editing, GLE does not handle frame rotations, and the displacements of each coordinate are evaluated independently. Thus, for brevity, we concentrate on obtaining the x coordinates only; the other two coordinates require the same steps, perhaps with different bases and weights if wanted. At the cost of self-repetition, let us clearly outline the algorithm:

1. Pick an orthonormal basis e_i of $L_2(\mathcal{S})$; here $\mathcal{S}$ is the edited surface.
2. Assign penalty weights μ_i to each of the basis functions e_i.
3. Define $K(\mathbf{p},\mathbf{q}) = \sum_i e_i(\mathbf{p})e_i(\mathbf{q})/\mu_i$.
4. Let l handle vertices $\mathbf{p}_i, i = 1, ..., l$ and their x-coordinate displacements $d_i, i = 1, ..., l$ be given. Construct vector $\boldsymbol{d}$ whose ith entry is d_i. Solve the equation
$$(\gamma I_l + K')\boldsymbol{a} = \boldsymbol{d}$$
for $\boldsymbol{a}$, where $K'_{ij} = K(\mathbf{p}_i, \mathbf{p}_j)$ is an $l \times l$ matrix. Here γ is a user set positive parameter – the larger is γ the less is pressure to satisfy the displacement constraints, the more weight is given to the regularization term.
5. Compute the new x coordinate $\mathbf{q}'_x$ of the point $\mathbf{q}$ using the formula $\mathbf{q}'_x = \mathbf{q}_x + \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q})$.

When dealing with some region of interest instead of the whole mesh, one will require the boundary to stay constant. This can be achieved in GLE by using basis functions that are supported on the region of interest.

One may also wish to enforce the handle constraints with different weights. Suppose that weights w_i are associated with handle vertices $\mathbf{p}_i, i = 1, ..., l$. One then can construct the $l \times l$ diagonal matrix W with $W_{ii} = 1/w_i$; now in step 4, one replaces the equation for $\boldsymbol{a}$ by $(\gamma W + K')\boldsymbol{a} = \boldsymbol{d}$.

Notice that both the Laplacian and gradient based mesh editing are examples of GLE. To provide further examples, let us use *diffusion wavelets* [26] as a basis. The following discussion will be geared towards implementation, providing the discrete counterparts of the steps above.

Let P be a set of points on a mesh. Clearly, any real-valued function on P can be recorded as a vector in $R^{|P|}$. Then, a diffusion operator T (that linearly acts on these vectors) is a matrix that satisfies certain properties, most notably its eigenvalues are in the $[0, 1]$ interval. In the implementation of diffusion wavelets that we worked with [27], among a few choices for T we used the following: vertices of the mesh are considered as a point cloud, and for each point, one finds a fixed number of nearest neighbors and assigns the corresponding matrix entry to be $\exp(-\Delta * dist^2)$, where $dist$ is the distance to the point in question. This choice on our part was forced by the fact that the diffusion wavelet code was written by the manifold learning community and was geared toward working with point clouds.

Diffusion wavelet construction uses the diadic powers of T to construct subspaces V_i and W_i of $R^{|P|}$ together with their orthogonal bases; the subscript of a subspace is called its *level.* These subspaces satisfy $R^{|P|} = W_0 \oplus V_0$, $V_0 = W_1 \oplus V_1$, $V_1 = W_2 \oplus V_2$, and so on. In fact, in the implementation we worked with, $W_0 = \{\mathbf{0}\}$, and $V_0 = R^{|P|}$ with delta-function basis. To obtain the basis of subspace V_{i+1}, one acts by the diadic power of the diffusion operator T^{2^i} on the basis of V_i, and applies some thresholding and a special locality preserving orthogonalization. An important property of these bases is that different levels represent different *scales* – together they provide a multi-scale basis of $R^{|P|}$. For example, the basis vectors in V_0 represent the smallest scale possible – their support is just one vertex. Meanwhile, the basis vectors of subspaces of high level have global nature, because they are obtained by many repetitive applications of the diffusion operator – in a sense, the function values are diffused throughout the mesh.

Fixing some maximum level m that one desires to work with, one can write

$$R^{|P|} = W_0 \oplus W_1 \oplus W_2 \oplus \cdots \oplus W_m \oplus V_m.$$

In addition, gathering the bases of the subspaces appearing in this equality, one obtains an orthogonal basis of $R^{|P|}$. It is this basis that we will use for editing. Let us construct the $|P| \times |P|$ matrix E which contains the basis vectors as its columns. Next, we need penalty weights $\mu_k, k = 1, ..., |P|$. Since functions in higher levels are smoother, we have made a design decision to set $\mu_k = 1/l^{\alpha}$, where l is the level from which the basis vector $e_k$comes from, and α is a positive user set parameter: the larger it is the more high-level functions are preferred. Very similar results are obtained by setting $\mu_k = 1/\beta^l$, with $\beta > 1$.

Consider the matrix M which has $M_{kk} = 1/\mu_k$, and has zeros everywhere else. Notice that the discrete counterpart of the kernel function $K(\mathbf{p}, \mathbf{q})$ is a $|P| \times |P|$ matrix which we denote by K as well; clearly, $K = EME^T$. Implementing the rest of the steps should be straightforward.

7 Results and Discussion

We provide preliminary results of GLE using diffusion wavelets. Our starting surface is the unit sphere; the mesh is obtained through platonic subdivision using the trimesh2 library. The number of vertices is $|P| = 482$. We choose $m = 12$ because V_{12} is one-dimensional, and so no more basis vectors coming from W_i's are produced. Choosing a smaller value of m will lead to less smooth edits, because smoother basis functions belong to higher levels. Figure 1 depicts the result of applying the following editing constraints: the south pole stays the same, the north pole's z-coordinate increases by 1. For comparison, the result of Laplacian editing using cotangent weights is shown in the same figure.

Diffusion wavelets work with more complicated constraints as well. The constraints for Figure 2 are as follows: the vertices with $|z| < 0.1$ are moved closer to the origin by setting $x \rightarrow 0.5x$, $y \rightarrow 0.5y$, and the vertices with $|z| > 0.7$ are kept same.

Our preliminary implementation was done in MATLAB, and it uses the diffusion wavelet code provided on Mauro Maggioni's website [27]. To give an idea about timing: computation of diffusion wavelet basis for $|P| = 482$ took about 9 seconds, and editing took about 0.5 seconds. However, if the number of vertices is doubled, the time spent to compute the diffusion wavelet basis increases to about 90 seconds, while edit time goes up to 6 seconds. The experiments were run on T7200 @ 2.0GHz laptop with 1GB main memory, running Windows XP Professional.

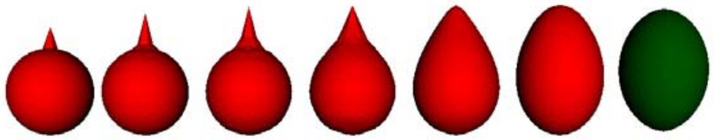


Fig. 1. Diffusion wavelet deformations of the sphere are in red. The parameters are as follows: $m = 12$, and $\alpha = 0, 1, 2, 4, 8, 16$ in respective order. The result of Laplacian editing with the same constraints is given last in green.

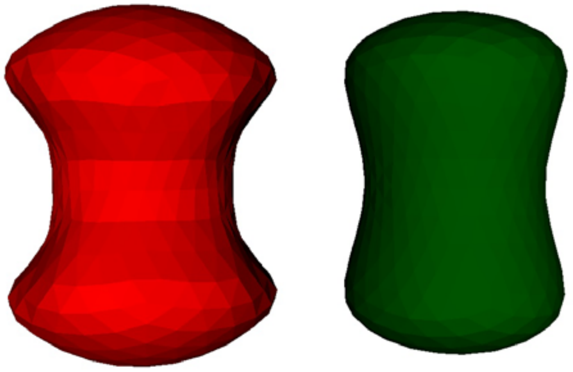


Fig. 2. Diffusion wavelet deformation of the sphere is in red; $m = 12$ and $\alpha = 16$ were used. The result of Laplacian editing with the same constraints is in green.

7.1 Discussion

First, going back to the experiment of Figure 1, notice that by increasing α we decrease the penalty of using high level basis vectors, this effectively suppresses the use of lower level vectors, and results in smoother, more global edits. Thus, the choice of this parameter can be tuned by whether less smooth local or more smooth global edits are wanted. Such scale control in Laplacian editing can only be achieved by varying the region of interest. This is due to the global nature of Laplacian eigenfunctions – their support is the whole mesh. This results in a “butterfly effect” – one may intend to make a little bump on the sphere, but would end up with an ellipsoid.

Second, the reader would have noted our use of uniform sphere sampling. With the implementation of diffusion wavelets that we used, we feel that, at least theoretically, uniformity is necessary. Indeed, the computed diffusion wavelet basis vectors are orthogonal with respect to the standard inner product on $R^{|P|}$ – an inner product that does not correspond to anything geometrically meaningful unless the mesh vertices are distributed uniformly over the area of the surface. However, in practice non-uniform sampling leads to mixed results. For example Figure 3 a) shows a very satisfying result of diffusion wavelet editing on a sphere

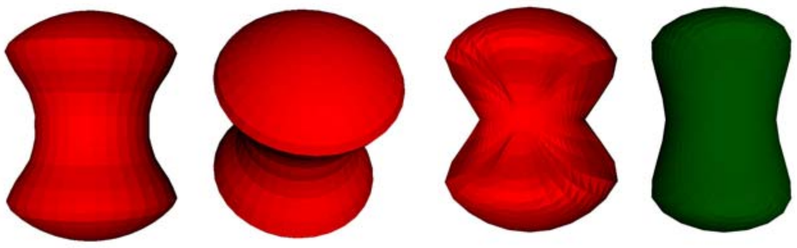


Fig. 3. a) Diffusion wavelet deformation of longitude-latitude triangulated sphere. The parameter values $m = 12$ and $\alpha = 16$ were used. Same constraints as for Figure 2. b) Same as in a) except that the "squishing" circle passes through the poles. The result of Laplacian editing with this constraint is in green.

mesh obtained using longitude-latitude triangulation. Figure 3 b) shows what happens if constraints include both of the poles of this non-uniformly sampled sphere. It is certainly not very satisfying when compared to Laplacian editing.

Let us conclude by noting that one could in principle circumvent this issue by first sampling a uniform point cloud on the surface, evaluating the diffusion wavelets, and then interpolating their values from the point cloud to the mesh vertices.

8 Connections to Segmentation

Motion based segmentation. [28,29,30] inputs an animation sequence of a boundary mesh, and clusters together points that move in accord to produce a segmentation. Now one can imagine feeding to such an algorithm "animations" generated by applying Laplacian editing or GLE to the mesh – this would give a segmentation associated with a given mesh editing approach.

Do we need to explicitly produce these animations in order to get the associated segmentation? Looking at the solution of GLE,

$$f^*(\mathbf{q}) = \sum_{i=1}^{l} a_i K(\mathbf{p}_i, \mathbf{q}),$$

we notice that the value of $K(\mathbf{p}_i, \mathbf{q})$ measures how much point $\mathbf{q}$ is influenced by editing the handle at $\mathbf{p}_i$. Consequently, within the GLE framework, the magnitude of the kernel function $K(\mathbf{p}, \mathbf{q})$ is a measure of how much the points $\mathbf{p}$ and $\mathbf{q}$ move in accord – making the kernel function a natural measure of similarity for motion based segmentation. This is not a new approach to segmentation: for example, [31] spells out similar ideas and uses the Green's function of Laplace-Beltrami operator – the kernel function of gradient-based editing – to obtain pose-invariant segmentation of meshes.

The manifold learning perspective on this association of editing and segmentation is interesting. Indeed, there is a similarity between mesh segmentation and unsupervised manifold learning. In mesh segmentation the objective is to assign a discrete set of labels, segment identifiers, to the mesh vertices. This is similar to defining an integer valued function on a point cloud in a meaningful way – an instance of unsupervised manifold learning. But for each supervised method based on regularization, one can device a corresponding unsupervised method by simply using the same regularization term. In this case one would search for a function $f : \mathcal{S} \to R$ satisfying appropriate conditions – such as being a fuzzy membership function that clusters points into even clusters – that minimizes $\|f\|_I^2$. Thus, it is not surprising to have a segmentation scheme canonically associated with an editing method.

We conclude by noticing that the connection between mesh editing and segmentation can be helpful when designing bases for GLE, assuming that a basis that produces better segmentations should result in better editing.

9 Summary and Future Work

We have stated the formal relationship between manifold learning and naive Laplacian and gradient-based editing. As a result, we were able to introduce a generalized editing approach that allows the use of diffusion wavelets for mesh editing. We believe that diffusion wavelets are a very promising alternative to Laplacian eigenfunctions because due to their multi-scale nature. More experiments will be required to verify this; including experiments where handle rotations are propagated either using the existing approaches or through some new approach based once more on diffusion wavelets. Replacing the naive Laplacian editing by GLE in iterative methods such as [1] would provide another venue for experimentation. Another very important direction for future research is devising diffusion wavelets for triangle meshes rather than point clouds and resolving the issue of non-uniform sampling. Faster algorithms to evaluate such wavelets would be indispensable for practical applications.

On the intriguing side, we would like to investigate whether the equivalence extends to non-linear mesh editing approaches as well, and if so, to exploit these equivalences in the opposite direction – see if the insights gained from mesh editing can help in the realm of manifold learning.

References

1. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 109–116 (2007)
2. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with Poisson-based gradient field manipulation. In: TOG(SIGGRAPH), 644–651 (2004)
3. Alexa, M.: Differential coordinates for local mesh morphing and deformation. The Visual Computer 19(2-3), 105–114 (2003)

4. Sorkine, O., Cohen-Or, D., Toledo, S.: High-pass quantization for mesh encoding. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, Eurographics Association, pp. 42–51 (2003)
5. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Transactions on Visualization and Computer Graphics (2007) (to appear)
6. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. TOG(SIGGRAPH) 22(3), 313–318 (2003)
7. Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 179–188. ACM Press, New York (2004)
8. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. Comput. Graph. Forum 24(3), 601–609 (2005)
9. Lipman, Y., Cohen-Or, D., Gal, R., Levin, D.: Volume and shape preservation via moving frame manipulation. ACM Trans. Graph. 26(1), 5 (2007)
10. Sorkine, O.: Differential representations for mesh processing. Computer Graphics Forum 25(4), 789–807 (2006)
11. Sheffer, A., Kraevoy, V.: Pyramid coordinates for morphing and deformation. In: 3DPVT 2004: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, Washington, DC, USA, pp. 68–75. IEEE Computer Society, Los Alamitos (2004)
12. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. 24(3), 479–487 (2005)
13. Botsch, M., Pauly, M., Gross, M., Kobbelt, L.: Primo: coupled prisms for intuitive surface modeling. In: SGP 2006: Proceedings of the fourth Eurographics symposium on Geometry processing, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, pp. 11–20 (2006)
14. Lévy, B.: Laplace-Beltrami eigenfunctions: Towards an algorithm that understands geometry. In: Shape Modeling International (2006)
15. Vallet, B., Lévy, B.: Spectral geometry processing with manifold harmonics. Computer Graphics Forum (Proceedings Eurographics) (2008)
16. Rong, G., Cao, Y., Guo, X.: Spectral mesh deformation. Vis. Comput. 24(7), 787–796 (2008)
17. Volodine, T., Vanderstraeten, D., Roose, D.: Smoothing of meshes and point clouds using weighted geometry-aware bases. In: Kim, M.-S., Shimada, K. (eds.) GMP 2006. LNCS, vol. 4077, pp. 687–693. Springer, Heidelberg (2006)
18. Borrel, P., Rappoport, A.: Simple constrained deformations for geometric modeling and interactive design. ACM Trans. Graph. 13(2), 137–155 (1994)
19. Turk, G., O'Brien, J.F.: Modelling with implicit surfaces that interpolate. In: SIGGRAPH 2005: ACM SIGGRAPH 2005 Courses, p. 21. ACM, New York (2005)
20. Reuter, P., Tobor, I., Schlick, C., Dedieu, S.: Point-based modelling and rendering using radial basis functions. In: GRAPHITE 2003: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, pp. 111–118. ACM, New York (2003)
21. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Comput. Graph. Forum 24(3), 611–621 (2005)
22. Mahadevan, S.: Adaptive mesh compression in 3D computer graphics using multiscale manifold learning. In: ICML 2007: Proceedings of the 24th international conference on Machine learning, pp. 585–592. ACM, New York (2007)
23. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. 7, 2399–2434 (2006)

24. Coifman, R.R.: Special issue on diffusion maps. Appl. Comput. Harmon. Anal. 21(1), 3 (2006)
25. Belkin, M., Niyogi, P.: Semi-supervised learning on riemannian manifolds. Mach. Learn. 56(1-3), 209–239 (2004)
26. Coifman, R.R., Maggioni, M.: Diffusion wavelets. Appl. Comput. Harmon. Anal. 21(1), 53–94 (2006)
27. Maggioni, M.: Diffusion wavelet code, http://www.math.duke.edu/~mauro/DWCode.zip
28. Lee, T.Y., Lin, P.H., Yan, S.U., Lin, C.H.: Mesh decomposition using motion information from animation sequences: Animating geometrical models. Comput. Animat. Virtual Worlds 16(3-4), 519–529 (2005)
29. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 209–217. ACM Press, New York (2005)
30. Günther, J., Friedrich, H., Wald, I., Seidel, H.P., Slusallek, P.: Ray tracing animated scenes using motion decomposition. Computer Graphics Forum 25(3), 517–525 (2006) (Proceedings of Eurographics)
31. Rustamov, R.M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In: Symposium on Geometry Processing (2007)

Gradient Approximation on Uniform Meshes by Finite Differences and Cubic Spline Interpolation

P. Sablonnière

Centre de Mathématiques, INSA de Rennes, France
psablonn@insa-rennes.fr

Abstract. For the approximation of gradients from data values at vertices of a uniform grid, we compare two methods based on cubic spline interpolation with a classical method based on finite differences. For univariate cubic splines, we use the so-called de Boor's Not a Knot property and a new method giving pretty good slopes. Then these methods are used on parallels to the axes for estimating gradients on bivariate grids. They are illustrated by several numerical examples.

Keywords: gradient approximation, cubic splines.

1 Introduction

Let $\Omega := [a,b] \times [c,d]$ be a rectangular domain endowed with the tensor-product partition $\mathcal{Z}_{m,n} := \mathcal{X}_m \otimes \mathcal{Y}_n$ where $\mathcal{X}_m := \{x_i = a + ih, 0 \leq i \leq m\}$ and $\mathcal{Y}_n := \{y_j = c + jk, 0 \leq j \leq n\}$ are uniform partitions with steplength h and k on $[a,b]$ and $[c,d]$ respectively.

Given a set of data values $z_{i,j} := f(M_{i,j})$ at gridpoints $M_{i,j} := (x_i, y_j)$, where f is a (known or unknown) function, we want to compute good approximations of the gradients $g_{i,j} := (\partial_1 f(M_{i,j}), \partial_2 f(M_{i,j}))$, with $\partial_1 f := \frac{\partial f}{\partial x}$ and $\partial_2 f := \frac{\partial f}{\partial y}$. They can be used e.g. for the computation of the area of the associated surface. We first compare two methods of approximation of first derivatives, based on univariate interpolation by C^2 cubic splines, with classical approximations by finite differences (FD). The first comes from the so-called de Boor's Not a Knot interpolating spline (abbr. NAK, see [3]). The second is a new method based on a better approximation of end derivatives, giving pretty good slopes (abbr. PGS). Moreover, it has been recently proved by the author that the FD method comes from a specific cubic spline quasi-interpolant. Then, we use these methods on parallels to the axes for estimating gradients on bivariate grids.

Here is a brief outline of the paper: in Section 2, we recall the classical finite difference formulas giving fourth order approximations of derivatives. In Section 3.1, we recall the computation of derivatives using the de Boor's NAK condition which only gives an overall third order approximation of these derivatives. In Section 3.2, we present our new PGS method which is based on a systematic fourth order approximation of derivatives. For both methods, we develop the full Cholesky decompositions which are particularly simple and lead to efficient and fast algorithms. Section 4 presents univariate numerical results illustrating the previous methods. Finally, in Section 5, we extend the univariate results to the bivariate grid described above. In that case, partial derivatives are estimated on parallels to the main axes. We present several numerical results illustrating the three methods. They show that our PGS method is very satisfying and overall slightly better than the two others.

There are, of course, many other ways of approximating gradients on uniform grids. In particular, one could also expect rather good approximations of gradients by using C^1 quadratic spline quasi-interpolants based either on criss-cross triangulations, as in [4][6], or on uniform Powell-Sabin triangulations, as in [5]. We plan to do a comparison with the methods presented in this paper. For the estimation of gradients from scattered data, see e.g. [11].

2 Approximation of Derivatives by Finite Differences

Classical approximations of the first derivatives $y_i' := \varphi'(x_i)$, $0 \le i \le m$, of a univariate function φ defined on $[a,b]$ by its values $y_i := \varphi(x_i)$, are given by finite differences (abbr. FD). As those given in Section 3 by cubic splines are in $O(h^4)$, we take the following, which is exact on $\mathbb{P}_4$ (the space of polynomials of degree at most 4), for $2 \le i \le n-2$:

$$y_i' \approx d_i := \frac{1}{h}\left(\frac{1}{12}\,y_{i-2} - \frac{2}{3}\,y_{i-1} + \frac{2}{3}\,y_{i+1} - \frac{1}{12}\,y_{i+2}\right).$$

Using Taylor's expansions at x_i, we obtain a fourth order approximation of the first derivative:

$$d_i = y_i' - \frac{1}{30}h^4 y_i^{(5)} + O(h^5).$$

Near endpoints, there are two specific formulas:

$$y_0' \approx d_0 = \frac{1}{h}\left(-\frac{25}{12}\,y_0 + 4y_1 - 3y_2 + \frac{4}{3}y_3 - \frac{1}{4}y_4\right),$$

$$y_1' \approx d_1 = \frac{1}{h}\left(-\frac{1}{4}y_0 - \frac{5}{6}y_1 + \frac{3}{2}y_2 - \frac{1}{2}y_3 + \frac{1}{12}y_4\right).$$

Taylor's expansions at x_0 and x_1 give respectively

$$d_0 = y_0' - \frac{1}{5}h^4 y_0^{(5)} + 0(h^5), \qquad d_1 = y_1' + \frac{1}{20}h^4 y_1^{(5)} + 0(h^5).$$

In a similar way, we take

$$y_{m-1}' \approx d_{m-1} = \frac{1}{h}\left(\frac{1}{4}y_m + \frac{5}{6}y_{m-1} - \frac{3}{2}y_{m-2} + \frac{1}{2}y_{m-3} - \frac{1}{12}y_{m-4}\right).$$

$$y_m' \approx d_m = \frac{1}{h}\left(\frac{25}{12}\,y_m - 4y_{m-1} + 3y_{m-2} - \frac{4}{3}y_{m-3} + \frac{1}{4}y_{m-4}\right),$$

whose Taylor's expansions at x_{m-1} and x_m give respectively

$$d_{m-1} = y'_{m-1} + \frac{1}{20}h^4 y^{(5)}_{m-1} + 0(h^5), \qquad d_m = y'_m - \frac{1}{5}h^4 y^{(5)}_m + 0(h^5),$$

Remark. Recently, the author has proved that these quantities can be obtained as first derivatives of a specific cubic spline quasi-interpolant

$$Qf = \sum_{i=0}^{m+2} \lambda_i(f) B_i$$

where the B_i's are cubic B-splines with knots $\mathcal{X}_m$ and the λ_i's are discrete linear functionals defined by the following formulas

$$\lambda_i(f) := -\frac{1}{6}f_{i-2} + \frac{4}{3}f_{i-1} - \frac{1}{6}f_i, \quad 2 \le i \le m,$$

where $f_i = f(x_i)$, $0 \le i \le m$, with specific formulas at the endpoints

$$\lambda_0(f) := \frac{1}{36}(35f_0 + 4f_1 - 6f_2 + 4f_3 - f_4)$$

$$\lambda_1(f) := \frac{1}{18}(5f_0 + 26f_1 - 21f_2 + 10f_3 - 2f_4)$$

$$\lambda_{m+1}(f) := \frac{1}{18}(5f_m + 26f_{m-1} - 21f_{m-2} + 10f_{m-3} - 2f_{m-4})$$

$$\lambda_{m+2}(f) := \frac{1}{36}(35f_m + 4f_{m-1} - 6f_{m-2} + 4f_{m-3} - f_{m-4})$$

Moreover, the infinite norm of this QI satisfies $\|Q\|_\infty \le 32/9$ and the values of the QI on $\mathcal{X}_m$ satisfy $f(x_i) - Qf(x_i) = O(h^4)$ for all $0 \le i \le m$.

3 Univariate Cubic Spline Interpolation

Let $\mathcal{S} := \mathcal{S}_3^2(I, \mathcal{X}_m)$ be the space of C^2 cubic splines on the uniform partition of I in m subintervals $I_i := [x_{i-1}, x_i]$, $1 \le i \le m$. The restriction $p_i(t) \in \mathbb{P}_3$ of a spline $S \in \mathcal{S}$ to this subinterval can be written in the cubic Bernstein basis, via the change of variable $t := (x - x_{i-1})/h$, as follows:

$$p_i(t) = y_{i-1}b_0^{(3)}(t) + (y_{i-1} + \frac{h}{3}d_{i-1})b_1^{(3)}(t) + (y_i - \frac{h}{3}d_i)b_2^{(3)}(t) + y_i b_3^{(3)}(t),$$

where $d_i := S'(x_i)$, $0 \le i \le m$ and

$$b_0^{(3)}(t) := (1-t)^3, \quad b_1^{(3)}(t) := 3t(1-t)^2, \quad b_2^{(3)}(t) := 3t^2(1-t), \quad b_3^{(3)}(t) = t^3.$$

The unknown derivative values are solutions of the system of $m-1$ linear equations deduced from the C^2 continuity of S at points x_i :

$$D^2S(x_i^-) = D^2S(x_i^+), \quad \text{or} \quad D^2p_i(1) = D^2p_{i+1}(0), \quad 1 \le i \le m-1,$$

to which we must add two conditions at the endpoints, in order to get $m+1$ equations for the $m+1$ unknowns $\{d_i,\, 0 \le i \le m\}$. As the second derivative in x of p_i is equal to

$$\frac{1}{h^2}D^2 p_i(t) = \\ \frac{6}{h^2}\left((y_i - y_{i-1} - (d_i + 2d_{i-1}))(1-t) + (y_{i-1} - y_i + \frac{h}{3}(d_{i-1} + 2d_i))t\right)$$

the main equations are

$$\frac{3}{h}(y_{i-1} - y_i) + (d_{i-1} + 2d_i) = \frac{3}{h}(y_{i+1} - y_i) - (d_{i+1} + 2d_i)$$

and, setting $p_i := (y_i - y_{i-1})/h$, they can be written

$$d_{i-1} + 4d_i + d_{i+1} = 3(p_i + p_{i+1}), \quad 1 \le i \le m-1.$$

3.1 De Boor's Not a Knot Condition

Linear system. De Boor [3] suggest to take as additional conditions the C^3 continuity of S at knots $x = x_1$ and $x = x_{m-1}$, therefore the corresponding NAK cubic spline has no discontinuity at that point (whence the name "Not a Knot"). At the first point, this equation reads as follows:

$$y_1 - 3\left(y_1 - \frac{h}{3}d_1\right) + 3\left(y_0 + \frac{h}{3}d_0\right) - y_0 = y_2 - 3\left(y_2 - \frac{h}{3}d_2\right) + 3\left(y_1 + \frac{h}{3}d_1\right) - y_1,$$

or equivalently

$$d_0 - d_2 = (-2y_0 + 4y_1 - 2y_2)/h = 2(p_1 - p_2).$$

Subtracting this equation from the first equation $d_0 + 4d_1 + d_2 = 3(p_1 + p_2)$, we get

$$2d_1 + d_2 = (p_1 + 5p_2)/2.$$

In a similar way, at the point $x = x_{m-1}$, one has

$$d_{m-2} - d_m = (-2y_{m-2} + 4y_{m-1} - 2y_m)/h = 2(p_{m-1} - p_m).$$

Adding this equation to the last equation $d_{m-2} + 4d_{m-1} + d_m = 3(p_{m-1} + p_m)$, we get

$$d_{m-2} + 2d_{m-1} = (5p_{m-1} + p_m)/2.$$

Setting $d := [d_1, \ldots, d_{m-1}]^T \in \mathbb{R}^{m-1}$, $c := [(p_1 + 5p_2)/2, 3(p_2 + p_3) \ldots, 3(p_{m-2} + p_{m-1}), (5p_{m-1} + p_m)/2]^T \in \mathbb{R}^{m-1}$, and introducing the tridiagonal and positive definite matrix

$$K := \begin{bmatrix} 2 & 1 & 0 & & \dots & 0 & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 & 0 \\ \dots & 1 & 4 & 1 & & \dots & \\ \dots & & 1 & 4 & 1 & & \dots \\ \dots & & & \dots & \dots & \dots & \\ 0 & 0 & \dots & 0 & 1 & 4 & 1 \\ 0 & 0 & \dots & & 0 & 1 & 2 \end{bmatrix}$$

we have to solve the system

$$Kd = c.$$

Errors on derivatives. The errors on derivatives:

$$e'_i := d_i - y'_i, \quad 0 \le i \le m$$

satisfy the system of equations

$$\begin{aligned} 2e'_1 + e'_2 &= \qquad \varepsilon_1 := (p_1 + 5p_2)/2 - (2y'_1 + y'_2) \\ e'_{i-1} + 4e'_i + e'_{i+1} &= \quad \varepsilon_i := 3(p_i + p_{i+1}) - (y'_{i-1} + 4y'_i + y'_{i+1}), \\ e'_{m-2} + 2e'_{m-1} &= \varepsilon_{m-1} := (5p_{m-1} + p_m)/2 - (y'_{m-2} + 2y'_{m-1}) \end{aligned}$$

where $2 \le i \le m-2$. Using Taylor's expansions at x_i, we obtain

$$\varepsilon_i = -\frac{1}{30} h^4 y_i^{(5)} + O(h^5).$$

It remains to expand :

$$\varepsilon_1 = (p_1 + 5p_2)/2 - (2y'_1 + y'_2), \quad \varepsilon_{m-1} = (5p_{m-1} + p_m)/2 - (y'_{m-2} + 2y'_{m-1}).$$

Using Taylor's expansions at x_1, we first obtain

$$(p_1 + 5p_2)/2 = 3y'_1 + hy''_1 + \frac{1}{2}h^2 y_1^{(3)} + \frac{1}{12}h^3 y_1^{(4)} + \frac{1}{40}h^4 y_1^{(5)} + O(h^5)$$

$$y'_2 + 2y'_1 = 3y'_1 + hy''_1 + \frac{1}{2}h^2 y_1^{(3)} + \frac{1}{6}h^3 y_1^{(4)} + \frac{1}{24}h^4 y_1^{(5)} + O(h^5).$$

and finally

$$2e'_1 + e'_2 = \varepsilon_1 = -\frac{1}{12}h^3 y_1^{(4)} - \frac{1}{60}h^4 y_1^{(5)} + O(h^5).$$

In a similar way, using Taylor's expansions at x_{m-1}, we first obtain

$$(5p_{m-1} + p_m)/2 = 3y'_{m-1} - hy''_{m-1} + \frac{1}{2}h^2 y_{m-1}^{(3)} - \frac{1}{12}h^3 y_{m-1}^{(4)} + \frac{1}{40}h^4 y_{m-1}^{(5)} + O(h^5)$$

$$y'_{m-2} + 2y'_{m-1} = 3y'_{m-1} - hy''_{m-1} + \frac{1}{2}h^2 y_{m-1}^{(3)} - \frac{1}{6}h^3 y_{m-1}^{(4)} + \frac{1}{24}h^4 y_{m-1}^{(5)} + O(h^5).$$

and finally

$$e'_{m-2} + 2e'_{m-1} = \varepsilon_{m-1} = \frac{1}{12}h^3 y^{(4)}_{m-1} - \frac{1}{60}h^4 y^{(5)}_{m-1} + O(h^5).$$

Therefore, we have

$$e'_0 - e'_2 = \varepsilon_0 = 2(p_1 - p_2) - (y'_0 - y'_2)$$

$$e'_{m-2} - e'_m = \varepsilon_m = 2(p_{m-1} - p_m) - (y'_{m-2} - y'_m).$$

The corresponding expansions are respectively

$$\varepsilon_0 = \frac{1}{6}h^3 y^{(4)}_1 + O(h^5), \qquad \varepsilon_m = \frac{1}{6}h^3 y^{(4)}_{m-1} + O(h^5).$$

The errors satisfy the system of linear equations (with $2 \le i \le m-2$) :

$$\begin{aligned} e'_0 - e'_2 &= \tfrac{1}{6}h^3 y^{(4)}_1 + O(h^5) && (1)\\ 2e'_1 + e'_2 &= -\tfrac{1}{12}h^3 y^{(4)}_1 - \tfrac{1}{60}h^4 y^{(5)}_1 + O(h^5) && (2)\\ e'_{i-1} + 4e'_i + e'_{i+1} &= -\tfrac{1}{30}h^4 y^{(5)}_i + O(h^5) && (3)\\ e'_{m-2} + 2e'_{m-1} &= \tfrac{1}{12}h^3 y^{(4)}_{m-1} - \tfrac{1}{60}h^4 y^{(5)}_{m-1} + O(h^5) && (4)\\ e'_{m-2} - e'_m &= \tfrac{1}{6}h^3 y^{(4)}_{m-1} + O(h^5) && (5) \end{aligned}$$

Doing the linear combinations (1) + 2(2) and 2(4) − (5) give the new equations

$$e'_0 + 4e'_1 + e'_2 = -\frac{1}{30}h^4 y^{(5)}_1 + O(h^5), \quad e'_{m-2} + 4e'_{m-1} + e'_m = -\frac{1}{30}h^4 y^{(5)}_{m-1} + O(h^5).$$

In spite of the fact that all equations, except the first and the last, have a right hand side in $O(h^4)$, the overall error e'_i, for $0 \le i \le m$, is only $O(h^3)$ (see numerical examples in Section 5).

3.2 Improving De Boor's Not a Knot Condition : PGS Method

Linear system. We shall improve on de Boor's NAK condition of the previous section, in order to get errors on derivatives which are all in $O(h^4)$. We call this method PGS="Pretty Good Slopes". Instead of the first de Boor's equation

$$d_0 - d_2 = 2(p_1 - p_2),$$

we consider the new equation :

$$(E_0) \qquad a d_0 + b d_1 + c d_2 = \alpha p_1 + \beta p_2 + \gamma p_3,$$

where $a, b, c, \alpha, \beta, \gamma$ are free parameters. Of course, this equation must not be identical to equation

$$(E_1) \qquad d_0 + 4d_1 + d_2 = 3(p_1 + p_2)$$

So we get

$$ae'_0 + be'_1 + ce'_2 = \varepsilon_0 = \alpha p_1 + \beta p_2 + \gamma p_3 - (ay'_0 + by'_1 + cy'_2).$$

Using Taylor's expansions at x_1, swe obtain

$$\varepsilon_0 = (\alpha + \beta + \gamma - a - b - c)y'_1 + \frac{1}{2}hy''_1(-\alpha + \beta + 3\gamma + 2a - 2c)$$

$$+\frac{1}{6}h^2 y_1^{(3)}(\alpha + \beta + 7\gamma - 3a - 3c) + \frac{1}{24}h^3 y_1^{(4)}(-\alpha + \beta + 15\gamma + 4a - 4c) + O(h^4).$$

If we want ε_0 to be a $O(h^4)$, the 6 parameters have to satisfy the 4 following equations

$$\begin{aligned} \alpha + \beta + \gamma - a - b - c &= 0 \\ -\alpha + \beta + 3\gamma + 2a - 2c &= 0 \\ \alpha + \beta + 7\gamma - 3a - 3c &= 0 \\ -\alpha + \beta + 15\gamma + 4a - 4c &= 0 \end{aligned}$$

from which we deduce, taking a and c as parameters :

$$b = 3a + c, \quad \alpha = \frac{1}{6}(17a + c), \quad \beta = \frac{1}{3}(4a + 5c), \quad \gamma = \frac{1}{6}(-a + c).$$

Therefore, the new equation reads as follows:

$$(E_0) \qquad ad_0 + (3a + c)d_1 + cd_2 = \alpha p_1 + \beta p_2 + \gamma p_3,$$

The linear combination $(E_0) - a(E_1)$ of the two first equations gives

$$(c - a)(d_1 + d_2) = (\alpha - 3a)p_1 + (\beta - 3a)p_2 + \gamma p_3 = (c - a)\frac{1}{6}(p_1 + 10p_2 + p_3).$$

As we must have $c - a \neq 0$, we take for example $c - a = 1$ and we choose

$$a = 1, \quad c = 2, \quad b = 5, \quad \alpha = \frac{19}{6}, \quad \beta = \frac{14}{3}, \quad \gamma = \frac{1}{6},$$

whence the new equation (E_0) :

$$d_0 + 5d_1 + 2d_2 = \frac{1}{6}(19p_1 + 28p_2 + p_3).$$

Combining with equation (E_1)

$$d_0 + 4d_1 + d_2 = 3(p_1 + p_2)$$

we get the simpler and more symmetric equation

$$d_1 + d_2 = \frac{1}{6}(p_1 + 10p_2 + p_3) = c'_1,$$

and similarly

$$d_{m-2} + d_{m-1} = \frac{1}{6}(p_{m-2} + 10p_{m-1} + p_m) = c'_{m-1}.$$

The new system is $K'd = c'$ with the new matrix of size $m-1$:

$$K' := \begin{bmatrix} 1 & 1 & 0 & \dots & 0 & 0 \\ 1 & 4 & 1 & 0 \dots & 0 & 0 \\ \dots & 1 & 4 & 1 & \dots & \\ \dots & & 1 & 4 & 1 & \dots \\ \dots\dots & & & & & \\ 0 & 0 & \dots 0 & 1 & 4 & 1 \\ 0 & 0 & \dots & 0 & 1 & 1 \end{bmatrix}$$

and the new vector in the right-hand side

$$c' = [c'_1, 3(p_2 + p_3), \dots, 3(p_{m-2} + p_{m-1}), c'_{m-1}]^T.$$

Errors on derivatives. These errors $e'_i = d'_i - y'_i$ satisfy the equations

$$\begin{aligned} e'_1 + e'_2 &= \qquad \varepsilon'_1 := (p_1 + 10p_2 + p_3)/6 - (y'_1 + y'_2) \\ e'_{i-1} + 4e'_i + e'_{i+1} &= \qquad \varepsilon_i := 3(p_i + p_{i+1}) - (y'_{i-1} + 4y'_i + y'_{i+1}) \\ e'_{m-2} + e'_{m-1} &= \varepsilon'_{m-1} := (p_{m-2} + 10p_{m-1} + p_m)/6 - (y'_{m-2} + y'_{m-1}) \end{aligned}$$

where $2 \leq i \leq m-2$. From Taylor's expansions in Section 3.1, we deduce:

$$\frac{1}{6}(p_1 + 10p_2 + p_3) = 2y'_1 + hy''_1 + \frac{1}{2}h^2 y_1^{(3)} + \frac{1}{6}h^3 y_1^{(4)} + \frac{7}{120}h^4 y_1^{(5)} + O(h^5),$$

$$y'_1 + y'_2 = 2y'_1 + hy''_1 + \frac{1}{2}h^2 y_1^{(3)} + \frac{1}{6}h^3 y_1^{(4)} + \frac{1}{24}h^4 y_1^{(5)} + O(h^5),$$

whence, using the same technique for the last equation $(m-1)$:

$$\varepsilon'_1 = \frac{1}{60}h^4 y_1^{(5)} + O(h^5), \qquad \varepsilon'_{m-1} = \frac{1}{60}h^4 y_{m-1}^{(5)} + O(h^5),$$

and, for $2 \leq i \leq m-1$,

$$\varepsilon_i = -\frac{1}{30}h^4 y_i^{(5)} + O(h^5).$$

Therefore, we see that all right hand sides $\varepsilon_i, 0 \leq i \leq m$, are in $O(h^4)$, which implies that all errors are also in $O(h^4)$. This result is confirmed by numerical results (Section 5).

4 Solving Tridiagonal Systems for Cubic Splines

As both matrices are positive definite, their Choleski decompositions are

$$K = LL^T \quad \text{and} \quad K' = MM^T$$

with lower bidiagonal matrices L and M:

$$L := \begin{bmatrix} \lambda_1 & 0 & 0 & & \dots & 0 & 0 \\ \kappa_1 & \lambda_2 & 0 & 0 & \dots & 0 & 0 \\ 0 & \kappa_2 & \lambda_3 & 0 & & \dots & \\ \dots\dots & & & & & & \\ \dots & & \kappa_{i-1} & \lambda_i & 0 & & \dots \\ \dots\dots & & & & & & \\ 0 & 0 & \dots & 0 & \kappa_{m-3} & \lambda_{m-2} & 0 \\ 0 & 0 & \dots & & 0 & \kappa_{m-2} & \lambda_{m-1} \end{bmatrix}$$

$$M := \begin{bmatrix} \mu_1 & 0 & 0 & & \dots & 0 & 0 \\ \nu_1 & \mu_2 & 0 & 0 & \dots & 0 & 0 \\ 0 & \nu_2 & \mu_3 & 0 & & \dots & \\ \dots\dots & & & & & & \\ \dots & & \nu_{i-1} & \mu_i & 0 & & \dots \\ \dots\dots & & & & & & \\ 0 & 0 & \dots & 0 & \nu_{m-3} & \mu_{m-2} & 0 \\ 0 & 0 & \dots & & 0 & \nu_{m-2} & \mu_{m-1} \end{bmatrix}.$$

4.1 Computation of the Matrix *L* and Solution of the System

For the matrix L, we obtain successively

$$\lambda_1^2 = 2, \quad \text{hence} \quad \lambda_1 = \sqrt{2}, \quad \kappa_1 = 1/\lambda_1 = 1/\sqrt{2},$$

$$\lambda_i^2 + \kappa_{i-1}^2 = 4, \quad \text{hence} \quad \lambda_i = \sqrt{4 - \kappa_{i-1}^2}, \quad \kappa_i = 1/\lambda_i, \;\text{ for }\; 2 \le i \le m-2,$$

$$\lambda_{m-1} = \sqrt{2 - \kappa_{m-2}^2},$$

For solving $Kd = c$, one has to solve successively

$$L\delta = c \quad \text{and} \quad L^T d = \delta,$$

which leads to the simple algorithm:

$$\delta_1 = c_1/\lambda_1, \quad \delta_i = (c_i - \kappa_{i-1}\delta_{i-1})/\lambda_i, \quad i = 2 \dots m-1,$$

$$d_{m-1} = \delta_{m-1}/\lambda_{m-1} \quad d_i = (\delta_i - \kappa_i d_{i+1})/\lambda_i, \quad i = m-2 \dots 1.$$

The computational cost for (u_n) is (about) $3m$ flops, for L also $3m$ flops, and for the solution of each system $3m$ flops, thus the global cost is about $12m$ flops.

4.2 Computation of the Matrix M and Solution of the System

In the same way, for the matrix M, we obtain successively

$$\mu_1^2 = 1, \quad \text{hence} \quad \mu_1 = 1 \quad \text{and} \quad \nu_1 = 1/\mu_1 = 1.$$

$$\mu_i^2 + \nu_{i-1}^2 = 4, \quad \text{hence} \quad \mu_i = \sqrt{4 - \nu_{i-1}^2}, \quad \nu_i = 1/\mu_i, \ \text{for} \ 2 \le i \le m-2,$$

$$\mu_{m-1} = \sqrt{1 - \nu_{m-2}^2},$$

For solving $K'd' = c'$, one has to solve successively

$$M\delta' = c' \quad \text{and} \quad M^T d' = \delta',$$

which leads to the simple algorithm:

$$\delta_1' = c_1'/\mu_1, \quad \delta_i' = (c_i' - \nu_{i-1}\delta_{i-1}')/\mu_i, \quad i = 2 \dots m-1,$$

$$d_{m-1}' = \delta_{m-1}'/\mu_{m-1} \quad d_i' = (\delta_i' - \nu_i d_{i+1}')/\mu_i, \quad i = m-2 \dots 1.$$

The computational cost is about $12m$ flops, as for the preceding system.

5 Numerical Results

We compare the computations of derivatives using the three above methods (FD=finite differences, NAK=Not a Knot, PGS=Pretty Good Slopes) on various types of functions. Tables give the max of absolute values of errors at data points. The notation $4.88(-4)$ means 4.88×10^{-4}.

We easily verify the convergence orders $O(h^4)$ for FD and PGS and $O(h^3)$ for NAK. We also notice that PGS is in general better than FD: though one has to solve a linear system, the computational time is not much higher.

Examples 1 & 2 : $f(x) = \dfrac{1}{1+5x^2}$, $f(x) = \exp(-3x)\sin(3\pi x)$, $x \in [-1, 1]$.

m	FD	NAK	PGS
32	2.52(-3)	4.88(-4)	4.88(-4)
64	1.73(-4)	3.52(-5)	3.03(-5)
128	1.10(-5)	4.20(-6)	1.86(-6)
256	6.96(-7)	5.14(-7)	1.16(-7)
512	4.36(-8)	6.35(-8)	7.27(-9)
1024	2.72(-9)	7.89(-9)	4.54(-10)
2048	1.70(-10)	9.83(-10)	2.84(-11)

m	FD	NAK	PGS
32	0.08	0.75	0.08
64	1.95(-2)	7.56(-2)	1.60(-2)
128	1.72(-3)	7.92(-3)	1.31(-3)
256	1.21(-4)	8.84(-4)	9.07(-5)
512	8.00(-6)	1.03(-4)	5.92(-6)
1024	5.11(-7)	1.25(-5)	3.77(-7)
2048	3.23(-8)	1.53(-6)	2.38(-8)

Examples 3 & 4: $f(x) = 1/\cosh(x - 0.5)$, $x \in [-1, 1]$, $f(x) = \exp(-x^2)$, $x \in [-2, 2]$.

m	FD	NAK	PGS
32	1.47(-4)	3.52(-5)	1.06(-4)
64	8.23(-6)	1.15(-5)	5.94(-6)
128	4.75(-7)	1.85(-6)	3.46(-7)
256	2.84(-8)	2.56(-7)	2.08(-8)
512	1.73(-9)	3.35(-8)	1.27(-9)
1024	1.07(-10)	4.28(-9)	7.86(-11)
2048	6.64(-12)	5.40(-10)	4.88(-12)

m	FD	NAK	PGS
32	2.54(-4)	4.28(-4)	1.42(-4)
64	1.65(-5)	5.88(-5)	4.27(-6)
128	1.04(-6)	7.54(-6)	1.73(-7)
256	6.50(-8)	9.49(-7)	1.08(-8)
512	4.06(-9)	1.19(-7)	6.77(-10)
1024	2.54(-10)	1.49(-8)	4.23(-11)
2048	1.58(-11)	1.86(-9)	2.64(-12)

6 Estimation of Gradients

6.1 The Methods

We remind the reader of the notations of the introduction : $\Omega := [a,b] \times [c,d]$ is a rectangular domain endowed with the tensor-product partition $\mathcal{Z}_{m,n} := \mathcal{X}_m \otimes \mathcal{Y}_n$ where $\mathcal{X}_m := \{x_i = a + ih, 0 \le i \le m\}$ and $\mathcal{Y}_n := \{y_j = c + jk, 0 \le j \le n\}$ are uniform partitions with steplength h and k on $[a,b]$ and $[c,d]$ respectively.

Given a set of data values $z_{i,j} := f(M_{i,j})$ at gridpoints $M_{i,j} := (x_i, y_j)$, where f is a (known or unknown) function, we want to find good approximations of the gradients $g_{i,j} := (p_{i,j} = \partial_1 f(M_{i,j}), q_{i,j} = \partial_2 f(M_{i,j})$. The method consists of approximating derivatives in x (resp. in y) of restrictions of f to the $n+1$ horizontal lines (resp. to the $m+1$ vertical lines) for $\partial_1 f(M_{i,j})$ (resp. for $\partial_2 f(M_{i,j})$). Thus, for NAK and PGS methods, one has to solve $n+1$ tridiagonal systems in $m+1$ variables and $m+1$ tridiagonal systems in $n+1$ variables. The global computational cost is about $24mn$ flops, i.e. proportional to the number of data points.

6.2 Numerical Examples

We now compare the results obtained by using both methods with those obtained by using finite differences in x and y. We denote by R_F, R_K, R_S respectively the ratios of successive errors. They should be close to 8 (resp. 16) for a convergence order equal to 3 (resp. 4). The following tables give the maximum values of errors on all gridpoints for both partial derivatives. However, when f satisfies $f(x,y) = f(y,x)$ (examples 1 and 2), we only give one table.

Examples 1 & 2 : $f(x,y) = x^5y^5$, $\Omega = [0,1]^2$; $f(x,y) = \exp(-x^2 - y^2)$, $\Omega = [-1,1]^2$, $m = n$.

n	FD	R_F	NAK	R_K	PGS	R_S
8	5.8(-3)		3.7(-2)		4.3(-3)	
16	3.7(-4)	15.7	4.9(-3)	7.5	2.7(-4)	15.9
32	2.3(-5)	16.1	6.4(-4)	7.6	1.7(-5)	15.9
64	1.4(-6)	16.4	8.1(-5)	7.9	1.0(-6)	16.3
128	8.9(-8)	15.7	1.0(-5)	8.1	6.6(-8)	15.1

n	FD	R_F	NAK	R_K	PGS	R_S
8	2.0(-2)		1.6(-2)		1.4(-2)	
16	6.6(-4)	30.3	2.5(-3)	6.4	4.1(-4)	34
32	1.6(-5)	41.2	3.2(-4)	7.8	8.5(-6)	48
64	1.0(-6)	16	4.1(-5)	7.8	1.8(-7)	47
128	6.5(-8)	15.4	5.1(-6)	8	1.2(-8)	15

Example 3. $f(x, y) = \sin(\pi(x + 3y))$, $\Omega = [0, 1]^2$, $m = n$. The two tables give max errors on partial derivatives in x and y.

n	FD	R_F	NAK	R_K	PGS	R_S
8	1.4(-2)		3.4(-2)		1.1(-2)	
16	9.3(-4)	15.5	4.3(-3)		7.96.8(-4)	15.7
32	5.8(-5)	16	5.3(-4)	8;1	4.3(-5)	15.8
64	3.6(-6)	16.1	6.6(-5)	8.0	2.7(-6)	15.9
128	2.3(-7)	15.6	8.3(-6)	8.0	1.7(-7)	15.9
256	1.4(-8)	16.4	1.0(-6)	8.3	1.0(-8)	16.3

FD	R_F	NAK	R_K	PGS	R_S
2.9		2.6		2.4	
0.21	15.8	0.34	7.8	0.16	14.8
1.4(-2)	15	4.3(-2)	7.9	1.0(-2)	16
8.8(-4)	15.9	5.4(-3)	8.0	6.5(-4)	15.4
5.5(-5)	16	6.7(-4)	8.0	4.1(-5)	15.8
3.5(-6)	15.7	8.4(-5)	8.0	2.5(-6)	16.4

These examples confirm the convergence rates observed in the univariate case and the good behaviour of the PGS method.

The two following examples are more complex : the corresponding functions have huge variations. We still observe the good behaviour of FD and PGS methods, with a slight advantage to the latter. However, the computational cost of the PGS method, in the bivariate case, is between 4 and 6 times that of the FD method.

Example 4 : Franke's function [7][1]

$$\begin{aligned} f_2(x, y) = {} & .75 \exp(-\frac{1}{4}((9x-2)^2 + (9y-2)^2) \\ & + .75 \exp(-\frac{1}{49}(9x+1)^2 - \frac{1}{10}(9y+1)) \\ & + .5 \exp(-\frac{1}{4}((9x-7)^2 + (9y-3)^2) - .2 \exp(-(9x-4)^2 - (9y-7)^2), \end{aligned}$$

$$\Omega = [0, 1]^2, \quad m = n.$$

n	FD	R_F	NAK	R_K	PGS	R_S
16	0.12	7.25	6.6(-2)	9.1	7.9(-2)	13.3
32	1.1(-2)	10.9	1.1(-2)	6	2.2(-3)	35.9
64	7.4(-4)	14.9	1.5(-3)	7.3	1.3(-4)	16.9
128	4.8(-5)	15.6	1.8(-4)	8.3	8.0(-6)	16.2

FD	R_F	NAK	R_K	PGS	R_S
0.11	9.9	0.15	3.6	0.18	8.9
2.4(-2)	4.6	1.1(-2)	13.6	1.6(-2)	11.2
7.6(-4)	31.6	1.5(-3)	7.3	4.7(-4)	34
4.8(-5)	16	1.9(-4)	8.1	1.4(-5)	34

Note that the results for $\partial_2 f$ are rather unstable. We need finer meshes to get correct estimates of the order.

Example 5 : Nielson's function [9][1]

$$f(x, y) = 0.5\, y\, (\cos(4(x^2 + y - 1)))^4, \quad (x, y) \in [0, 1]^2$$

n	FD	R_F	NAK	R_K	PGS	R_S
16	2.36		2.25		2.27	
32	0.29	8.1	0.33	6.8	0.23	9.9
64	2.2(-2)	13.2	4.4(-2)	7.5	1.7(-2)	13.8
128	1.5(-3)	14.9	5.6(-3)	7.8	1.1(-3)	15.3

FD	R_F	NAK	R_K	PGS	R_S
0.18		0.21		0.15	
1.3(-2)	13.3	2.8(-2)	7.5	1.0(-2)	15
8.9(-4)	15.2	3.6(-3)	7.8	6.6(-4)	15.2
5.7(-5)	15.7	4.5(-4)	7.9	4.2(-5)	15.7

References

1. Arcangéli, R., López de Silanes, M.C., Torrens, J.J.: Multidimensional minimizing splines. Kluwer, Dordrecht (2004)
2. Barnhill, R.E., Nielson, G. (eds.): Surfaces. Special issue of Rocky Mountain J. Math. 14(1) (1984)
3. de Boor, C.: A practical guide to splines. Revised edition. Springer, Berlin (2000)
4. Foucher, F., Sablonnière, P.: Approximating partial derivatives of first and second order by quadratic spline quasi-interpolants on uniform meshes. Math. Comput. Simul. 77, 202–208 (2008)
5. Foucher, F., Sablonnière, P.: Superconvergence properties of some bivariate C^1 quadratic spline quasi-interpolants. In: Cohen, A., Merrien, J.-L., Schumaker, L.L. (eds.) Curves and Surfaces Fitting: Avignon 2006, pp. 160–169. Nashboro Press, Brentwood (2007)
6. Foucher, F., Sablonnière, P.: Quadratic spline quasi-interpolants and collocation methods. In: International Congress Mamern 2007, University of Granada (July 2007) (submitted)
7. Franke, R.: Scattered data interpolation: tests of some methods. Math.Comp. 38, 181–200 (1982)
8. Lai, M.J., Schumaker, L.L.: Splines on triangulations. Cambridge University Press, Cambridge (2007)
9. Nielson, G.: A first order blending method for triangles based upon cubic interpolation. Int. J. Numer. Meth. Eng. 15, 308–318 (1978)
10. Schumaker, L.L.: Spline functions: basic theory. John Wiley & Sons, New-York (1981); reprinted by Cambridge University Press (2008)
11. Stead, S.E.: Estimation of gradients from scattered data. In: [2], pp. 265–279

Metric Methods in Surface Triangulation

E. Saucan[1] and E. Appleboim[2]

[1] Mathematics Department, Technion, Haifa 32000, Israel
[2] Electrical Engineering Department, Technion, Haifa 32000
semil@tx.technion.ac.il, eliap@ee.technion.ac.il

Abstract. We consider the problem of better approximating surfaces by triangular meshes. The approximating triangulations are regarded as finite metric spaces and the approximated smooth surfaces are viewed as their Haussdorff-Gromov limit. This allows us to define in a more natural way the relevant elements, constants and invariants, such as principal directions and Gauss curvature, etc. By a "natural way" we mean intrinsic, discrete, metric definitions as opposed to approximating or paraphrasing the differentiable notions. Here we consider the problem of determining the Gauss curvature of a polyhedral surface, by using the *metric curvatures* in the sense of Wald, Menger and Haantjes. We present three modalities of employing these definitions for the computation of Gauss curvature.

1 Introduction

The paramount importance of triangulations of surfaces and their ubiquity in various implementations, such as numerous algorithms applied in robot and computer vision, computer graphics and geometric modeling, has hardly to be underlined here. Applications range from industrial ones, to biomedical engineering, cartography and astrography – to name just a few.

In consequence, determining the intrinsic properties of the surfaces under study, and especially computing their Gaussian curvature is essential. However Gauss curvature is a notion that is defined for smooth surfaces only, and usually attacked with differential tools. A common approach for dealing with non-smooth surfaces is to use discretizations of these differential tools such as numerical schemes for first and second order derivatives. Such an approach, though effective in various problems, can hardly represent good approximations for curvature for piecewise-flat (e.g. triangulated surface reconstructed from sample points), and even less so for grayscale images (see Figure 1 for a standard example), which are the actual objects under study in all real life aspects.

Moreover, since considering triangulations, one is faced with finite graphs, or, in many cases (when given just the vertices of the triangulation) only with finite – thus discrete – metric spaces. Therefore, the following natural questions arise:

Question 1. *Can one find discrete, metric equivalents of the differentiable notions, notions that are intrinsically more apt to describe the properties of the finite spaces under investigations?*

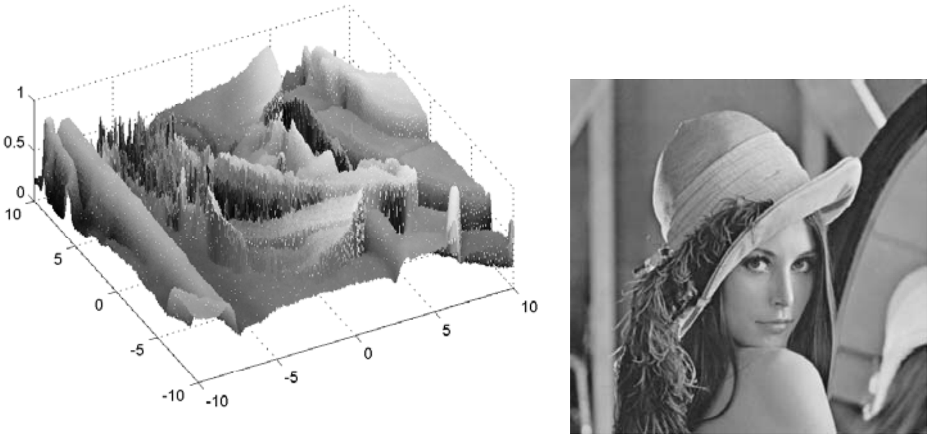


Fig. 1. A discrete grayscale "surface" reconstructed from sample points (left) of the ubiquitous "Lena" of image processing (right)

Question 2. *Is one fully justified in employing discrete metric spaces when evaluating numerical invariants of continuous surfaces?*

In this paper we review some fundamental studies that concerned these and other similar questions and show that the answers for them are affirmative. The outcome of this study is a fully rigorous mathematical theory of metric geometry which is, in a one sentence summary, the ability to apply differential geometry in metric spaces, the metric of whom may not be smooth. It is shown that their role is not restricted to that of being yet another discrete version of Gauss curvature, but permits one to attach a meaningful notion of curvature to points where the surface fails to be smooth, such as *cone points* and *critical lines.*

This metric method has already been successfully used in the such diverse fields as geometric group theory, geometric topology and hyperbolic manifolds, and geometric measure theory. Their relevance to computer graphics in particular and applied mathematics in general is made even more poignant by the study of *clouds of points* (see [13], [17]), in image processing (wavelets) [27] and also in applications in chemistry and biology (see [25]).

We propose to employ metric geometry to the study of triangulated surfaces, with a view to applications. Our method permits applying curvature reconstruction to cases where the classical notions do not apply, such as surfaces with "folds", "ridges" and "facets". The idea of using metric geometry in such "real life problems" is where the main novelty of this paper lies.

This is becoming more and more pertinent in recent years, because, starting with [12], isometric embeddings of manifolds have been considered, in various aspects and applications of and Image Processing and Computer Graphics. (We mention [5], (and its subsequent papers), [14], [26], but this represents a far from exhaustive list.)

The paper is organized as follows: in Section 2 we concentrate our efforts on the theoretical level and study the Lipschitz and Gromov-Hausdorff distances between metric spaces, and show that approximating smooth surfaces by nets and triangulations is not only permissible, but is, in a way, the natural thing to do. In particular we show that every compact surface is the Gromov-Hausdorff limit of a sequence of finite graphs. In Section 3 we introduce two metric analogues for the curvature of curves, namely the Menger, and Haantjes curvatures and study their mutual relationship. Furthermore, we show how to relate to these notions as metric analogues of sectional curvature and how to employ them in the evaluation of Gauss curvature of triangulated surfaces. Further in this section we present a metric version of curvature for, not necessarily smooth, surfaces. This is the embedding, or Wald curvature. We study its proprieties and investigate the relationship between Wald and Gauss curvatures, and show that for smooth surfaces they coincide. Hence, the Wald curvature represents a legitimate discrete candidate for approximating the Gaussian curvature for triangulated surfaces. Section 4 is dedicated to developing formulas that allow the computation of Wald curvature: first the precise ones, based upon the Cayley-Menger determinants, and then we develop (after Robinson) elementary formulas that approximate well the embedding curvature. In Section 5 in which we present some preliminary numerical results obtained by employing metric geometry tools for sampled surfaces and approximating Gaussian curvature by metric curvatures. Finally, in Section 6, we bring a few brief concluding remarks.

2 The Gromov-Haussdorff Limits

In this section we give a brief review of metric geometry, and mention those results and theorems that are the most relevant to the latter part of the paper. While most of the detailed and lengthy proofs are omitted, we try to give a flavor of the theory and its techniques. However, the main theorem will be stated along with its proof. For the full depth of the material presented in this section, we refer the reader to the beautiful and inspiring (but not always easy to follow) [11], and to [6], for an excellent, clear textbook. (A somewhat lengthier "digest" can be found in [23].)

We start by first introducing the classical *Hausdorff distance*:

Definition 1. *Let (X, d) be a metric space and let $A, B \subseteq X$. We define the* Hausdorff distance *between A and B as:*

$$d_H(A, B) = \inf\{r > 0 \,|\, A \subset U_r(B),\, B \subset U_r(A)\}\,,$$

where $U_r(A)$ is the r-neighborhood of A, $U_r(A) \overset{\triangle}{=} \bigcup_{a \in A} B_r(a)$.

Another equivalent way of defining the Hausdorff distance is as follows:

$$d_H(A, B) = \max\{\sup_{a \in A} d(a, B),\, \sup_{b \in B} d(b, A)\}\,.$$

Note that, while in general d_H is only a semi-metric, we have the following

Proposition 1. *Let (X,d) be a metric space. Then d_H is a metric on the set $\mathcal{M}(X)$ of closed subsets of X.*

Moreover, we have the following important results, (the second one being due to Blaschke), which we bring without their lengthy proofs (see [6], pp. 253-254):

Theorem 3. *(a) If X is complete, then $\mathcal{M}(X)$ complete.*
(b) If X is compact, then $\mathcal{M}(X)$ compact.

We wish to extend the Hausdorff metric to non-compact spaces. To do this, we proceed along the following basic guide-lines: we want to obtain the *maximum* distance d_{GH} that satisfies the two conditions below:

1. $d_{GH}(A,B) \leq d_H(A,B)$, for any $A, B \subset X$ (i.e. sets that are close as subsets of a given metric space X will still be close as abstract metric spaces);
2. X is isometric to Y iff $d_{GH}(X,Y) = 0$.

The definition we seek is the following one:

Definition 2. *Let X, Y be metric spaces. Then the* Gromov-Hausdorff distance *between X and Y is defined by:*

$$d_{GH}(X,Y) = \inf d_H^Z(f(X), g(Y))\,;$$

where the infimum is taken over all metric spaces Z in which both X and Y can be isometrically embedded and over all such isometric embeddings.

Remark 1. It is sufficient to consider embeddings f into the disjoint union of the spaces X and Y, $X \coprod Y$. $X \coprod Y$ is made into a metric space by defining

$$d(x,y) = \begin{cases} inf_{z \in X \cap Y}\left(d_X(x,z) + d_Y(z,y)\right), & (x \in X) \text{ and } (y \in Y)\,; \\ \infty\,, & X \cap Y = \emptyset\,. \end{cases}$$

The following notion is also important both in the theoretical setting and for our applicative purposes:

Definition 3. *Let (X,d) be a metric space, and let $A \subset X$. A is called an* ε-net *iff $d(x,A) \leq \varepsilon$, for all $x \in X$.*

The approximation of spaces by ε-nets is, as already hinted above, an important topic, both in the Differential Geometric context (see, e.g. [11]) and for its Vision and Graphics applications. In presenting some of its applications, we begin by citing the following basic result:

Theorem 4. *d_{GH} is a finite metric on the set of isometry classes of compact metric spaces.*

Moreover, the following important result holds:

Proposition 2. *Let $X, \{X_n\}_1^\infty$ compact metric spaces. Then: $X_n \underset{GH}{\longrightarrow} X$ iff for all $\varepsilon > 0$, there exist* finite *ε-nets $S \subset X$ and $S_n \subset X_n$, such that $S_n \underset{GH}{\longrightarrow} S$ and, moreover, $|S_n| = |S|$, for large enough n.*

The proposition above can be summarized as the convergence of geometric properties of S_n to those of S, as $X_n \underset{GH}{\longrightarrow} X$. A typical example is that of the *intrinsic metric* i.e. the metric induced by a length structure (i.e. path length) by a metric on a subset of a given metric space. (The classical example of surfaces in $\mathbb{R}^3$ being the motivating one.) It can be stated in an equivalent but less concise and elegant manner, that is, perhaps, more familiar to the Applied Mathematics community:

Proposition 3. *Let X, Y be compact metric spaces. Then:*
(a) If Y is a (ε, δ)-approximation of X, then $d_{GH}(X, Y) < 2\varepsilon + \delta$.
(b) If $d_{GH}(X, Y) < \varepsilon$, then Y is a 5ε-approximation of X.

Recall that *ε-δ-approximations* are defined as follow:

Definition 4. *Let X, Y be compact metric spaces, and let $\varepsilon, \delta > 0$. X, Y are called ε-δ-*approximations *(of each-other) iff: there exist sequences $\{x_i\}_{i=1}^N \subset X$ and $\{y_i\}_{i=1}^N \subset Y$ such that*
(a) $\{x_i\}_{i=1}^N$ is an ε-net in X and $\{y_i\}_{i=1}^N$ is an ε-net in Y;
(b) $|\, d_X(x_i, x_j) - d_(y_i, y_j)\, | < \delta$ for all $i, j \in \{1, ..., N\}$.
*An $(\varepsilon, \varepsilon)$-approximation is called, for short an ε-*approximation.

Amongst metric spaces, those whose metric d is intrinsic, are called *length spaces* and are of special interest in Geometry. The following theorem shows that length spaces are closed in the *GH-topology*:

Theorem 5. *Let $\{X_n\}$ be length spaces and let X be a complete metric space such that $X_n \underset{GH}{\longrightarrow} X$. Then X is a length space.*

The next theorem and its corollary are of paramount importance (and not only for our present study):

Theorem 6 (Gromov). *Any compact length space is the GH-limit of a sequence of finite graphs.*

Since the proof of the theorem above is constructive and thus also important in applications, we bring it below:

Proof. Let ε, δ ($\delta \ll \varepsilon$) small enough, and let S be a δ-net in X. Also, let $G = (V, E)$ be the graph with $V = S$ and $E = \{(x, y) \,|\, d(x, y) < \varepsilon\}$. We shall prove that G is an ε-approximation of X, for δ small enough (for $\delta < \frac{\varepsilon^2}{4}$\diam$(X)$, to be more precise).

But, since S is an ε-net both in X and in G, and since $d_G(x, y) \geq d_X(x, y)$, it is sufficient to prove that:

$$d_G(x, y) \leq d_X(x, y) + \varepsilon\,.$$

Let γ be the shortest path between x and y, and let $x_1, ..., x_n \in \gamma$, such that $n \leq \text{length}(\gamma)/\varepsilon$ (and $d_X(x_i, x_{i+1}) \leq \varepsilon/2$). Since for any x_i there exists $y_i \in S$, such that $d_X(x_i, y_i) \leq \delta$, it follows that $d_X(y_i, y_{i+1}) \leq d_X(x_i, x_{i+1}) + 2\delta < \varepsilon$.

Therefore, (for $\delta < \varepsilon/4$), there exists an edge $e \in G, e = y_i y_{i+1}$. From this we get the following upper bound for $d_G(x, y)$:

$$d_G(x, y) \leq \Sigma_0^n d_X(y_i, y_{i+1}) \leq \Sigma_0^n d_X(x_i, x_{i+1}) + 2\delta n$$

But $n < 2\,\text{length}(\gamma)/\varepsilon \leq 2\text{diam}(X)/\varepsilon$. Moreover: $\delta < \varepsilon^2/4\,\text{diam}(X)$. It follows that:

$$d_G(x, y) \leq d_X(x, y) + \delta \frac{4\text{diam}(X)}{\varepsilon} < d_X(x, y) + \varepsilon\,.$$

Thus, for any $\varepsilon > 0$, there exists an ε-approximation of X, $G = G_\varepsilon$. Hence $G_n \underset{\varepsilon}{\rightarrow} X$.

Corollary 7. *Let X be a compact length space. Then X is the Gromov-Hausdorff limit of a sequence $\{G_n\}_{n \geq 1}$ of finite graphs, isometrically embedded in X.*

Remark 2. (a) If $G_n \underset{\varepsilon}{\rightarrow} X$, $G_n = (V_n, E_n)$. If there exists $N_0 \in \mathbb{N}$ such that

$$(*) \quad |E_n| \leq N_0, \text{ for all } n \in \mathbb{N}\,,$$

then X is a finite graph.

(b) If condition $(*)$ is replaced by:

$$(**) \quad |V_n| \leq N_0, \text{ for all } n \in \mathbb{N}\,,$$

then X will still be always a graph, but not necessarily finite.

By Theorem 5, geometric properties of metric spaces are inherited by their Gromov-Hausdorff limits. Thus, we can use the Gromov-Hausdorff limit each and every time the geometric properties of X_n can be expressed in term of a finite number of points, and, by passing to the limit, automatically obtain proprieties of X. This is essentially the affirmative answer for Question 1. Moreover, this is the apparatus used in "the great scheme of things" of Differential Geometry, mainly for the *Cheeger-Gromov Precompactness Theorem* (see, [11]) and its subsequent developments in "classical" Riemannian Geometry (see, e.g. [19]), the theory of *Alexandrov spaces* (see, e.g. [21]), and, more recently, in the study of *metric measure spaces* ([11], [28], [30], [15], [20]).

In the next section we will consider Question 2 and discuss the ability of efficient approximation of some geometric properties of a smooth surface by those of a sequence of sampled surfaces, where the smooth surface is considered as the Gromov-Hausdorff limit of the sampled surfaces. The property we will focus on is curvature. Moreover, in view of the Theorem 6 and its Corollary, triangulations are considered as the finite graphs given by their 1-skeleta. We shall not, however, adopt the methods mentioned in the preceding paragraph, but rather a more direct (and more classical) approach, since we believe it is better fitted for applicative goals.

3 Metric Curvatures

In the light of the discussion at the end of the previous section, we focus in this section on a number of metric versions of curvature for rather general metric spaces. We begin by introducing metric analogues for the curvature of plane curves and in the sequel some metric definitions for Gauss curvature are considered.

We begin by introducing the most elementary of the metric curvatures, the *Menger* curvature: this is a metric expression for the circumradius of a triangle – thus giving in the limit a metric definition of the osculatory circle – and it is based upon some elementary high-school formulas:

Definition 5. *Let (M,d) be a metric space, and let $p,q,r \in M$ be three distinct points. Then:*

$$K_M(p,q,r) = \frac{\sqrt{(pq+qr+rp)(pq+qr-rp)(pq-qr+rp)(-pq+qr+rp)}}{pq \cdot qr \cdot rp} \, ;$$

is called the Menger Curvature *of the points p,q,r. (Here and throughout this section the distance between the points p, q is denoted, for brevity, by pq, etc.)*

We can now define the Menger curvature at a given point by passing to the limit:

Definition 6. *Let (M,d) be a metric space and let $p \in M$ be an accumulation point. We say that M has at p* Menger curvature $\kappa_M(p)$ *iff for any $\varepsilon > 0$, there exists $\delta > 0$, such that for any triple of points p_1, p_2, p_3, satisfying $d(p,p_i) < \delta$, $i = 1,2,3$; the following inequality holds: $|K_M(p_1,p_2,p_3) - \kappa_M(p)| < \varepsilon$.*

Applications of Menger curvature include, most notably, estimates (obtained via the Cauchy integral) for the regularity of fractals and the flatness of sets in the plane (see [18]). Also, it was employed, in a more practical implementation, for curve reconstruction (see [10]).

Since $\kappa_M(p)$ is modeled after a specific geometric property of the Euclidean plane, it conveys this Euclidean type of curvature upon the space it is defined on. Therefore it is unsuited for the geometrization of generic metric spaces. There exists, however, another notion of curvature that does not closely mimic $\mathbb{R}^2$, therefore is better fitted for generalizations (e.g. for the metrization of graphs – see [25]):

Definition 7. *Let (M,d) be a metric space and let $c : I = [0,1] \overset{\sim}{\to} M$ be a homeomorphism, and let $p,q,r \in c(I)$, $q,r \neq p$. Denote by $\widehat{qr}$ the arc of $c(I)$ between q and r, and by qr the line segment from q to r.*

Then c has Haantjes curvature $\kappa_H(p)$ *at the point p iff:*

$$\kappa_H^2(p) = 24 \lim_{q,r \to p} \frac{l(\widehat{qr}) - d(q,r)}{\left(l(\widehat{qr})\right)^3} \, ;$$

where "$l(\widehat{qr})$" denotes the length – in intrinsic metric induced by d – of $\widehat{qr}$.

Remark 3. It should be noted that κ_H exists only for rectifiable curves, but if κ_M exists at any point p of c, then c is rectifiable. Moreover, the following theorem (see [4], Theorem 10.2):

Theorem 8. *Let $c : I \to M$ be a rectifiable curve, and let $p \in M$. If $\kappa_M(p)$ exists (and is finite), then $\kappa_H(p)$ exists and $\kappa_H(p) = \kappa_M(p)$.*

It follows that Haantjes curvature represents, in fact, the more general notion (which should not be surprising in view of the fact, noted above, that the Menger curvature is inherently Euclidean).

Of course, both Menger and Haantjes curvatures can be employed, as approximations to sectional curvatures, of triangulated surfaces. In Section 5 we include some preliminary results in this direction.

3.1 Wald Curvature

A more powerful approach stems from Gauss' original method of comparing surface curvature to a standard, model surface (i.e. the unit sphere in $\mathbb{R}^3$). It was Wald's idea to use more types of gauge surfaces and to restrict oneself to the study of the minimal geometric figure that allows this comparison.

Definition 8. *Let (M,d) be a metric space, and let $Q = \{p_1, ..., p_4\} \subset M$, together with the mutual distances: $d_{ij} = d_{ji} = d(p_i, p_j)$; $1 \leq i,j \leq 4$. The set Q together with the set of distances $\{d_{ij}\}_{1 \leq i,j \leq 4}$ is called a* metric quadruple.

Remark 4. One can define metric quadruples in slightly more abstract manner, without the aid of the ambient space: a metric quadruple being a 4 point metric space, i.e. $Q = \big(\{p_1, ..., p_4\}, \{d_{ij}\}\big)$, where the distances d_{ij} verify the axioms for a metric.

Before passing to the next definition, let us introduce the following notation: S_κ denotes *the complete, simply connected surface of constant Gauss curvature* κ (or *space form*), i.e. $S_\kappa \equiv \mathbb{R}^2$, if $\kappa = 0$; $S_\kappa \equiv \mathbb{S}^2_{\sqrt{\kappa}}$, if $\kappa > 0$; and $S_\kappa \equiv \mathbb{H}^2_{\sqrt{-\kappa}}$, if $\kappa < 0$. Here $S_\kappa \equiv \mathbb{S}^2_{\sqrt{\kappa}}$ denotes the sphere of radius $R = 1/\sqrt{\kappa}$, and $S_\kappa \equiv \mathbb{H}^2_{\sqrt{-\kappa}}$ stands for the hyperbolic plane of curvature $\sqrt{-\kappa}$, as represented by the Poincaré model of the plane disk of radius $R = 1/\sqrt{-\kappa}$.

Definition 9. *The* embedding curvature $\kappa(Q)$ *of the metric quadruple Q is defined to be the curvature κ of the gauge surface S_κ into which Q can be isometrically embedded. (See Figure 3.1 for embeddings of a metric quadruple in $\mathbb{S}^2_{\sqrt{\kappa}}$ and $\mathbb{H}^2_{\sqrt{-\kappa}}$, respectively.)*

The embedding curvature at a point can now be defined in a natural way as a limit:

Definition 10. (M,d) *be a metric space, let $p \in M$ and let N be a neighbourhood of p. Then N is called* linear *iff N is contained in a geodesic curve.*

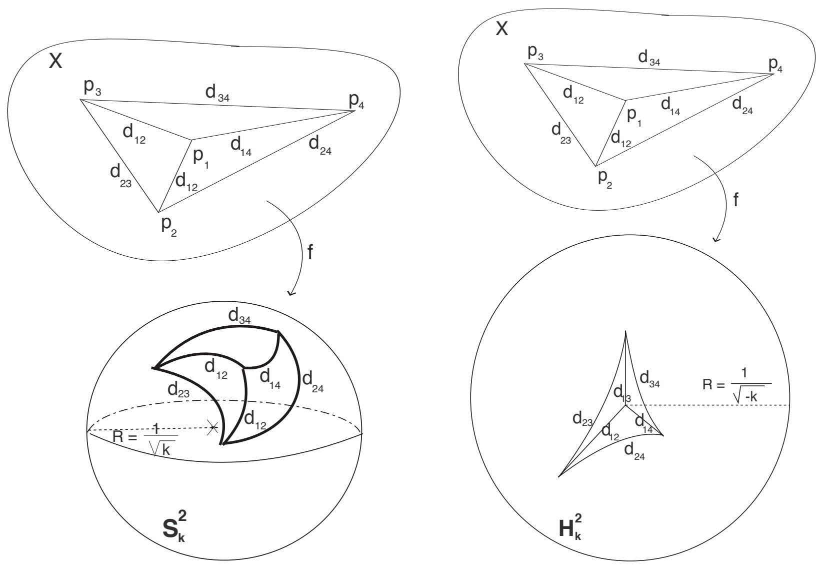


Fig. 2. Embedding of a metric quadruple in $\mathbb{S}^2_{\sqrt{\kappa}}$ (left) and $\mathbb{H}^2_{\sqrt{\kappa}}$ (right)

Definition 11. *Let (M, d) be a metric space, and let $p \in M$ be an accumulation point. Then M has (*embedding*)* Wald curvature *$\kappa_W(p)$ at the point p iff*

1. *Every neighbourhood of p is non-linear;*
2. *For any $\varepsilon > 0$, there exists $\delta > 0$ such that if $Q = \{p_1, ..., p_4\} \subset M$ and if $d(p, p_i) < \delta\,, i = 1, ..., 4$; then $|\kappa(Q) - \kappa_W(p)| < \varepsilon$.*

Note that if one uses the second (abstract) definition of the metric curvature of quadruples, then the very existence of $\kappa(Q)$ is not assured, as it is shown by the following counterexample (see [4]):

Counterexample 9. *The metric quadruple of lengths $d_{12} = d_{13} = d_{14} = 1$, $d_{23} = d_{24} = d_{34} = 2$ admits no embedding curvature.*

Moreover, even if a quadruple has an embedding curvature, it still may be not unique (even if Q is not linear), indeed, one can study the following examples:

Example 10. *(a) The quadruple Q of distances $d_{ij} = \pi/2, 1 \le i < j \le 4$ is isometrically embeddable both in $S_0 = \mathbb{R}^2$ and in $S_1 = \mathbb{S}^2$.*

(b) The quadruple Q of distances $d_{13} = d_{14} = d_{23} = d_{24} = \pi$, $d_{12} = d_{34} = 3\pi/2$ admits exactly two embedding curvatures: $\kappa_1 \in (1.5, 2)$ and $\kappa_2 = 3$.

However, for "good" metric spaces (i.e. spaces that are locally "plane like") the embedding curvature exists and it is unique. Moreover, this embedding curvature coincides with the classical Gaussian curvature. The proof of this result is rather involved, hence we shall not present here even a sketch of it. However, we note that the proof makes appeal to the notion of *semi-dependent quadruples*, which we describe below because it will prove useful in the following section.

Definition 12. *A metric quadruple $Q = Q(p_1, p_2, p_3, p_4)$, of distances $d_{ij} = dist(p_i, p_j)$, $i = 1, ..., 4$, is called* semi-dependent *(or a* sd-quad*, for brevity), iff 3 of its points are on a common geodesic, i.e. there exist 3 indices, e.g. 1,2,3, such that: $d_{12} + d_{23} = d_{13}$.*

Note that for sd-quads the uniqueness of the embedding curvature is assured:

Proposition 4. *An sd-quad admits at most one embedding curvature.*

In fact, a classification criterion for embedding curvature possibilities can be given (cf. Berestkovskii [1], see also [21], Theorem 18):

Theorem 11. *Let M, Q be as above. Then one and only one of the following assertion holds:*

1. *Q is linear.*
2. *Q has exactly one embedding curvature.*
3. *Q can be isometrically embedded in some $\mathcal{S}^m_\kappa$, $m \geq 2$; where $\kappa \in [\kappa_1, \kappa_2]$ or $(-\infty, \kappa_0]$, where $\mathcal{S}^m_\kappa \equiv \mathbb{R}^m$, if $\kappa = 0$; $\mathcal{S}^m_\kappa \equiv \mathbb{S}^m_{\sqrt{\kappa}}$, if $\kappa > 0$; and $\mathcal{S}^m_\kappa \equiv \mathbb{H}^m_{\sqrt{-\kappa}}$, if $\kappa < 0$. Moreover, $m = 2$ iff $\kappa \in \{\kappa_0, \kappa_1, \kappa_2\}$. (Here $\mathbb{S}^m_{\sqrt{\kappa}}$ denotes the m-dimensional sphere of radius $R = 1/\sqrt{\kappa}$, and $\mathbb{H}^m_{\sqrt{-\kappa}}$ stands for the m-dimensional hyperbolic space of curvature $\sqrt{-\kappa}$, as represented by the Poincaré model of the ball of radius $R = 1/\sqrt{-\kappa}$).*
4. *There exist no m and k such that Q can be isometrically embedded in $\mathcal{S}^m_\kappa$.*

3.2 Wald and Gauss Curvatures Comparison

The discussion above would have little relevance in Differential Geometry in general and for the problem of approximating curvatures of triangulated surfaces, in particular, were it not for the fact that the metric (Wald) and the classical (Gauss) curvatures coincide whenever the second notion makes sense, that is for smooth (i.e. of class $\geq \mathcal{C}^2$) surfaces in $\mathbb{R}^3$. More precisely the following theorem holds:

Theorem 12 (Wald [31]). *Let $S \subset \mathbb{R}^3$, $S \in \mathcal{C}^m$, $m \geq 2$ be a smooth surface. Then, given $p \in S$, $\kappa_W(p)$ exists and $\kappa_W(p) = \kappa_G(p)$.*

Moreover, Wald also proved the following partial reciprocal theorem:

Theorem 13. Let M be a compact and convex metric space. If $\kappa_W(p)$ exists, for all $p \in M$, then M is a smooth surface and $\kappa_W(p) = \kappa_G(p)$, for all $p \in M$.

Note that if one tries to restrict oneself, in the building of Definition 11 only to sd-quads, then Theorem 13 holds only if the following presumption is added:

Condition 14. *M is locally homeomorphic to $\mathbb{R}^2$.*

Unfortunately, the proof of these facts is laborious and, as such, beyond the scope of this paper. Therefore we shall restrict ourselves to note that the main idea is to show that if a metric M space admits a Wald curvature at any point, than M is locally homeomorphic to $\mathbb{R}^2$, thus any metric proprieties of $\mathbb{R}^2$ can be translated to M, in particular the first fundamental form. (For the detailed proofs of the results above, see [3], [4] and, for a a succinct description of the principal steps of the proofs, [24].)

4 Computing Wald Curvature

In this section we bring formulas for the computation and approximation of embedding curvature of quadruples. In the beginning we follow the classical approach of Wald-Blumenthal (see, e.g., [3], [4]) that employs the so-called *Cayley-Menger determinants* (see below). Unfortunately, the formulas obtained, albeit precise, are transcendental, and as such difficult to employ in practical implementations. Therefore we next present the approximate formulas developed by C.V. Robinson [22].

The Cayley-Menger Determinant. Given a general metric quadruple $Q = Q(p_1, p_2, p_3, p_4)$, of distances $d_{ij} = dist(p_i, p_j),\ i = 1, ..., 4$; denote by $D(Q) = D(p_1, p_2, p_3, p_4)$ the following determinant:

$$D(p_1, p_2, p_3, p_4) = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12}^2 & d_{13}^2 & d_{14}^2 \\ 1 & d_{12}^2 & 0 & d_{23}^2 & d_{24}^2 \\ 1 & d_{13}^2 & d_{23}^2 & 0 & d_{34}^2 \\ 1 & d_{14}^2 & d_{24}^2 & d_{34}^2 & 0 \end{vmatrix} \tag{4.1}$$

The determinant $D(Q) = D(p_1, p_2, p_3, p_4)$ is called the *Cayley-Menger determinant* (of the points $p_1, ...p_4$). This definition readily generalizes to any dimension, as do the results below. To get some geometric intuition regarding formula (4.1) we first examine the Euclidean case (see [3], [2] for details).

We start with the following proposition:

Proposition 5. The points $p_1, ..., p_4$ are the vertices of a non-degenerate simplex in $\mathbb{R}^3$ iff $D(p_1, p_2, p_3, p_4) \neq 0$.

In fact, a much stronger result can be proven:

Theorem 15. *Let $d_{ij} > 0, 1 \leq 4, i \neq j$. Then there exists a simplex $T = T(p_1, ..., p_4) \subseteq \mathbb{R}^3$ such that $dist(x_i, x_j) = d_{ij}, i \neq j$; iff $D(p_i, p_j) < 0$, for any $\{i, j\} \subset \{1, ..., 4\}$ and $D(p_i, p_j, p_k) > 0$, for any $\{i, j, k\} \subset \{1, ..., 4\}$; where, for instance,*

$$D(p_1,p_2) = \begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & d_{12}^2 \\ 1 & d_{12}^2 & 0 \end{vmatrix}$$

and

$$D(p_1,p_2,p_3) = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & d_{12}^2 & d_{13}^2 \\ 1 & d_{12}^2 & 0 & d_{23}^2 \\ 1 & d_{13}^2 & d_{23}^2 & 0 \end{vmatrix} \; ;$$

etc...

The development of the expressions of volumes as Cayley-Menger determinants, in the spherical and hyperbolical cases, are far too technical for this limited exposition; suffice therefore to add that they essentially reproduce the proof given in the Euclidean case, taking into account the fact that, when performing computations in the spherical (resp. hyperbolic) metric, one has to replace the distances d_{ij} by $\cos d_{ij}$ (resp. $\cosh d_{ij}$) – see [3] for the full details. Now the following formula for the *embedding curvature* $\kappa(Q)$ of Q (and its dependence upon the curvature's sign of the embedding space) is natural:

$$\kappa(Q) = \begin{cases} \quad 0 \quad \text{if } D(Q) = 0\,; \\ \kappa,\ \kappa < 0 \text{ if } det(\cosh\sqrt{-\kappa}\cdot d_{ij}) = 0\,; \\ \kappa,\ \kappa > 0 \text{ if } det(\cos\sqrt{\kappa}\cdot d_{ij}) \text{ and } \sqrt{\kappa}\cdot d_{ij} \leq \pi \\ \qquad\qquad \text{and all the principal minors of order 3 are} \geq 0. \end{cases} \tag{4.2}$$

Approximate Formulas. We have noted in the preceding section that the formulas (4.2) we just developed are not only transcendental, but also the computed curvature may fail to be unique. However, uniqueness is guaranteed for sd-quads. Moreover, the relatively simple geometric setting of sd-quads facilitates the development of simple (i.e. rational) formulas for the approximation of the embedding curvature.

Theorem 16 ([22]). Given the metric semi-dependent quadruple $Q = Q(p_1,p_2,p_3,p_4)$, of distances $d_{ij} = dist(p_i,p_j),\ i,j = 1,...,4$; the embedding curvature $\kappa(Q)$ admits a rational approximation given by:

$$K(Q) = \frac{6(\cos\measuredangle_0 2 + \cos\measuredangle_0 2')}{d_{24}\big(d_{12}\sin^2(\measuredangle_0 2) + d_{23}\sin^2(\measuredangle_0 2')\big)} \tag{4.3}$$

where: $\measuredangle_0 2 = \measuredangle(p_1p_2p_4)$, $\measuredangle_0 2' = \measuredangle(p_3p_2p_4)$ represent the angles of the Euclidian triangles of sides d_{12}, d_{14}, d_{24} and d_{23}, d_{24}, d_{34} , respectively.

Moreover, the *absolute error* R satisfies the following inequality:

$$|R| = |R(Q)| = |\kappa(Q) - K(Q)| < 4\kappa^2(Q)diam^2(Q)/\lambda(Q)\,, \tag{4.4}$$

where $\lambda(Q) = d_{24}(d_{12}\sin\measuredangle_0 2 + d_{23}\sin\measuredangle_0 2')/S^2$, and where $S = Max\{p,p'\}$; $2p = d_{12} + d_{14} + d_{24}$, $2p' = d_{32} + d_{34} + d_{24}$.

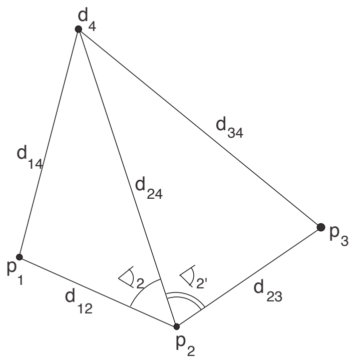


Fig. 3. An sd-quad

Since Robinson's result is, unfortunately, rather forgotten, and since the proof is a good illustration of the classical methods employed in the study of embedding curvature, we include it below.

Proof. The basic idea of the proof is to mimic, in a general metric setting, the Gauss map (see, e.g. [9]) – in this case one measures the curvature by the amount of "bending" one has to apply to a general planar quadruple so that it can be isometrically embedded as a *sd-quad*) in S_κ, for some κ.

Consider two planar (i.e. embedded in $R^2 \equiv \mathcal{S}_0$) triangles $\triangle p_1p_2p_4$ and $\triangle p_2p_3p_4$, and denote by $\triangle p_1^\kappa p_2^\kappa p_4^\kappa$ and $\triangle p_2^\kappa p_3^\kappa p_4^\kappa$ their respective isometric embeddings into $\mathcal{S}_\kappa$. Then $p_{i,\kappa}p_{j,\kappa}$ will denote the geodesic (of $\mathcal{S}_\kappa$) through $p_{i,\kappa}$ and $p_{j,\kappa}$. Also, let $\measuredangle_\kappa 2$ and $\measuredangle_\kappa 2'$ denote, respectively, the following angles of $\triangle p_{1,\kappa}p_{2,\kappa}p_{4,\kappa}$ and $\triangle p_{2,\kappa}p_{3,\kappa}p_{4,\kappa}$: $\measuredangle_\kappa 2 = \measuredangle p_{1,\kappa}p_{2,\kappa}p_{4,\kappa}$ and $\measuredangle_\kappa 2' = \measuredangle p_{2,\kappa}p_{3,\kappa}p_{4,\kappa}$ (see Figure 4).

But $\measuredangle_\kappa 2$ and $\measuredangle_\kappa 2'$ are strictly increasing as functions of κ. Therefore the equation

$$\measuredangle_\kappa 2 + \measuredangle_\kappa 2' = \pi \tag{4.5}$$

has at most one solution κ^*, i.e. κ^* represents the unique value for which the points p_1, p_2, p_3 are on a geodesic in $\mathcal{S}_\kappa$ (for instance on p_1p_3), therefore κ^* is precisely the embedding curvature, i.e. $\kappa^* = \kappa(Q)$, where $Q = Q(p_1, p_2, p_3, p_4)$.

Equation (4.5) is equivalent to

$$\cos^2 \frac{\measuredangle_{\kappa^*} 2}{2} + \cos^2 \frac{\measuredangle_{\kappa^*} 2'}{2} = 1$$

The basic idea being the comparison between metric triangles with equal sides, embedded in $\mathcal{S}_0$ and $\mathcal{S}_\kappa$, respectively, it is natural to consider instead of the previous equation, the following equality:

$$\theta(\kappa, 2) \cdot \cos^2 \frac{\measuredangle_0 2}{2} + \theta(\kappa, 2') \cdot \cos^2 \frac{\measuredangle_0 2'}{2} = 1\,, \tag{4.6}$$

where we denote:

$$\theta(\kappa,2)=\frac{\cos^2\frac{\measuredangle_{\kappa^*}2}{2}}{\cos^2\frac{\measuredangle_0 2}{2}}\,;\;\;\theta(\kappa,2')=\frac{\cos^2\frac{\measuredangle_{\kappa^*}2'}{2}}{\cos^2\frac{\measuredangle_0 2}{2}}\,.$$

Since we want to approximate $\kappa(Q)$ by $K(Q)$ we shall resort – naturally – to expansion into MacLaurin series. We are able to do this because of the existence of the following classical formulas:

$$\cos^2\frac{\measuredangle_\kappa 2}{2}=\frac{\sin(p\sqrt{\kappa})\cdot\sin(d\sqrt{\kappa})}{\sin(d_{12}\sqrt{\kappa})\cdot\sin(d_{24}\sqrt{\kappa})}\,;\;\kappa>0\,;$$

$$\cos^2\frac{\measuredangle_\kappa 2}{2}=\frac{\sinh(p\sqrt{\kappa})\cdot\sinh(d\sqrt{\kappa})}{\sinh(d_{12}\sqrt{\kappa})\cdot\sinh(d_{24}\sqrt{\kappa})}\,;\;\kappa<0\,;$$

and, of course

$$\cos^2\frac{\measuredangle_0 2}{2}=\frac{pd}{d_{12}d_{24}}\,;$$

where $d=p-d_{14}=(d_{12}+d_{24}-d_{14})/2$ (and the analogous formulas for $\cos^2\frac{\measuredangle_{\kappa'}2}{2}$).

By using the development into series of $f_1(x)=\frac{\sin\sqrt{x}}{\sqrt{x}}$ and $f_2(x)=\frac{\sinh\sqrt{x}}{\sqrt{x}}$; one (easily) gets the desired expansion for $\theta(\kappa,2)$:

$$\theta(\kappa,2)=1+\frac{1}{6}\kappa d_{12}d_{24}\big(\cos(\measuredangle_0 2)-1\big)+r\,; \tag{4.7}$$

where: $|r|<\frac{3}{8}\kappa^2p^4$, for $|\kappa p^2|<1/16$.

From (4.7) and (4.6), we obtain:

$$\Big[1+\frac{1}{6}\kappa^* d_{12}d_{24}\big(\cos(\measuredangle_0 2)-1\big)+r\Big]\cos^2\frac{\measuredangle_0 2}{2}+ \tag{4.8}$$
$$\Big[1+\frac{1}{6}\kappa^* d_{23}d_{24}\big(\cos(\measuredangle_0 2')-1\big)+r'\Big]\cos^2\frac{\measuredangle_0 2'}{2}=1\,;$$

for: $|r|+|r'|<\frac{3}{4}(\kappa^*)^2(Max\{p,p'\})^4=\frac{3}{4}(\kappa^*)^2S^4$.

By solving the linear equation (in variable κ^*) (4.8) and using some elementary trigonometric transformation one has:

$$\kappa^*=\frac{6(\cos\measuredangle_0 2+\cos\measuredangle_0 2')}{d_{24}\big(d_{12}\sin^2(\measuredangle_0 2)+d_{23}\sin^2(\measuredangle_0 2')\big)}+R\,;$$

where:

$$|R|<\frac{12(|r|+|r'|)}{d_{24}\big(d_{12}\sin^2(\measuredangle_0 2)+d_{23}\sin^2(\measuredangle_0 2')\big)}$$
$$<\frac{9(\kappa^*)^2\max\{p,p'\}}{d_{24}\big(d_{12}\sin^2(\measuredangle_0 2)+d_{23}\sin^2(\measuredangle_0 2')\big)}\,.$$

But $\kappa^*\equiv\kappa(Q)$, so we get the desired formula (4.3) .

To prove the correctness of the bound (4.4) one has only to observe that:

$$S = Max\{p, p'\} < 2\text{diam}(Q),\ \big(\text{diam}(Q) = \max_{1\leq i<j\leq 4}\{d_{ij}\}\big),$$

and perform the necessary arithmetic manipulations.

Example 17 ([22]). *Let Q_0 be the quadruple of distances $d_{12} = d_{23} = d_{24} = 0.15, d_{14} = d_{34}$ and of embedding curvature $\kappa = \kappa(Q_0) = 1$. Then $\kappa S^2 < 1/16$ and $K(Q_0) \approx 1.0030280$, which shows that the actual computed error can be far less then the one given by formula (4.4), which, in this case gives $|R| < 0.45$.*

Remark 5. (a) The function $\lambda = \lambda(Q)$ is continuous and 0-homogenous as a function of the d_{ij}-s. Moreover: $\lambda(Q) \geq 0$ and $\lambda(Q) = 0 \Leftrightarrow \sin\measuredangle_0 2 = \sin\measuredangle_0 2' = 0$, i.e. iff Q is linear. (Therefore for sd-quads $\lambda(Q) > 0$. Moreover, when $\lambda(Q)$ tends to 0, Q approaches linearity.)

(b) Since $\lambda(Q) \neq 0$ it follows that: $K(Q) \in \mathbb{R}$ for any quadrangle Q. Moreover: $\text{sign}(\kappa(Q)) = \text{sign}(K(Q))$.

(c) If Q is any sd-quad, then $\kappa^2(Q)\text{diam}^2(Q)/\lambda(Q) < \infty$. Moreover, $|R|$ is small if Q is not close to linearity. In this case $|R(Q)| \sim \text{diam}^2(Q)$ (for any given Q).

Since the Gaussian curvature $K_G(p)$ at a point p is given by:

$$K_G(p) = \lim_{n\to 0}\kappa(Q_n)\,;$$

where $Q_n \to Q = \Box p_1pp_3p_4\,;\ \text{diam}(Q_n) \to 0$, from Remark 5(c) we immediately infer that the following holds:

Theorem 18. *Let S be a differentiable surface. Then, for any point $p \in S$:*

$$K_G(p) = \lim_{n\to 0} K(Q_n)\,;$$

for any sequence $\{Q_n\}$ of sd-quads that satisfy the following condition:

$$Q_n \to Q = \Box p_1pp_3p_4\,;\ \text{diam}(Q_n) \to 0\,.$$

Remark 6. In the following special cases even "nicer" formulas are obtained:

1. If $d_{12} = d_{32}$, then

$$K(Q) = \frac{12}{d_{13}\cdot d_{24}}\cdot\frac{\cos\measuredangle_0 2 + \cos\measuredangle_0 2'}{\sin^2\measuredangle_0 2 + \sin^2\measuredangle_0 2'}\,; \tag{4.9}$$

   (here we have of course: $d_{13} = 2d_{12} = 2d_{32}$); or, expressed as a function of distances alone:

$$K(Q) = 12\frac{2d_{12}^2 + 2d_{24}^2 - d_{14}^2 - d_{13}^2}{8d_{12}^2d_{24}^2 - (d_{12}^2 + d_{24}^2 - d_{14}^2)^2 - (d_{12}^2 + d_{24}^2 - d_{34}^2)^2} \tag{4.10}$$

2. If $d_{12} = d_{32} = d_{24}$ and if the following condition also holds:
3. $\measuredangle_0 2' = \pi/2$; i.e. if $d_{34}^2 = d_{12}^2 + d_{24}^2$ or, considering 2., also: $d_{34}^2 = 2d_{12}^2$, then

$$K(Q) = \frac{6\cos\measuredangle_0 2}{d_{12}(1+\sin^2\measuredangle_0 2)} = \frac{2d_{12}^2 - d_{14}^2}{4d_{12}^4 + 4d_{14}^2d_{12}^2 - d_{14}^4}\,. \tag{4.11}$$

5 Experimental Results

In this section we view some preliminary numerical results of approximating Gauss curvature of the torus by the metric curvature computed on a sequence of sampled tori with increasing resolutions. The precise parametrization of the torus is known, therefore computational error can be precisely assessed, in particular if one uses (as it is commonly done) the standard square grid in the parametric plane. In addition to that the torus was chosen since it has both positive, zero and negative Gauss curvature. Computations were done using various definitions of metric curvatures. In the graphs below (Figures 4 and 5) approximation results are given. In both examples Haantjes and Robinson approximations are done while computing sectional curvatures along the mesh edges. Therefore in order to approximate Gauss curvature more accurately one needs to adjust the triangulation so that its edges will best coincide with geodesic lines of the surface. This is done by adding all the diagonals, in one direction, in the square grid mentioned above. The second graph shows improvement of the results after such an adjustment was done.

We also analyzed the relative performances of the considered algorithms as displayed in the regions of various sign of Gauss curvature, both for the original numerical schemes and for the improved ones. Notice that both Robinson and Wald based algorithms display a "jump" at the boundary between the elliptic and hyperbolic regions – see Figure 6. This step-function behaviour is due to the

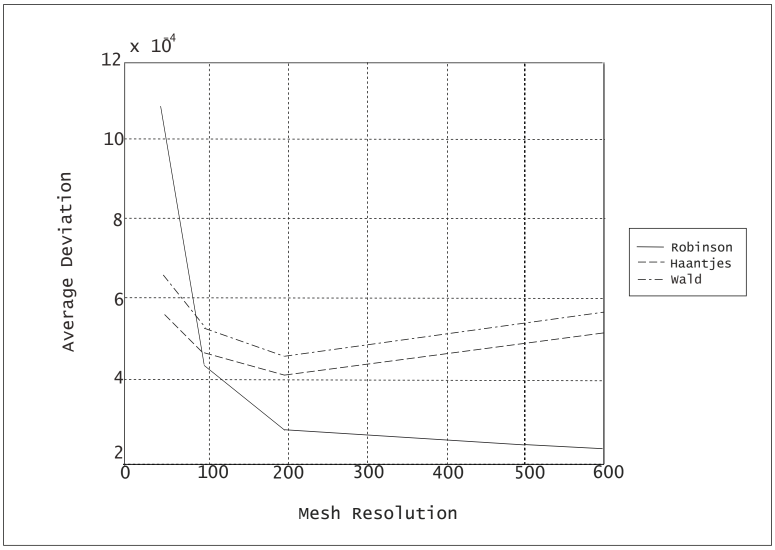


Fig. 4. Approximation errors with respect to mesh resolution (as number of triangles)

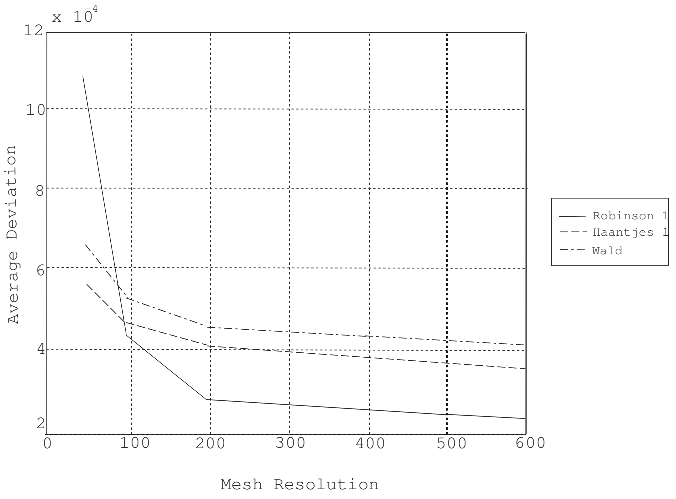


Fig. 5. Improved approximation errors with respect to mesh resolution (as number of triangles)

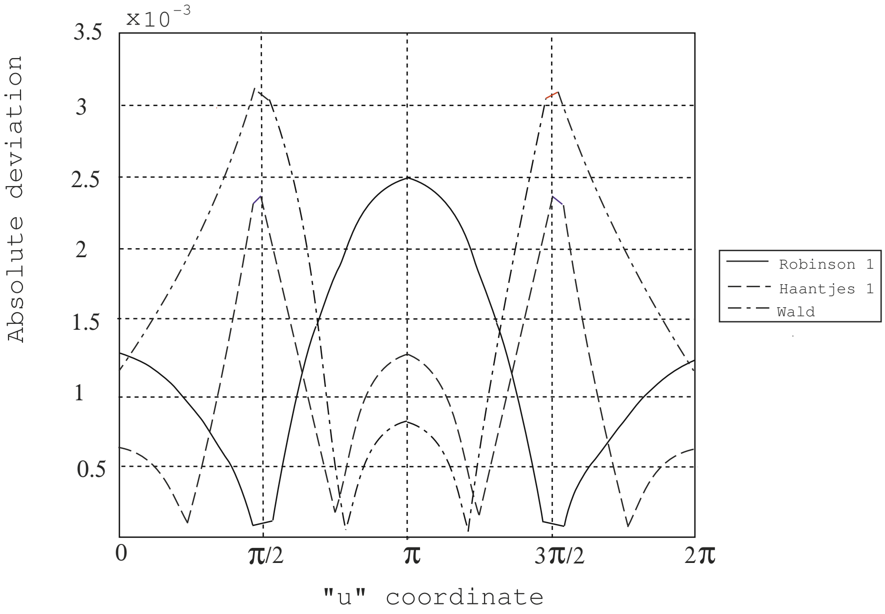


Fig. 6. Average deviation as a function of the "meridian" coordinate "u", for the torus of revolution: $\mathbf{T}^2 = \mathbf{T}^2(u, v) = \big((R + r\cos u)\cos v, (R + r\cos u)\sin v, r\sin u\big)$

trichotomy intrinsic to both methods, trichotomy that is induced by the sign of the curvature.

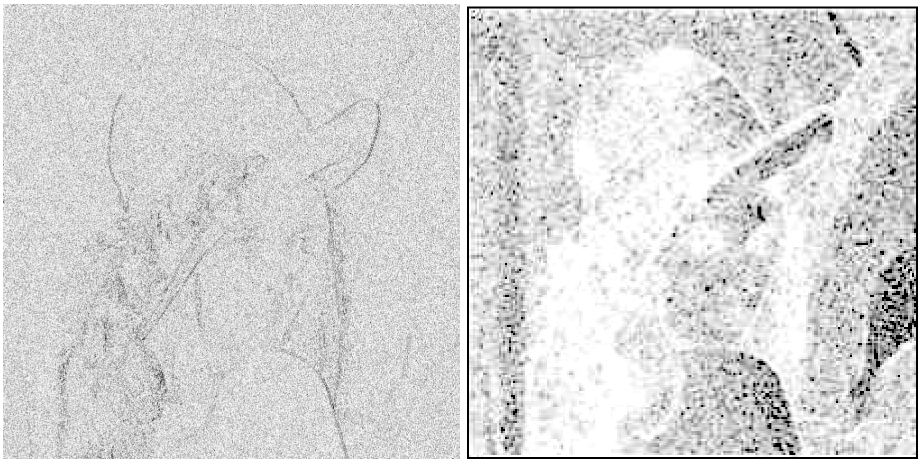


Fig. 7. Mean curvature, computed using the Haantjes curvature (left) and Wald curvature (right) of the grayscale "Lena"

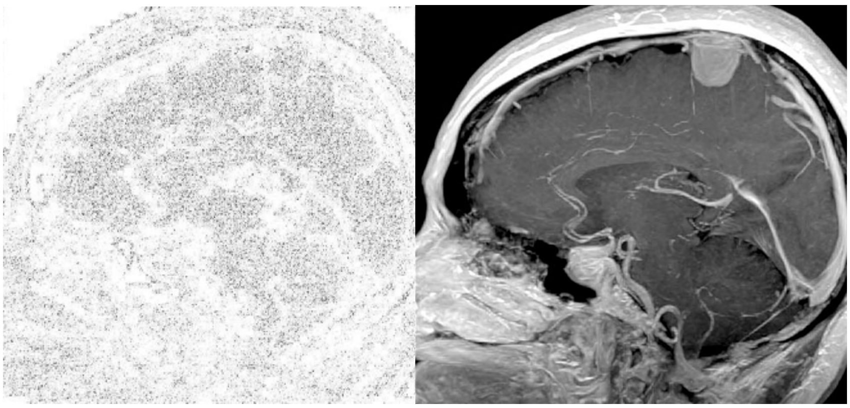


Fig. 8. Wald curvature (right) of a cerebral radiographic image (left)

Of course, for a better evaluation of the capabilities and limits of the metric methods, further experiments on more diverse surfaces and with finer meshes are to be undertaken. In addition, some experiments were also performed on real data, both on standard test images (see Figure 7) and on medical images (see Figure 8). We shall briefly discuss the performance of our methods, as shown by these examples, in the concluding section below.

6 Final Remarks

In the preceding sections we brought positive answers to both of the questions posed in the introduction. More precisely, we have shown that using the

Gromov-Hausdorf, one can assure the convergence of geometric properties of approximating meshes, or just of point samples, to those of the targeted smooth surface. Moreover, for curvature, there are a number of excellent options for the purely metric quantities representing discretizations of the classical notions.

However, as can be seen from the results above, the convergence rates of these metric methods is rather slow, and, at least for smooth surfaces, they are outperformed by more classical methods. Even when good convergence is theoretically assured, as in the case of Robinson's formula, numerical instability is high. One is therefore entitled to ask oneself whether there are means to improve their performance and, moreover, if there is a place for such methods, beyond the theoretical setting mentioned above.

To the first of these questions, one can answer in various ways, trying to illuminate the various facets of this problem. Results regarding the convergence rate, in the Gromov-Hausdorf, of the considered approximations, can be found in [8]. The influence of the shape of simplices of the triangulation on the performance of approximation algorithms, as far as curvature is concerned, has also been extensively studied. Due to the lack of space, we include here only two references: [7] for a far reaching theoretical study, and [26] for some applicative aspects. For the Wald curvature, better ways or "sampling" the directions on the surfaces can, and should, be devised. But beyond these potential improvements, more-or-less technical in nature, the importance of these metric curvatures lies in the applications were a semi-discrete (or semi-continuous) notion is required (such as those introduced in [25] and [27]). In addition, they offer natural quantization of the error in "guessing" a smooth surface, whenever a learning approach is adopted (see [26]).

Acknowledgements

The first author would like to thank Prof. Gershon Elber of The Computer Science Department, The Technion, Haifa, who motivated and sustained this project in its incipient phase and Prof. Shahar Mendelson – his warm support is gratefully acknowledged.

Both authors would like to thank their students who helped produce some of the images herein.

References

1. Berestowskii, V.N.: Spaces with bounded curvature and distance geometry. Siberian Math. J. 16, 8–19 (1986)
2. Berger, M.: Geometry I, II, Universitext. Springer, Berlin (1987)
3. Blumenthal, L.M.: Distance Geometry – Theory and Applications. Claredon Press, Oxford (1953)
4. Blumenthal, L.M., Menger, K.: Studies in Geometry. Freeman and Co., San Francisco (1970)
5. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Three-dimensional face recognition. International Journal of Computer Vision 64(1), 5–30 (2005)

6. Burago, D., Burago, Y., Ivanov, S.: Course in Metric Geometry. GSM 33. AMS, Providence (2000)
7. Cheeger, J., Müller, W., Schrader, R.: On the Curvature of Piecewise Flat Spaces. Comm. Math. Phys. 92, 405–454 (1984)
8. Dai, J., Luo, W., Jin, M., Zeng, W., He, Y., Yau, S.-T., Gu, X.: Geometric accuracy analysis for discrete surface approximation. Computer Aided Geometric Design 24, 323–338 (2007)
9. do Carmo, M.P.: Differential Geometry of Curves and Surfaces. Prentice-Hall, Englewood Cliffs (1976)
10. Giesen, J.: Curve Reconstruction, the Traveling Salesman Problem and Menger's Theorem on Length. In: Proceedings of the 15th ACM Symposium on Computational Geometry (SoCG), pp. 207–216 (1999)
11. Gromov, M.: Metric structures for Riemannian and non-Riemannian spaces. Progress in Mathematics, vol. 152. Birkhauser, Boston (1999)
12. Hallinan, P.: A low-dimensional representation of human faces for arbitrary lighting conditions. In: Proc. CVPR, pp. 995–999 (1994)
13. Liu, G.H., Wong, Y.S., Zhang, Y.F., Loh, H.T.: Adaptive fairing of digitized point data with discrete curvature. Computer Aided Design 34(4), 309–320 (2002)
14. Kimmel, R., Malladi, R., Sochen, N.: Images as Embedded Maps and Minimal Surfaces: Movies, Color, Texture, and Volumetric Medical Images. International Journal of Computer Vision 39(2), 111–129 (2000)
15. Lott, J.: Optimal transport and Ricci curvature for metric-measure spaces. In: Cheeger, J., Grove, K. (eds.) Surveys in Differential Geometry XI, Metric and Comparison Geometry, pp. 229–257. International Press, Somerville (2007)
16. Machigashira, Y.: The Gaussian curvature of Alexandrov surfaces. J. Math. Soc. Japan 50(4), 859–878 (1998)
17. Maltret, J.-L., Daniel, M.: Discrete curvatures and applications: a survey (preprint) (2003)
18. Pajot, H.: Analytic Capacity, Rectificabilility, Menger Curvature and the Cauchy Integral. LNM, vol. 1799. Springer, Berlin (2002)
19. Peterson, P.: Convergence Theorems in Riemannian Geometry. In: Grove, K., Petersen, P. (eds.) Comparison Geometry. MSRI Publications, vol. 30, pp. 167–202. Cambridge University Press, Cambridge (1997)
20. Ollivier, Y.: Ricci curvature of Markov chains on metric spaces. J. Funct. Anal. (to appear)
21. Plaut, C.: Metric Spaces of Curvature $\geq k$. In: Daverman, R.J., Sher, R.B. (eds.) Handbook of Geometric Topology, pp. 819–898 (2002)
22. Robinson, C.V.: A Simple Way of Computing the Gauss Curvature of a Surface. Reports of a Mathematical Colloquium, Second Series 5-6, 16–24 (1944)
23. Saucan, E.: Surface triangulation – the metric approach, arxiv:cs.GR/0401023 (preprint) (2004)
24. Saucan, E.: Curvature – Smooth, Piecewise-Linear and Metric. In: Sica, G. (ed.) What is Geometry?, pp. 237–268. Polimetrica, Milano (2006)
25. Saucan, E., Appleboim, E.: Curvature based clustering for DNA microarray data analysis. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. LNCS, vol. 3523, pp. 405–412. Springer, Heidelberg (2005)
26. Saucan, E., Appleboim, E., Zeevi, Y.Y.: Sampling and Reconstruction of Surfaces and Higher Dimensional Manifolds. Journal of Mathematical Imaging and Vision 30(1), 105–123 (2008)
27. Saucan, E., Sagiv, C., Appleboim, E.: Geometric Wavelets for Image Processing: Metric Curvature of Wavelets (preprint) (2009)

28. Sturm, K.-T.: Generalized Ricci curvature bounds and convergence of metric measure spaces. C. R. Acad. Sci. Paris, Ser. I 340, 235–238 (2005)
29. Troyanov, M.: Tangent Spaces to Metric Spaces: Overview and Motivations. In: Séminaire Borel 2003 (IIIème Cycle Romand de Mathématiques), Berne (2003)
30. Villani, C.: Optimal Transport, Old and New. Grundlehren der mathematischen Wissenschaften, vol. 338. Springer, Heidelberg (2009)
31. Wald, A.: Begründung einer koordinatenlosen Differentialgeometrie der Flächen. Ergebnisse e. Mathem. Kolloquiums, First Series 7, 24–46 (1935)

Setback Vertex Blends in Digital Shape Reconstruction

Z. Terék[1] and T. Várady[2]

[1] Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Hungary
[2] Geomagic, Inc., Research Triangle Park, NC, USA

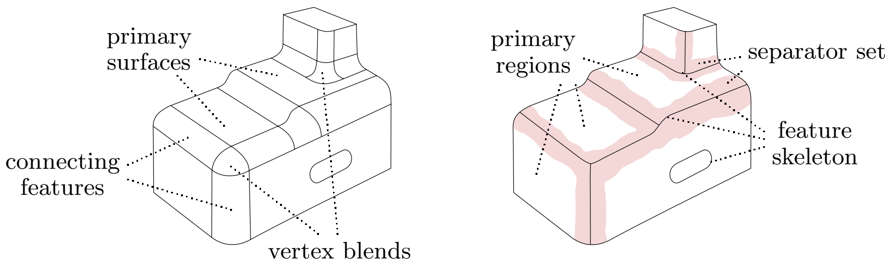
Abstract. Digital shape reconstruction (DSR) deals with creating CAD models of physical objects using 3D scanned data. Our primary interest is to reconstruct mechanical engineering objects that are usually composed of a hierarchy of surfaces – primary surfaces, connecting features (fillets) and vertex blends – and are structured by well-defined topological rules. After an overview of segmenting large polygonal meshes by the functional decomposition paradigm, we focus on the reconstruction of vertex blends using setbacks. This topic was thoroughly studied more than a decade ago in the context of constructive CAD; now the concept is revisited for DSR. A new method is presented to locate the optimal cross-sectional termination of fillets and construct the boundary curves of vertex blends on the mesh. These will correspond to the vertex blend boundaries of the final CAD model, as well. Finally, we discuss special cases of self-intersecting segmenting curve networks, and show how these problems can be resolved by setback vertex blends.

Keywords: digital shape reconstruction, functional decomposition, segmentation, vertex blends, setbacks.

1 Introduction

The goal of Digital Shape Reconstruction (formerly known as reverse engineering) is to create digital models from physical objects. DSR is a category within a broader discipline called Digital Shape Sampling and Processing (DSSP), which includes all point cloud related computations and techniques used in a wide variety of fields [1]. In some applications, e.g. medical areas, it is often sufficient to convert the measured point cloud into a polygonal mesh; lots of well-established *triangulation* based techniques are available to process them [2].

For CAD/CAM/CAE applications, usually further steps are required to create high quality surface models. In the functional decomposition approach first a likely topological structure of the unknown object is extracted automatically; then utilising this structure high-quality, constrained surfaces are computed with none or minimal user assistance [3]. The key issue is to properly *segment* the polygonal mesh into regions that correspond to the "not yet known" surface entities. Once such segmentation is available, good approximating surfaces are



(a) Typical surface layout (b) Simplified topological structure

Fig. 1. An example of a mechanical engineering object

fitted to the regions automatically, which are eventually *trimmed* and *stitched* together to obtain a boundary representation CAD model.

The majority of CAD objects in mechanical engineering are constructed by certain simple rules. As a result, the bounding surfaces can be arranged into a well-defined hierarchy, see Fig. 1a:

1. *Primary surfaces* are to satisfy various functional or aesthetic requirements; these are relatively large surfaces, which can be classified as simple, analytic types (such as planes, cylinders), swept surfaces (such as extrusions, surfaces of revolution) or free-form surfaces.
2. *Connecting features* lie between the primary surfaces; these are relatively narrow, highly-curved surface elements that join the primaries smoothly. The majority of connecting features represent fillets: however, other types, such as step surfaces, also frequently occur. In this paper, the algorithms will be described for fillets, though they directly generalise for more complex connecting feature types, as well.
3. *Vertex blends* smoothly connect fillets (features) and primaries, being represented by small, typically doubly-curved n-sided patches. (Alternatively, the term *corner patch* is used.)

1.1 Motivation

Segmentation is the most crucial part of shape reconstruction. The quality of the final surfaces depends on the correct segmentation: misclassified points often drastically worsen the quality of primary surfaces and fillets, or even may prohibit the creation of complete and consistent CAD models. Segmentation is a very difficult task since 'a priori' neither the surfaces, nor their topological structure is known [3].

Our goal is to define an optimal segmenting curve network on the mesh, which will one-to-one correspond to the edge-loop-face structure of the final CAD model. The segmenting curve network partitions the mesh, and defines the

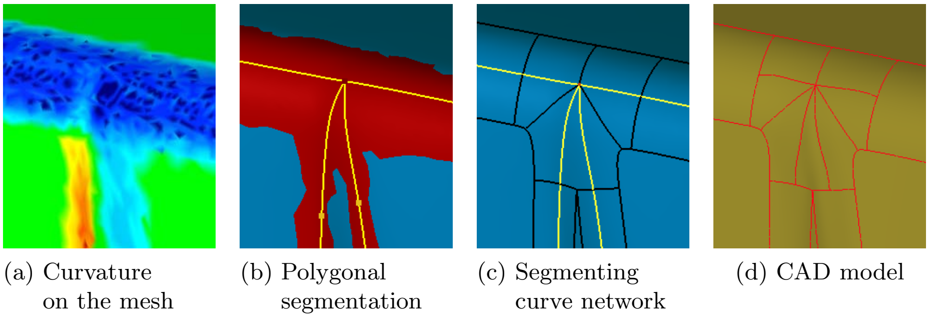

(a) Curvature on the mesh (b) Polygonal segmentation (c) Segmenting curve network (d) CAD model

Fig. 2. Sequence of vertex blend creation

point sets to be individually approximated. This defines (i) the mathematical surfaces to be fitted and (ii) the corresponding network of edges that will trim these surfaces. Of course, the final vertex blend surfaces can only be fitted once the adjacent primaries and fillets have been computed, but without creating the vertex blend structure on the mesh, we cannot create the correct point regions to be approximated by the primaries and the fillets.

In constructive CAD, designers do not create vertex blends explicitly, as CAD systems generate them as a 'by product' of the filleting operations. At the same time, creating smooth and natural vertex blends used to be a very difficult problem in geometric modelling and a huge amount of effort was invested to work out robust solutions. At first sight the problem may seem to be simple, but difficulties arise when convex and concave fillets with different radii need to be connected, that span different angles, including tangential and cuspate cases, etc. The analysis of these problems led to the introduction of the so-called *setback type vertex blends* that we are also going to apply; see details in [4,5].

To sum it up: our main interest is to compute a topologically consistent and geometrically pleasing structure of vertex blends on the mesh, which is an important step to complete DSR by functional decomposition. This facilitates the completion of the segmenting curve network, that integrates the individually traced longitudinal boundary curves of the "to be reconstructed" fillets. It also facilitates the actual creation of surfaces, when the final CAD model is computed. The process is illustrated in Fig 2, showing the individual steps from numerical curvature estimation through creating the segmenting curve network on the mesh to the final surface representation.

1.2 Previous Works

Digital shape reconstruction from 3D measured point clouds is an emerging high technology that has been intensely studied since the mid-nineties. Data acquisition and surface reconstruction techniques have been rapidly evolved, and by now several commercial systems can serve various engineering and medical applications. The earliest attempts of producing consistent and complete CAD models came from the computer vision community [6]. A broader overview on the most difficult problems from the geometric modelling point of view was first published in [3].

The conventional techniques of digital shape reconstruction aimed at creating only individual surfaces using manual segmentation methods or region growing techniques, such as [7,8]. The former approach was quite tedious and error prone for complex parts, while the second often lacked global consistency between the adjacent surface elements. Detecting features is still an open research area; in a recent publication a feature sensitive metric was proposed to support segmentation [9].

There have been alternative approaches that focused on objects bounded by simple analytic surfaces only; these limited the scope of object reconstruction, but developed useful techniques to apply various constrained fitting methods amongst the surface entities [10,11,12,13].

Another trend of digital shape reconstruction aimed at building complex surface models in a highly *automatic* manner, thus minimising user assistance [14,15]. A global structure of quadrilateral tiles was built, which assured watertight connections. The method worked in a computationally highly efficient manner, but it could not satisfy the CAD/CAM user community fully, since the structure lacked a proper alignment with the geometric features of the object and the most frequent engineering surface elements could not be reproduced in exact, trimmed form.

Currently there are two leading approaches to build full CAD models, that are capable to support downstream CAD/CAM applications, see [16]. The first approach – "redesign" – mimics the sequence of operations of constructive CAD, but at each operation it is possible to extract and utilise corresponding measured data. In this way, a parametric history file can be created, however, the operations of constructive CAD need to be performed step by step [20]. The alternative approach – "functional decomposition" – places the emphasis on the global automation of the reconstruction process. The user can extract automatically or with minimal assistance the full structure of the object, which makes it possible to generate a complete CAD model that consists of primary surfaces and connecting features stitched together [21]. This paper investigates one particular aspect of the latter approach of creating vertex blends to join various fillets.

2 Technical Background

Functional decomposition is an automatic procedure (i) to create a topological structure on the polygonal mesh, and (ii) generate a surface model according to this structure with its implied geometric properties and dependencies. For further details the reader is referred to [17]. In this section we discuss this framework focusing on the segmentation process, but keeping in mind the successive steps, as well. This will be followed by an overview on setback vertex blends.

2.1 Segmentation

The reconstruction of vertex blends is part of segmentation. A schematic object illustrates this process, see Figs. 1a and 1b. We assume that the input is a good quality triangulated mesh (noise reduced, holes filled, properly decimated, etc.). Segmentation consists of the following five phases, each operation builds on top of the previous one.

1. *Hierarchical Morse segmentation.* First we locally compute curvature-like measures at the vertices of the mesh and apply a segmentation technique based on discrete Morse-theory [18,19]. This partitions the mesh into a set of topologically meaningful components that correspond to the primary regions. The boundaries of the components are zig-zagged polylines.
2. *Primary regions and separator set.* In order to obtain a more suitable representation we compute the likely polygonal extent of the fillets and the vertex blends by thickening the polyline boundaries over the mesh. This yields the so-called *separator set*, see Fig. 1b, which separates the resulting, somewhat shrunken primary regions.
3. *Extract a feature skeleton.* The feature skeleton is the medial axis of the separator set. Its edges and nodes will correspond to the fillets and vertex blends, resp. The feature skeleton replaces the zig-zagged polylines by a set of smoothed contour lines, running in the middle of the separator set.
4. *Extract longitudinal boundaries.* The next step is to trace a pair of boundary curves for each individual fillet. This is an automatic process: boundary generation is driven by the contour lines of the feature skeleton and the width of the underlying separator set.
5. *Create vertex blends and complete segmenting curve network.* The individual boundary curves of the fillets need to be integrated into a consistent *segmenting curve network.* This involves the computation of vertex blends, which simultaneously determines the cross-sectional terminations of the incoming fillets and supplements small corner arcs for the loops of primary regions, as will be discussed in details in Sec. 3.

The above segmentation steps result in a geometrically well-aligned structure that is needed for creating high-quality surface models. Until now we have performed mesh operations only, the forthcoming ones are to define surface geometries step by step. Based on the primary regions an *automatic classification* is performed which will set the most likely surface type for each region (analytic, swept, free-form, etc.). The primary surfaces will approximate the primary region data by the detected surface type. The fillets are *dependent* entities, which not only approximate the underlying data, but are also constrained to smoothly join the adjacent – already existing – primaries. For rolling ball blends tangential (G^1) continuity is assured, for free-form objects generally curvature continuous fillets are needed. Vertex blends are also dependent entities that need to connect the already existing fillets and primaries. The final CAD model is created by *trimming and stitching* the individual surface entities together.

2.2 Vertex Blends

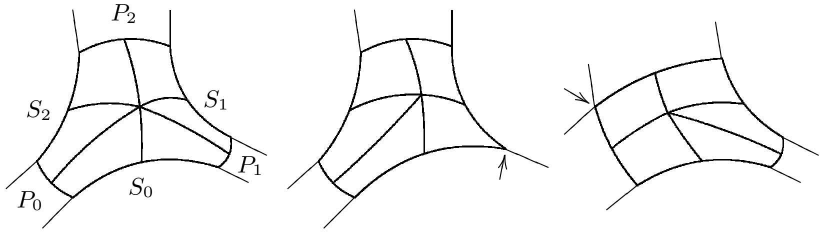
Vertex blends are relatively small surface elements that smoothly connect incoming fillets at their junction. The simplest form of a vertex blend is the well-known “suitcase-corner,” which is a *three-sided* patch (an octant of a sphere) that connects *three fillets* of the same radii. The corners of the vertex blend are obtained by intersecting the adjacent fillet boundaries. Unfortunately, it is hardly possible to generalise this blending technique to a wide variety of complex vertex blend configurations due to geometrical and topological problems.

To create general vertex blends that connect an arbitrary number of fillets and permit arbitrary combination of convex and concave edges with different radii, the concept of *setback type vertex blends* was introduced [4,5]. The basic idea is to extend the vertex blends and shrink the incoming fillets by *setback distances.* The scheme produces natural transitions for many difficult vertex blend configurations.

The terminology along with a few special cases is explained in Fig. 3. In the most general case n fillets are blended using a $2n$-sided vertex blend. The boundary edges of the vertex blend, called *profile* and *spring curves,* alternate. The profile curve is a shared boundary with a fillet (P_i in Fig. 3a); the spring curve is shared boundary with a primary surface, denoted by S_i. A profile curve always connects two adjacent primary surfaces, the spring curve always connects the boundaries of two adjacent fillets.

Fig. 3b shows a special case where one of the profile curves degenerates; this corresponds to a vertex blend with a sharp edge. Fig. 3c illustrates the case of a missing spring curve, where two adjacent profile curves come together at the intersection point of two longitudinal fillet boundaries. In principle, any number of sides between n and $2n$ can represent a valid vertex blend configuration.

As discussed earlier, vertex blends are always dependent on their neighbouring fillets and primaries, and must be fitted using constrained algorithms. There are different methods to approximate the vertex blend regions by smooth surfaces. In this paper we do not deal with the geometric aspects of fitting, just mention two basic approaches. Using the *trimmed* approach, a single quadrilateral surface is fitted to the data points and that is trimmed by the projected



(a) Complete (b) Missing profile curve (c) Missing spring curve

Fig. 3. Configurations of setback vertex blend with central split

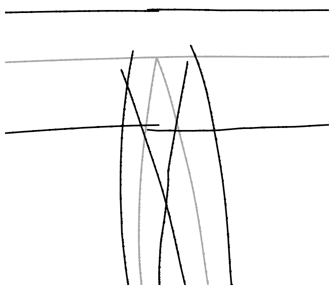
boundaries of the region in order to make it stitchable to the adjacent surfaces. An alternative way is the *central split* approach, where the vertex blend region is subdivided into n quadrilateral patches around a centre point, as can be depicted in Fig. 3. A separate surface is then fitted to each of these patches with appropriate continuity constraints. We have preferred the central split approach in our implementation; however, trimmed vertex blend fitting is also possible.

3 Computing the Boundaries of Vertex Blends

In the previous steps the adjacency relationships of the regions have already been defined, and now the set of neighbouring primaries and fillets is already known for each vertex blend. For each fillet a constant or variable width function has been estimated and a pair of longitudinal boundary curves have been traced on the mesh, as shown in Fig. 4a. In this section we present the algorithm that fully defines the setback vertex blends and integrates the independently generated fillet boundaries into a single, consistent segmenting curve network, as depicted in Fig. 4b.

As discussed earlier, the location of vertex blend boundaries significantly influences the quality of fillets and vertex blends that will be fitted later. We define the following postulates to govern our algorithm:

1. *Orthogonal profile curves.* The profile curves terminate the fillets and thus influence their internal parameterisations. If the profile curves are 'close to orthogonal' to the longitudinal boundaries, the u and v iso-parameter lines of the fillet will span a natural, 'close to orthogonal' grid, which is a desired requirement for high-quality fillets.
2. *Tangent continuous spring curves.* Fillet boundaries and spring curves need to be connected with at least tangent continuity in order to obtain smooth trim-loops for the primary surfaces. Having the orthogonality criterion, this also implies that the boundary curves of a vertex blend are also 'close to


(a) Longitudinal feature boundaries

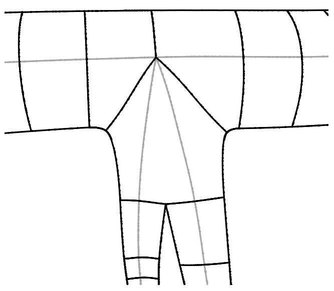

(b) Setback vertex blend

Fig. 4. Reconstruction of vertex blends

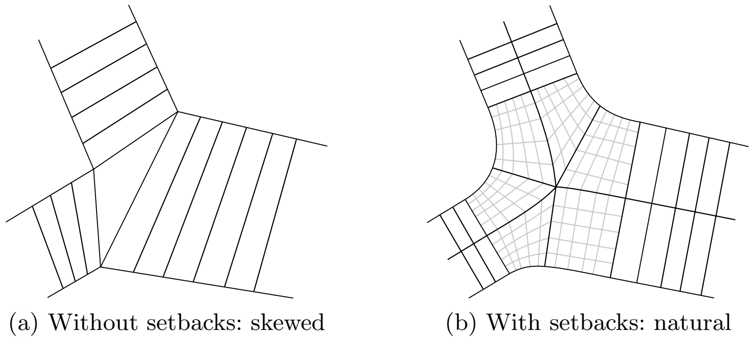


Fig. 5. Iso-parameter lines of transition surfaces

orthogonal' at its corners, which yields a natural local parameterisation for the corner quadrilaterals (Fig. 4b).

3. *Symmetric spring curves.* As a consequence of the central split approach, the internal parameterisation of the vertex blends depends on the position of the centre point and the middle points of the spring curves. If the adjacent setback distances significantly differ, an uneven parameterisation is created and low quality surfaces may be obtained.

Fig. 5a illustrates a case with non-orthogonal profile curves. As setbacks are missing, the boundaries of the fillets intersect at different angles, and the parameter lines of the fillets get skewed. Fig. 5b shows the natural parameterisation of the vertex blend and the incoming fillets when the setbacks and the boundary curves are optimised.

In this section, we first describe methods to estimate the optimal setback distances and construct profiles and spring curves. Finally, some interference issues between independently generated fillets and vertex blends will be discussed.

3.1 Estimation of Setback Distances

The notations used for computing setback distances can be depicted in Fig. 6. Consider the vertex blend with centre point O and n outgoing fillets indexed by $i \in [0, n-1]$ in counter-clockwise order. Any index out of this range should be interpreted in mod n. Let C_i denote the midline curve of the fillet i starting at O and parameterised by quasi-arc length. Similarly, let B_i^R, B_i^L be the right and the left longitudinal boundary curves, respectively. We rely on the bijective mapping defined between the midline and the boundary curves, referred by $\mu_i^R : C_i \mapsto B_i^R$ and $\mu_i^L : C_i \mapsto B_i^L$.

For each fillet there are two *setback distances* to be determined on its two boundary curves, denoted by s_i^R and s_i^L. To make these values comparable, we define them in the same domain, in the parametric space of the midline C_i. Given a setback distance, the related setback point can be expressed by the

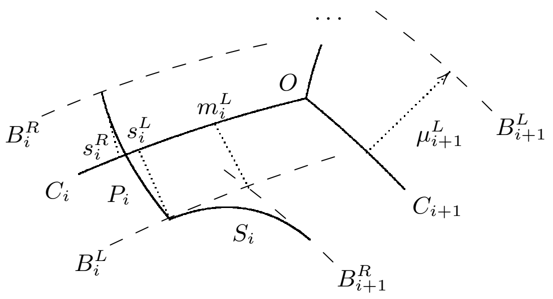


Fig. 6. Notations used in setback estimation

mappings $\mu_i^R(C_i^R(s_i^R))$ and $\mu_i^L(C_i^L(s_i^L))$. In other words, the setback distance is the parametric position of the setback point related to the corresponding point on the midline.

The domain of the possible setback distances is limited by the surrounding geometry. The intersection of the related fillet boundary curves (or their extensions) defines a minimum for the setback distance. For example, the constraint $s_i^L \geq m_i^L$ results from the intersection of the boundary curves B_i^L and B_{i+1}^R, see Fig. 6. The upper limit for the setback distance is defined by the setback at the other end of the fillet, related problems will be investigated later in Sec. 3.3.

To find the optimal setback distances for each (not yet existing) boundary curve of the vertex blend, we define an *error function* that expresses its imperfection in terms of the related setbacks. Suppose $E_i^P(s_i^R, s_i^L)$ describes the deviation of P_i from the perfect profile orthogonal to the midline, while $E_i^S(s_i^L, s_{i+1}^R)$ measures the asymmetry of the spring curve between fillet i and $i+1$. With an additional term $E_i^*(s_i^R, s_i^L)$ we can avoid the setback distances becoming unduly large. So the following total error term is formulated:

$$E(s_0^R, s_0^L, \dots, s_{n-1}^L) = \sum_{i=0}^{n-1} \left(E_i^P(s_i^R, s_i^L) + \lambda E_i^S(s_i^L, s_{i+1}^R) + \nu E_i^*(s_i^R, s_i^L) \right) .$$

After setting the weights λ and ν to balance the partly contradicting constraints, the optimal s_i^R, s_i^L values are defined by minimising this expression.

The error term of the profile curves P_i is estimated using the approximate w_i widths of the setbacks as

$$E_i^P(s_i^R, s_i^L) = \left(\frac{s_i^R - s_i^L}{w_i} \right)^2 ,$$

which intuitively corresponds to the tangent of the angle between the actual and the ideal profile curve, if projected to a local plane. The asymmetry of the spring curves S_i is expressed similarly as the difference of the setback distances over their minima, normalised by the average fillet width $\bar{w}_i = (w_i + w_{i+1})/2$:

$$E_i^S(s_i^L, s_{i+1}^R) = \left(\frac{(s_i^L - m_i^L) - (s_{i+1}^R - m_{i+1}^R)}{\bar{w}_i} \right)^2 .$$

Finally, the term to reduce the bias is defined to increase quadratically while moving away from m_i^R, m_i^L, normalised against the fillet widths:

$$E_i^*(s_i^R, s_i^L) = \left(\frac{s_i^R - m_i^R}{w_i}\right)^2 + \left(\frac{s_i^L - m_i^L}{w_i}\right)^2.$$

The optimal setback distances minimising the total error at a vertex blend can be obtained in different ways. One can observe that $E(s_0^R, s_0^L, \ldots, s_{n-1}^L)$, the total error, is a quadratic function of each of its variables and diverges to $+\infty$ while any of them increases or decreases. Therefore the expression attains its minimum where the partial derivatives vanish:

$$\frac{\partial E}{\partial s_i^R} = 0, \quad \frac{\partial E}{\partial s_i^L} = 0, \quad i \in 0, \ldots, n-1.$$

This reduces to a sparse linear system of $2n$ equations that can be solved. Since all error terms were simple approximations, computing the exact solution might be an overkill. Instead, a few iterations to refine each variable against the error terms yields satisfactory results.

Very short spring curves are undesired, as they worsen the quality of the related surface elements. It is hard to express this constraint algebraically, but such cases can be eliminated in a post-processing step. If a spring curve turns out to be very short, that is, $s_i^L \approx m_i^L$ and $s_{i+1}^R \approx m_{i+1}^R$, the setback distances are snapped to their minima resulting in no spring curve. The effect of such modifications on the adjacent profile curves is negligible.

3.2 Construction of the Boundary Curves

Once the setback distances are available, the setback points are also defined. Based on this, we define the boundary curves of the vertex blend using interpolating cubic B-splines. Each curve interpolates 5 so-called marker points $(Q_0, Q_1, \ldots, Q_4)$ that lie on the mesh. Q_0 and Q_4 are the endpoints of the segment. The two tangents of the curve are set by a well-known technique using two parabolas spanned by (Q_0, Q_1, Q_2) and (Q_2, Q_3, Q_4), respectively. These constraints still allow infinitely many boundary segments. In order to find a simple and natural construction, we revisit double-quadratic arcs, that are composed of two twisted parabolas sharing a common tangent line at the middle. The final B-spline representation will closely approximate this double-quadratic arc, but will also assure C^2 continuity at the middle marker.

Spring curves. In addition to the endpoints, the end-tangents of the spring curves are inherited from the fillet boundaries, as determined by the tangent continuity rule. Below we overview the possible cases and construct the spring curves accordingly, see Fig. 8.

The endpoints P_0, P_1 and tangents $\mathbf{t}_0$, $\mathbf{t}_1$ define two lines that are typically not coplanar. Let $B_0 = P_0 + \lambda_0 \mathbf{t}_0$ and $B_1 = P_1 + \lambda_1 \mathbf{t}_1$ denote the endpoints of their perpendicular transversal. Different configurations are identified based

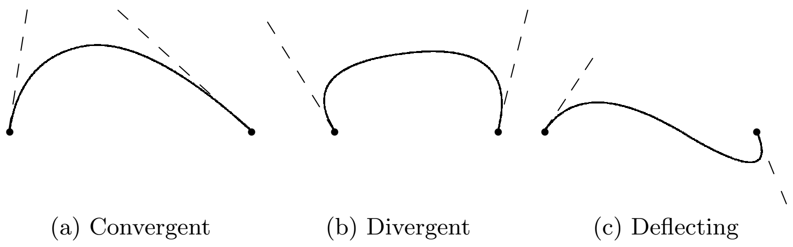


Fig. 7. Configuration of end tangents

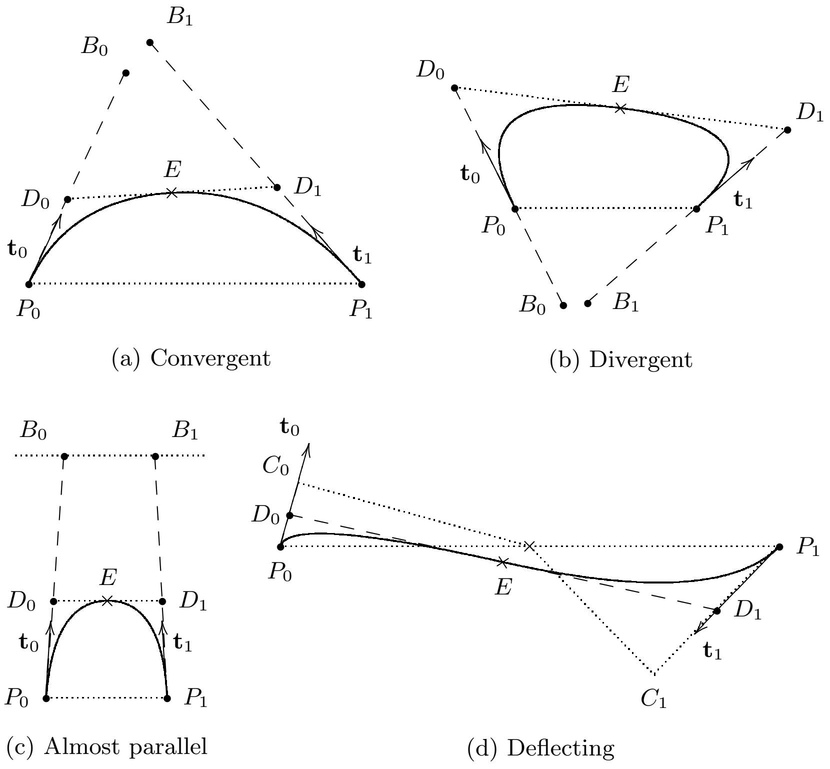
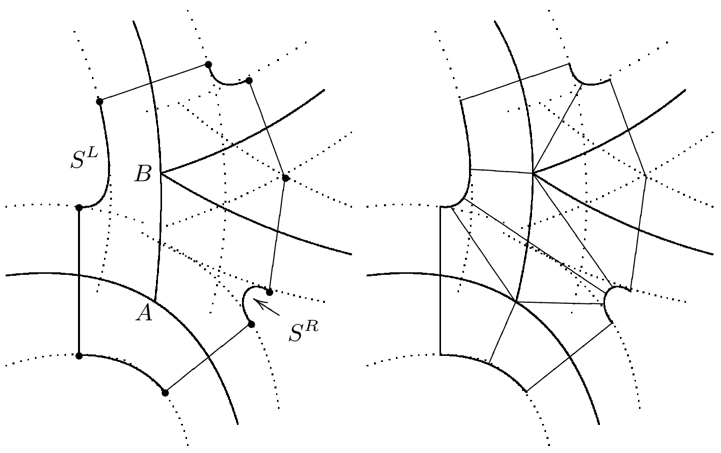


Fig. 8. Construction of spring curves

on the signs of λ_0 and λ_1, see Fig. 7. If both $\lambda_0 > 0$ and $\lambda_1 > 0$, we call the case *convergent* (a); the $\lambda_0 < 0$ and $\lambda_1 < 0$ case is referred as *divergent* (b); the rest corresponds to the case called *deflecting* (c). Special rules are applied for (almost) parallel tangents.

The details of the spring curve construction for the convergent case can be depicted in Fig. 8a. Points D_0 and D_1 split segments P_0B_0 and P_1B_1 at equal



(a) Composite vertex blend (b) Multi-central split

Fig. 9. Resolution for interfering vertex blends

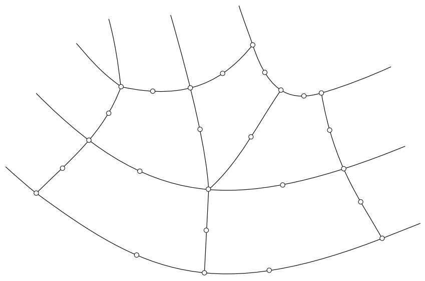


Fig. 10. B-spline boundaries created for a vertex blend

proportion, which is chosen depending on the fullness of the curve segment. Let E denote the midpoint of D_0D_1, then the spring curve is composed of two parabolic arcs defined by P_0D_0E and ED_1P_1.

The construction of a divergent spring curve is similar, see Fig. 8b. The difference is that point D_i is defined to be on the opposite side of P_i with respect to B_i, $i \in \{0, 1\}$. Note that divergent spring curves tend to occur rarely in practice.

Almost parallel tangents result in unstable, distant transversals, which has to be addressed separately, as shown in Fig. 8c. To overcome this issue, we restrict points B_0, B_1, to remain within a limited distance, which is typically defined by some related geometric entity, e.g. the centre of the vertex blend. As a consequence of this, the fullness of the resulting spring curve is also limited.

In the case of deflecting tangents the actual transversal is ignored, see Fig. 8d. Instead, the midpoint of P_0P_1 is projected to the tangent lines, defining C_0 and C_1. Sections P_0C_0 and P_1C_1 are split proportionally to obtain D_0 and D_1, again leaving some freedom of choosing the appropriate expansion for the spring curve. The rest of the construction is identical to the other cases.

Profile curves. The construction of profile curves follows the same logic as above; the only difference is that instead of a twisted double-quadratic segment, here we enforce the profile into a single plane defined by the setback points $\mu_i^R(C_i(s_i^R))$, $\mu_i^L(C_i(s_i^L))$, and the midpoint $C_i((s_i^R+s_i^L)/2)$. The tangents at the endpoints will be perpendicular to the locally estimated surface normals. For a more complex connecting feature this B-spline may significantly deviate from its projection on the mesh, and additional interpolating points may need to be inserted to refine its shape, however, for the profile of fillets five interpolation points are sufficient.

Fig. 10 shows a vertex blend, where two concave and one convex fillets run together. The geometric configuration forces to add two spring curves, thus the vertex blend becomes 5-sided. The markers of the B-spline boundaries created by the above algorithm are also shown.

3.3 Resolving Interference Issues

It is an intrinsic characteristic of functional decomposition that it builds up the entire structure automatically in one go and no 'historical' sequence of operations is created as in constructive CAD. The fillet boundaries and the setback distances have been computed independently and it may happen that unwanted interferences, e.g. overlaps occur, which need to be removed in a *post-processing phase* to get a valid segmenting curve network. In this section we deal with detecting and eliminating such self-intersections using special setback vertex blends. Two typical examples are presented.

Interfering vertex blends. If two vertex blends are connected with a very short fillet, there is a good chance that the setbacks overlap. A clear indication of this problem is that the sum of the setback distances at the two ends of the fillet exceeds the length of the related midline.

A proposed resolution of interfering vertex blends is illustrated in Fig. 9. Instead of two separate entities we create a *composite vertex blend* that covers both vertex areas and excludes the short fillet region. The setbacks and the boundary curves of the composite vertex blend are computed in the same manner as for a regular one but all incoming fillets are treated as if they joined at a single vertex. As a result, spring curves S^R and S^L are constructed corresponding to the short fillet between AB, see Fig. 9a. In this example, a degree 3 (vertex A) and a degree 4 (vertex B) were merged to a composite degree 5 vertex blend – with 5 profile curves and 4 spring curves, i.e. altogether a 9-sided patch is to be generated.

After the construction of the boundaries, the internal subdivision of the composite vertex blend has to be created. One possible way is to apply a central split as if it was a regular vertex blend. However, such composite vertex blends are usually elongated, which may lead to distorted quadrilaterals. For this reason we propose an alternative structure, called *multi-central split.* As it can be depicted

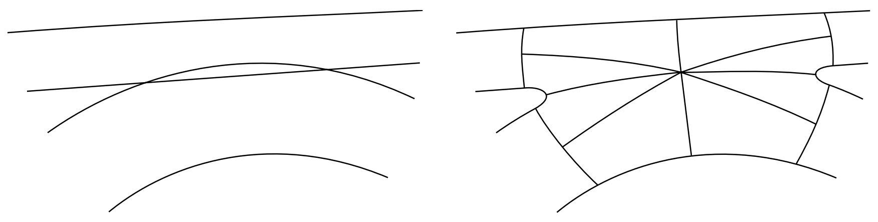

(a) Overlapping fillets (b) Insertion of a vertex blend

Fig. 11. Resolution of overlapping fillets

in Fig. 9b, this structure is also made up of quadrilaterals only, but here the two central splits are directly connected, cancelling out the original short fillet.

The problem of interference is not restricted to a single pair of vertex blends and it is possible that more than two vertex blends have to be merged into a composite vertex blend. The proposed technique is capable of handling such cases as well. The use of multi-central split is absolutely inevitable in complex cases with more than 2 vertex blends merged.

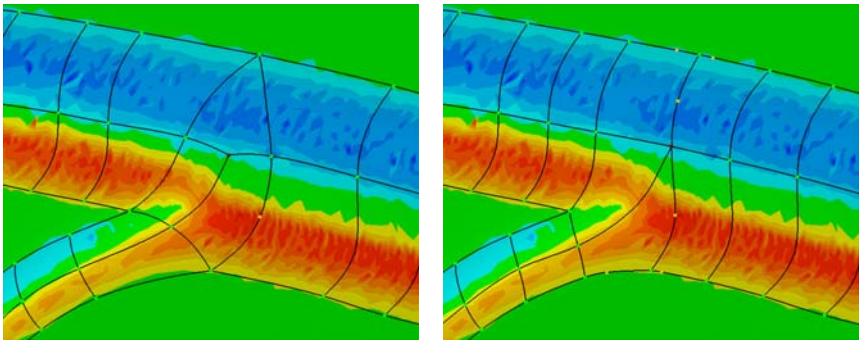
Overlapping features. Another common case of interference occurs when segmentation indicates that two fillets overlap, i.e. the individually traced fillet boundary curves intersect each other, see Fig. 11a. The problem is not originating in the vertex blend construction; but it is caused by the lack of vertex blends in the neighbourhood. This situation is resolved by inserting a new vertex blend in the post-processing phase.

Overlap can be detected by the intersection of fillet boundaries. Taking their polyline representation (projection) on the triangular mesh, if they intersect, the related polylines also share a triangle or a vertex, so the search to locate them can be performed in an efficient way.

Once such an intersection is found, we insert a special vertex blend as illustrated in Fig. 11b. This connecting surface element fully covers the area of the overlap, but retains the fillets outside the overlapping area. It may make sense to slightly overextend the vertex blend and insert spring curves using the setback mechanism. In this way, more space is created locally and a smoother connection is produced. Without spring curves the primary surface fitting may be unstable in the vicinity of the narrow, spike-like corner since none or only a small number of data points exist there. With the spring curves, the vertex blend will connect where the primary surfaces are already stable. In Fig. 11, four fillets meet at an 8-sided vertex blend. Multi-central split is also possible.

4 Examples

The presented vertex blend algorithm has been partly integrated into the segmenting module of a well-known commercial system, Geomagic Studio [21], and has been used to digitally reconstruct a wide range of parts in various industries.


(a) Curvature map on the polygonal mesh

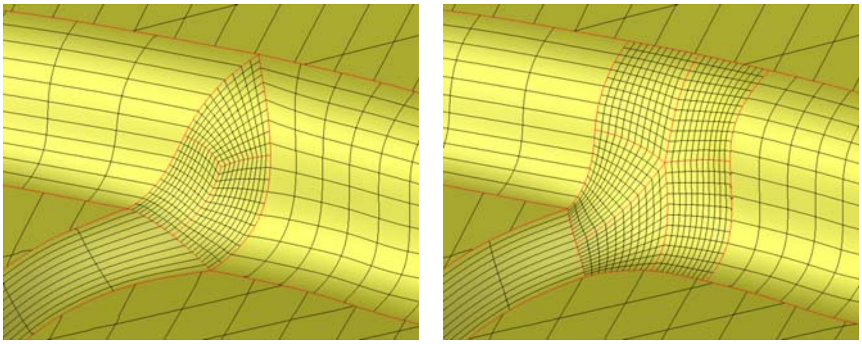
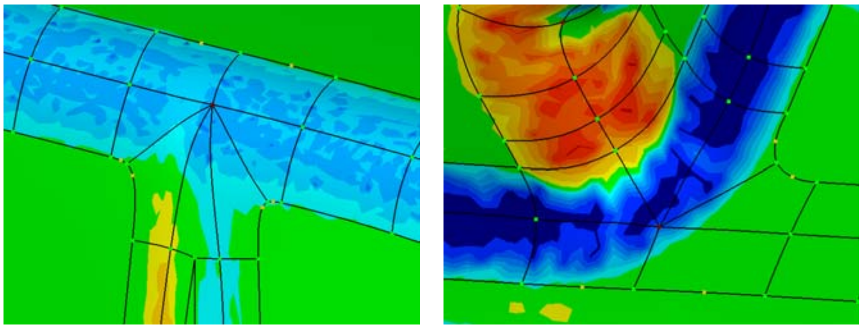

(b) Iso-parameter lines of the final CAD model

Fig. 12. Comparison of vertex blends without/with setbacks

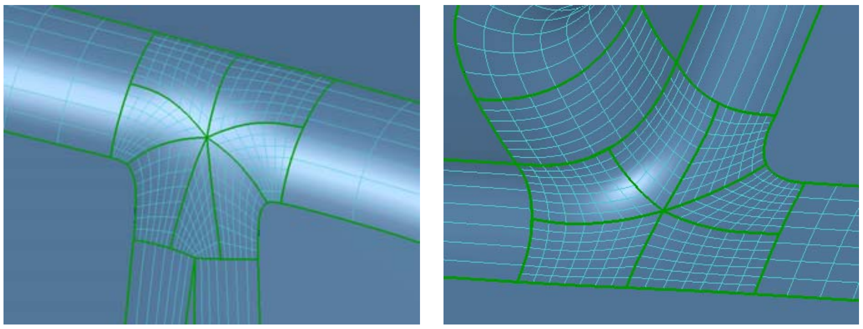
The previous, rendered pictures and the forthcoming examples have also been generated by Geomagic Studio.

First we show the difference between vertex blends without and with setbacks, as can be depicted in Fig. 12. The left pictures show a vertex blend that connects free-form steps without applying setbacks, while on the right side a properly parameterised representation is shown applying setbacks. In the second case, the iso-parameter lines (see Fig. 12b) are perfectly aligned with the principal curvature lines of the fillets. Fig. 13 shows two further examples of setback vertex blends. On the left side, two small fillets merge into a large one, altogether 4 fillets are connected with a 7-sided vertex blend. On the right side a convex-concave junction with and additional artificial (flat) fillet is shown, that forms a 6-sided vertex blend.

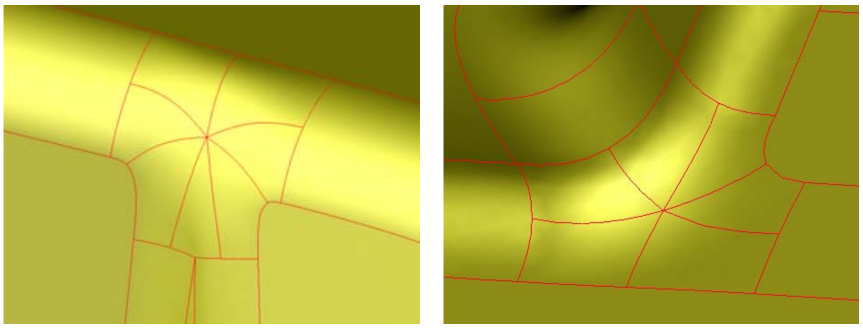
The example in Fig. 14 shows the reconstruction steps of a real industrial part. The curvature map estimated on the polygonal mesh (a) is followed by the automatically computed segmenting curve network (b), and the final CAD model (c). The structure of this part has been generated by means of the segmentation method described in this paper. The pictures show the corresponding boundary representation of the created CAD model including the automatically generated



(a) Curvature map and segmenting curve network on the mesh



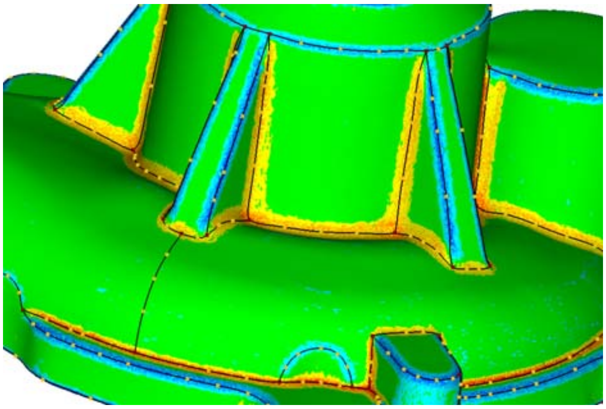
(b) Iso-parameter lines of the generated surfaces



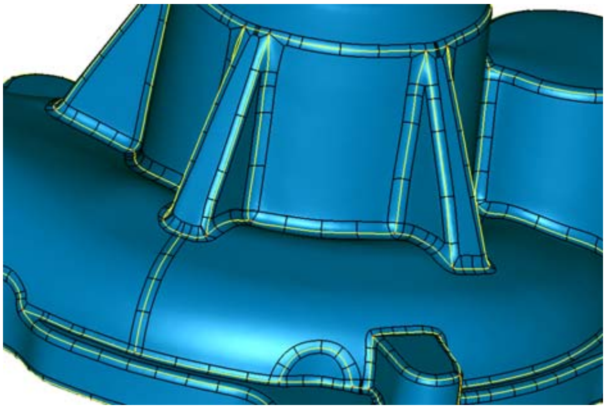
(c) Automatically created CAD model

Fig. 13. Examples of setback vertex blends

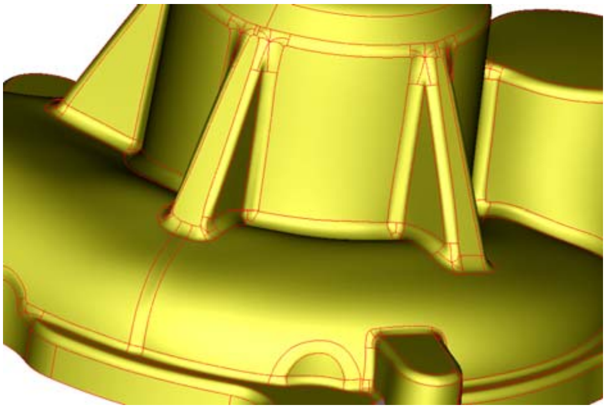
setback vertex blends with central splitting. (Note: for 4-sided vertex blends central splitting does not need to be applied.)



(a) Curvature map and the feature skeleton on the mesh



(b) Segmenting curve network and the polygonal mesh



(c) Automatically generated CAD model

Fig. 14. Reconstruction steps of an industrial part

5 Conclusion

The process of creating functionally decomposed CAD models from measured data consists of two basic phases [17]. In the first phase, the topological structure of the model is extracted using a multi-step segmentation procedure, that produces a consistent segmenting curve network being isomorphic to the edge structure of the final CAD model. In the second phase, this structure drives the geometric computations to classify and fit surfaces by a well-defined dependency of (i) primary surfaces, (ii) fillets, and (iii) vertex blends, and trim and stitch them together.

The algorithm presented here is the final touch to build up a segmenting network over a polygonal mesh; the created structure of vertex blends determines the extent of fillets and closes the corner sections of primary surface loops. In order to stitch together independently traced fillet boundaries in a consistent manner, the use of setback type vertex blends was proposed. Setback vertex blends help to resolve difficult topological configurations and make it possible to fit smooth, connecting surfaces with optimised parameterisation.

Acknowledgements

The authors would like to acknowledge the innovative atmosphere experienced at Geomagic Inc., and the inspiring technical discussions with Michael A. Facello and the other members of the international engineering team. Support from the Department of Computer Science at the Budapest University of Technology and Economics is also highly appreciated. The research efforts have been partly supported by two NSF-SBIR grants, namely Award #0450230, "Creating functionally decomposed surface models from measured data" and Award #0521838, "Applications of Morse theory in reverse engineering".

References

1. Marks, P.: Capturing a Competitive Edge through Digital Shape Sampling & Processing (DSSP). Blue Book Series. Society of Manufacturing Engineers (2005)
2. Botsch, M., Pauly, M., Kobbelt, L., Alliez, P., Lévy, B., Bischoff, S., Rössl, C.: Geometric modeling based on polygonal meshes. In: Eurographics Tutorial Notes, Prague, Czech Republic, Eurographics (2007)
3. Várady, T., Martin, R.R., Cox, J.: Reverse engineering of geometric models – an introduction. Comput. Aided Des. 29(4), 225–268 (1997)
4. Braid, I.C.: Non-local blending of boundary models. Comput. Aided Des. 29(2), 89–100 (1997)
5. Várady, T., Hoffmann, C.M.: Vertex blending: Problems and solutions. In: Dælen, M., Lyche, T., Schumaker, L.L. (eds.) Mathematical Methods for Curves and Surfaces II, Vanderbilt, pp. 501–527. University Press, Nashville (1998)
6. Besl, P.J., Jain, R.C.: Segmentation through variable-order surface fitting. IEEE Trans. Pattern Anal. Mach. Intell. 10(2), 167–192 (1988)

7. Sapidis, N.S., Besl, P.J.: Direct construction of polynomial surfaces from dense range images through region growing. ACM Trans. Graph. 14, 171–200 (1995)
8. Vieira, M., Shimada, K.: Surface mesh segmentation and smooth surface extraction through region growing. Comput. Aided Geom. Des. 22(8), 771–792 (2005)
9. Lai, Y.K., Zhou, Q.Y., Hu, S.M., Wallner, J., Pottmann, H.: Robust feature classification and editing. IEEE Trans. Vis. Comp. Graphics 13(1), 34–45 (2007)
10. Benkö, P., Martin, R.R., Várady, T.: Algorithms for reverse engineering boundary representation models. Comput. Aided Des. 33, 839–851 (2001)
11. Fitzgibbon, A.W., Eggert, D.W., Fisher, R.B.: High-level CAD model acquisition from range images. Comput. Aided Des. 29, 321–330 (1997)
12. Petitjean, S.: A survey of methods for recovering quadrics in triangle meshes. ACM Comput. Surv. 34(2), 211–262 (2002)
13. Benkö, P., Kós, G., Várady, T., Andor, L., Martin, R.R.: Constrained fitting in reverse engineering. Comput. Aided Geom. Des. 19(3), 173–205 (2002)
14. Eck, M., Hoppe, H.: Automatic reconstruction of B-spline surfaces of arbitrary topological type. In: Computer Graphics. SIGGRAPH, pp. 325–334 (1996)
15. Heckbert, P.S., Garland, M.: Survey of polygonal surface simplification algorithms. In: Multiresolution Surface Modeling Course. SIGGRAPH (1997)
16. Várady, T.: Automatic procedures to create CAD models from measured data. Computer-Aided Design and Applications 5(5), 577–588 (2008)
17. Várady, T., Facello, M.A., Terék, Z.: Automatic extraction of surface structures in digital shape reconstruction. Comput. Aided Des. 39(5), 379–388 (2007)
18. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. 28, 511–533 (2002)
19. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. Discrete Comput. Geom. 30, 87–107 (2003)
20. http://www.rapidform.com/
21. http://www.geomagic.com/

Learning a Self-organizing Map Model on a Riemannian Manifold

D.J. Yu[1,2], E.R. Hancock[2], and W.A.P. Smith[2]

[1] School of Computer Science, Nanjing University of Science and Technology, China
[2] Department of Computer Science, University of York, UK
njyudj@mail.njust.edu.cn, {erh,wsmith}@cs.york.ac.uk

Abstract. We generalize the classic self-organizing map (SOM) in flat Euclidean space (linear manifold) onto a Riemannian manifold. Both sequential and batch learning algorithms for the generalized SOM are presented. Compared with the classical SOM, the most novel feature of the generalized SOM is that it can learn the intrinsic topological neighborhood structure of the underlying Riemannian manifold that fits to the input data. We here compared the performance of the generalized SOM and the classical SOM using real 3-Dimensional facial surface normals data. Experimental results show that the generalized SOM outperforms the classical SOM when the data lie on a curved Riemannian manifold.

Keywords: SOM, Riemannian Manifold, Manifold Learning, Pattern Recognition.

1 Introduction

The self-organizing map (SOM) [1], also known as the topology-preserving map, has been a topic of sustained research activity for several decades. As an unsupervised learning strategy, the SOM can automatically learn the underlying topological neighborhood structure of the input space by applying a very simple adaptation rule. The basic idea of the SOM is to locate and adapt the best-matching unit (BMU) together with its topological neighbors with respect to the current input vector. This is done so as to adaptively learn the topological neighborhood structure of the input space that fits to the properties of the input data.

Because of its simplicity and effectiveness, the SOM has been widely and successfully applied in many fields such as data mining [2], biomedical applications [3,4], content-based information retrieval [5], dynamical system identification [6] and data visualization [7]. However, the classical SOM can only reveal flat Euclidean topological neighborhood structure in the input space, and will fail to discover a non-Euclidean topological neighborhood structure of the input space [8].

To solve this problem, many researchers [8,9] have investigated how to combine the classical SOM with kernel methods and this has led to the Kernel-SOM. In Kernel-SOM, data in original input space are firstly nonlinearly transformed to a higher, possibly infinite dimensional space, and then the classical SOM is applied to the transformed data. The key issue here is that the learning procedure on the transformed higher dimensional data can be indirectly derived, by using the kernel trick, from the learning procedure on the original input data. The resulting Kernel-SOM can preserve a non-Euclidean topological neighborhood structure which corresponds to the distance metric in the transformed higher dimensional space. Different non-Euclidean topological neighborhood structures can be learned by applying different kernel functions, i.e. the resulting non-Euclidean topological neighborhood structure is kernel function dependent. Thus, when the data lie on a specific Riemannian manifold, the non-Euclidean topological neighborhood structure learned by Kernel-SOM is **not** the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold on which the data reside.

In fact, in many real applications, the data may reside on some specific Riemannian manifold. For example, an unit surface normal of an object in $\mathbb{R}^3$ is a data point lying on unit 2-sphere manifold S^2. A set of unit surface normals of an object in $\mathbb{R}^3$ is a data point lying on Riemannian manifold $S^2(n)$, where n is the cardinality of the set. To learn the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold from such data, both the classic SOM and the Kernel-SOM do not apply. This paper aims to overcome this problem by generalizing the classical SOM onto a Riemannian manifold.

To the best of our knowledge, studies on this topic are far less extensive. Ritter [10] proposed a method for learning a SOM in non-Euclidean space. However, it can only arrange the output neurons in a spherical or hyperbolic lattice topology, and still uses the Euclidean distance metric for locating the BMU. Shi [11] proposed a geodesic distance based SOM, referred to as GDBSOM. The GDBSOM uses a geodesic distance metric instead of a Euclidean distance metric to locate the BMU. However, when updating the codebook vectors, it does **not** consider how to guarantee that the updated codebook vectors still reside on the underlying manifold. Simila [12] incorporated manifold learning technique (such as LLE) into the classical SOM and proposed the M-SOM. In M-SOM, manifold learning is first applied to learn the internal coordinates on the underlying manifold, next the codebook vectors of the SOM are adapted in both the internal and observation coordinates, with similarities defined on the internal coordinates.

In this paper, we approach with this problem in a quite different style by directly generalizing the classical SOM onto a Riemannian manifold. More specifically, to locate the BMU, the Riemannian geodesic distance metric is applied. In the adaptation step, codebook vectors are adapted along the intrinsic geodesic curve with the aid of the Riemannian *Exponential* and *Log* maps.

The structure of the paper is as follows. In Section 2 we give a brief introduction to the classic SOM. Section 3 presents some preliminaries concerning Riemannian manifolds. In addition, we also detail how to implement the Riemannian *Exponential* and Riemannian *Log* maps on the specific Riemannian manifold $S^2(n)$. We present both the *sequential* and *batch* learning algorithms for learning a SOM on a Riemannian manifold in Section 4. Experiments are carried out in Section 5 using real 3-Dimensional facial needle-map data. Performance comparison between the classical SOM and the generalized SOM are also performed. We conclude the paper in Section 6 by offering conclusions and direction for future work.

2 Self-organizing Map

The classical self-organizing map (SOM) in a flat Euclidean space (linear manifold) consists of a regular output grid (usually 1- or 2-dimensional), onto which the distribution of input vectors is projected nonlinearly [1]. The mapping tends to preserve the topological-metric relations between input vectors, i.e., similar inputs fed into the SOM will give similar outputs.

A SOM can be represented as a set of output neurons, denoted by $\{i\}_{i=1}^{K}$, on the output layer, where K is the number of neurons on the output layer. Each output neuron i is fully connected to all the input neurons and contains a d-dimensional *codebook* vector $\mathbf{w}_i \in \mathbb{R}^d$ (also referred to as a *prototype* vector). A SOM can be trained by iterative sequential or batch learning algorithms.

2.1 Sequential Learning Algorithm

Let $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^{N}$ be the learning dataset. One round of the sequential learning algorithm for SOM is briefly described as follow.

1) Choose the current learning sample: Randomly select a sample from the dataset $\{\mathbf{x}_i\}_{i=1}^{N}$ as the *current* learning sample $\mathbf{x}(t)$.

2) Locate the best-matching unit (BMU): The output neuron whose codebook vector is closest to the current learning sample $\mathbf{x}(t)$ is referred as the BMU, denoted by $c(\mathbf{x}(t))$ and satisfying

$$c(\mathbf{x}(t)) = \arg\min_{1\leq i\leq K} \parallel \mathbf{x}(t) - \mathbf{w}_i(t) \parallel. \tag{1}$$

3) Update the codebook vectors: The codebook vectors of the BMU and its topological neighborhood output nodes are updated using the following simple rule

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \underbrace{\alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot (\mathbf{x}(t) - \mathbf{w}_i(t))}_{\triangle}. \tag{2}$$

where t denotes the learning step, which decreases monotonically with the learning steps and $\alpha(t)$ is the learning rate. The Gaussian neighborhood function defined as

$$h_{c(\mathbf{x}(t)),i}(t) = exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{c(\mathbf{x}(t))}\|^2}{2\sigma^2(t)}\right) \tag{3}$$

where $\mathbf{r}_i$ and $\mathbf{r}_{c(\mathbf{x}(t))}$ are the vector locations of neuron i and neuron $c(\mathbf{x}(t))$ on the output grid, respectively and $\sigma(t)$ is the width of the neighborhood function, which is also decreasing monotonically with the learning steps. Notice that the

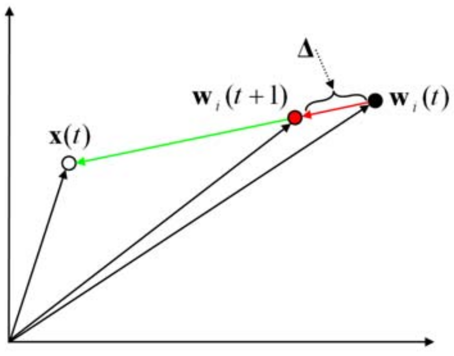


Fig. 1. Sequential adaptation of codebook vector in classical SOM

neighborhood function is not limited to be a Gaussian, and alternative neighborhood functions can also apply [1].

The aim of the updating process in (2) is to move the codebook vector $\mathbf{w}_i(t)$ closer to the current learning sample $\mathbf{x}(t)$ with respect to $\alpha(t)$ and $h_{c(\mathbf{x}(t)),i}(t)$. Fig.1. illustrates the effect of the update Equation (2) in a 2-dimensional Euclidean space. With the update equation, the black point corresponding to codebook vector $\mathbf{w}_i(t)$ is moved to the red point with a distance of $\|\triangle\|$ along the direction of $\mathbf{x}(t) - \mathbf{w}_i(t)$. The vector corresponding to the red point is the resulting updated codebook vector $\mathbf{w}_i(t+1)$.

2.2 Batch Learning Algorithm

In the sequential learning algorithm, only a single learning sample is fed to the SOM at each learning step, and the codebook vectors of the BMU together with its topological neighborhood nodes are updated. However, in a batch learning algorithm, the entire learning dataset is fed to the SOM before any adaptations are made. We briefly describe one round of batch learning as follows:

1) The entire learning dataset is first partitioned into Voronoi regions $\{V_j\}_{j=1}^K$ of the codebook vectors $\{\mathbf{w}_j\}_{j=1}^K$. More specifically, sample $\mathbf{x}_i \in \{\mathbf{x}_i\}_{i=1}^N$ is partitioned into Voronoi region V_j if the BMU of $\mathbf{x}_i$ is the output neuron j.

2) Let n_j be the number of samples in the Voronoi region V_j, then the average of the samples contained in V_j, denoted by $\overline{\mathbf{x}}_j$, can be computed, i.e.

$$\overline{\mathbf{x}}_j = \frac{1}{n_j}\sum_{p=1}^{n_j}\mathbf{x}_p. \tag{4}$$

where $\mathbf{x}_p \in V_j$, $1 \le p \le n_j$.

3) The codebook vectors are updated by using a simple linear weighted average scheme:

$$\mathbf{w}_i(t+1) = \frac{\sum_{j=1}^K h_{ji}(t)\cdot n_j \cdot \overline{\mathbf{x}}_j}{\sum_{j=1}^K h_{ji}(t)\cdot n_j}. \tag{5}$$

The batch learning algorithm can effectively accelerate the learning process and lessen the impact of the order of the sample data fed to the SOM. When the size of the learning dataset is large, batch learning is to be preferred.

3 Riemannian Manifolds

Let M be a Riemannian manifold. A Riemannian metric on M is a smoothly varying inner product $< \cdot, \cdot >$ on the tangent plane $T_w M$ at each point $w \in M$. The norm of a vector $v \in T_w M$ is given by $\|v\| =< \cdot, \cdot >^{\frac{1}{2}}$. Given a smooth curve segment on M, its length is computed by integrating the norm of the unit tangent vectors along the curve. Let w and x be two points lying on M. The Riemannian distance between them, denoted by $d(w, x)$, is defined as the minimum of lengths over all possible smooth curves between w and x. The *geodesic* is a smooth curve that locally minimizes the length between two points on M.

3.1 Riemannian *Exponential* Map and Riemannian *Log* Map

Let $v \in T_w M$ be a vector on the tangent plane to M at $w \in M$ and $v \neq 0$. γ_w^v be the geodesic that passes through point w (a.k.a. the base point) in the direction of v . The Riemannian *Exponential* map of v at base point w, denoted by $\mathrm{Exp}_w(v)$, maps v to the point, say x, on M along the *geodesic* at distance $\|v\|$ from w, i.e.

$$x = \mathrm{Exp}_w(v). \tag{6}$$

Note that the *Exponential* map preserves the geodesic distance from the base point to the mapped point, i.e. $d(w, x) = d(w, \mathrm{Exp}_w(v)) = \|v\|$.

The Riemannian *Log* map is the inverse of Riemannian *Exponential* map:

$$v = \mathrm{Log}_w(x). \tag{7}$$

We illustrate the concepts described above by using a unit 2-sphere manifold S^2. Riemannian *Exponential* map and *Log* map on S^2 at base point w are illustrated in Fig.2.

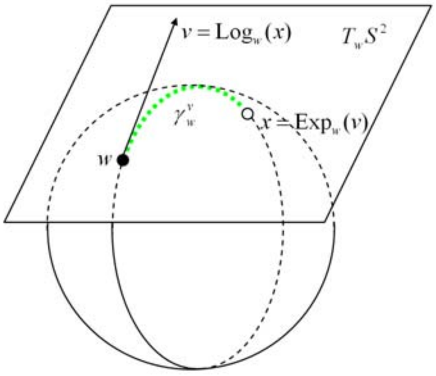


Fig. 2. *Exponential* and *Log* maps on S^2

3.2 Intrinsic Average and Weighted Intrinsic Average

The intrinsic average of the N points $\{x_1, x_2, \cdots, x_N\}$ lying on a Riemannian manifold M is defined as

$$\begin{aligned}\overline{x} &= \mathbf{IntrinsicAvg}(x_1; x_2; \cdots; x_N) \\ &= \arg\min_{x \in M} \sum_{i=1}^{N} (d(x, x_i))^2.\end{aligned} \tag{8}$$

Pennec [13] first proposed an iterative gradient-descent method to solve the aforementioned minimization problem

$$\overline{x}_{j+1} = \mathrm{Exp}_{\overline{x}_j}\left(\frac{\tau}{N}\sum_{i=1}^{N} \mathrm{Log}_{\overline{x}_j}(x_i)\right). \tag{9}$$

where τ is the step size. The uniqueness of the solution can be guaranteed when the data are well localized [14].

Let w_i be the weight value of x_i, $w_i \geq 0$, $1 \leq i \leq N$. Likewise, the weighted intrinsic average

$$\overline{x} = \mathbf{WIntrinsicAvg}(w_1, x_1; w_2, x_2; \cdots; w_N, x_N). \tag{10}$$

can be computed using the following iteration equation:

$$\overline{x}_{j+1} = \mathrm{Exp}_{\overline{x}_j}\left(\frac{\tau}{\sum_{i=1}^{N} w_i}\sum_{i=1}^{N} w_i \cdot \mathrm{Log}_{\overline{x}_j}(x_i)\right). \tag{11}$$

3.3 Embedding

Nash [15] has proved that every Riemannian manifold M can be isometrically embedded into some Euclidean space $\mathbb{R}^d$, i.e., there exists embedding mapping $\Phi : M \to \mathbb{R}^d$, which embed M into its ambient Euclidean space $\mathbb{R}^d$. Under the embedding Φ, points on M can be depicted by corresponding vectors in the Euclidean space $\mathbb{R}^d$. For example, a unit 2-sphere manifold S^2 can be embedded into a 3-dimensional Euclidean space by defining embedding mapping $\Phi : M \to \mathbb{R}^3$. Under this embedding, data point w lying on manifold S^2 can be depicted by a corresponding vector $\mathbf{w}$ in 3-dimensional Euclidean space. From now on, data point on M, unless otherwise stated, is depicted by the corresponding vector in the ambient Euclidean space $\mathbb{R}^d$.

3.4 Riemannian *Exponential* Map and *Log* Map on $S^2(n)$

We commence by considering how to implement Riemannian *Exponential* map and Riemannian *Log* map on a unit 2-sphere manifold S^2, which has been embedded into the Euclidean space $\mathbb{R}^3$.

Let point $\mathbf{p}_0 = (0, 0, 1)^T \in S^2$ be the base point, then a vector on the tangent plane $T_{\mathbf{p}_0}S^2$ can be written in the form of $\mathbf{v} = (v_x, v_y)^T$. Notice that here we choose the x and y axes of the ambient Euclidean space $\mathbb{R}^3$ as the coordinate system of $T_{\mathbf{p}_0}S^2$. Thus the Riemannian *Exponential* map on S^2 with base point $\mathbf{p}_0$ is given as [16]

$$\mathrm{Exp}_{\mathbf{p}_0}(\mathbf{v}) = \left(v_x \cdot \frac{\sin \|\mathbf{v}\|}{\|\mathbf{v}\|}, v_y \cdot \frac{\sin \|\mathbf{v}\|}{\|\mathbf{v}\|}, \cos \|\mathbf{v}\| \right)^T . \tag{12}$$

The corresponding Riemannian *Log* map for a point $\mathbf{q} = (q_x, q_y, q_z)^T$ on S^2 is given by

$$\mathrm{Log}_{\mathbf{p}_0}(\mathbf{q}) = \left(q_x \cdot \frac{\beta}{\sin \beta}, q_y \cdot \frac{\beta}{\sin \beta} \right)^T . \tag{13}$$

where $\beta = \arccos(q_z)$. Notice that the antipodal point $-\mathbf{p}_0$ is not in the domain of the Log map.

As stated above, a unit vector $\mathbf{p} \in \mathbb{R}^3$ can be considered as a point on the manifold S^2. Thus, a matrix $\mathbf{U} \in \mathbb{R}^{n\times 3}$, in which each row is a unit vector, can be considered as a point on manifold $S^2(n) = \prod_{i=1}^{n} S^2$. The *Exponential* and *Log* maps for $S^2(n)$ are just the direct products of n copies of the corresponding maps for S^2.

4 Learning SOM on Riemannian Manifold

In this section, we consider how to learn a SOM from data lying on a Riemannian manifold M. Note that the Riemannian manifold M has been embedded into an appropriate Euclidean space R^d by the embedding mapping $\Phi : M \to R^d$.

4.1 Sequential Learning Algorithm

First, we illustrate why the classic SOM can not *accurately* learn the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold on where the data reside. Let us commence by starting from a simple case where the data reside on the Riemannian manifold S^2. Note that S^2 has been embedded into the Euclidean space $\mathbb{R}^3$ as shown in Fig.3.

To learn a SOM on S^2, if we use the classical updating equation (2), then the resulting effect is to move the *black* point corresponding to vector $\mathbf{w}_i(t)$ to the *blue* point corresponding to vector $\mathbf{w}_i^{'}(t+1)$ by a distance of $\|\triangle\|$ along the direction of $\mathbf{x}(t) - \mathbf{w}_i(t)$, as shown in Fig.3. Note that the resulting data point corresponding to the updated codebook vector $\mathbf{w}_i^{'}(t+1)$ is **not** guaranteed to still reside on the manifold S^2. Thus the classical SOM fails to accurately learn the *intrinsic* geometric and topological properties of data lying on S^2.

The correct procedure for moving $\mathbf{w}_i(t)$ closer to the current learning sample $\mathbf{x}(t)$ is to move the *black* point corresponding to vector $\mathbf{w}_i(t)$ to the *red* point by a distance of $\|\triangle\|$ along the *geodesic* between $\mathbf{w}_i(t)$ and $\mathbf{x}(t)$. This can be implemented by three successive steps as follows

1) Map point $\mathbf{x}(t)$ to the vector $\mathbf{v}$ on the tangent plane using Riemannian *Log* map.

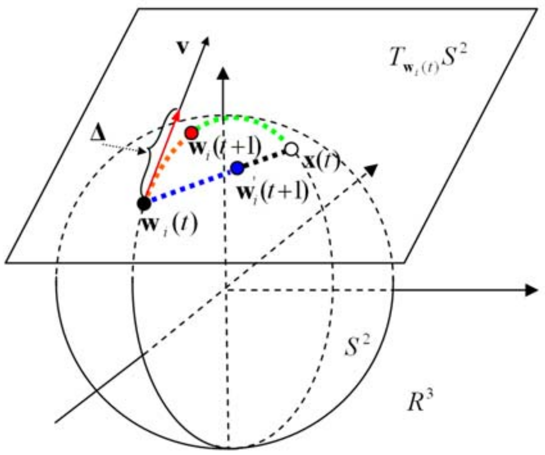


Fig. 3. Adaptation of codebook vector on S^2

2) Calculate $\triangle = \alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot \mathrm{Log}_{\mathbf{w}_i(t)}(\mathbf{x}(t))$, the magnitude and the direction of adjustment, with respect to $\alpha(t)$ and $h_{c(\mathbf{x}(t)),i}(t)$.

3) Map the $\triangle$ back to the point on manifold S^2 using the Riemannian *Exponential* map.

The aforementioned 3 steps can be formulated as the following update equation

$$\mathbf{w}_i(t+1) = \overbrace{\mathrm{Exp}_{\mathbf{w}_i(t)}\left(\underbrace{\alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot \overbrace{\mathrm{Log}_{\mathbf{w}_i(t)}(\mathbf{x}(t))}^{(\text{step 1})}}_{(\text{step 2}),\triangle}\right)}^{(\text{step 3})}. \tag{14}$$

While the update equation(14) can move the codebook vector closer to the current learning example, on the other hand, it can also guarantee the resulting codebook vector still reside on S^2. Notice that the *BMU* of $\mathbf{x}(t)$ is located by using the Riemannian *geodesic* distance metric, i.e. $c(\mathbf{x}(t)) = \arg\min_{i=1}^{K} \|\mathrm{Log}_{\mathbf{x}(t)}(\mathbf{w}_i(t))\|$.

Given data on any Riemannian manifold M, if the classical update equation in (2) is applied, the resulting codebook vector can **not** be guaranteed to still reside on M, and thus fails to *accurately* learn the *intrinsic* structure properties of the underlying manifold. Fortunately, our proposed scheme elegantly circumvents this problem.

In fact, the update equation (2) in the classical SOM is a special case of (14) in the generalized SOM. When the Riemannian manifold M approaches to a linear manifold, we have

$$\mathrm{Log}_{\mathbf{w}_i(t)}(\mathbf{x}(t)) = \mathbf{x}(t) - \mathbf{w}_i(t). \tag{15}$$

and

$$\mathrm{Exp}_{\mathbf{w}_i(t)}(\triangle) = \mathbf{w}_i(t) + \triangle \, . \tag{16}$$

Thus

$$\begin{aligned} &\text{the right-side of (14) becomes} \\ &= \mathrm{Exp}_{\mathbf{w}_i(t)}(\triangle) = \mathbf{w}_i(t) + \triangle \\ &= \mathbf{w}_i(t) + \alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot \mathrm{Log}_{\mathbf{w}_i(t)}(\mathbf{x}(t)) \\ &= \mathbf{w}_i(t) + \alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot (\mathbf{x}(t) - \mathbf{w}_i(t)) \end{aligned} . \tag{17}$$

Substituting (17) into (14), we have

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot (\mathbf{x}(t) - \mathbf{w}_i(t)). \tag{18}$$

Equation (18) is identical to Equation (2). In other words, the classical SOM is a special case of the generalized SOM.

4.2 Batch Learning Algorithm

Likewise, for data that reside on a Riemannian manifold M, points corresponding to the updated codebook vectors are not guaranteed to still reside on M when applying formula (4) and (5). In fact, batch learning algorithm for a SOM on a Riemannian manifold should first compute the *intrinsic* average $\overline{\mathbf{x}}_j$ of samples in each Voronoi region V_j, and then update the codebook vector to a *weighted intrinsic average* of $\overline{\mathbf{x}}_j, 1 \le j \le K$. More specifically, Equation (4) and (5) should be replaced respectively by

$$\overline{\mathbf{x}}_j = \mathbf{IntrinsicAvg}(\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_{n_j}). \tag{19}$$

$$\mathbf{w}_i(t+1) = \mathbf{WIntrinsicAvg}(w_{1i}, \overline{\mathbf{x}}_1; \cdots; w_{Ki}, \overline{\mathbf{x}}_K). \tag{20}$$

where w_{ji} is the weight value of $\overline{\mathbf{x}}_j$, defined as

$$w_{ji} = \frac{h_{ji}(t) \cdot n_j}{\sum_{j=1}^{K} h_{ji}(t) \cdot n_j}. \tag{21}$$

The right-sides of (19) and (20) can be solved by applying (9) and (11), respectively.

4.3 Initialization

Another key issue in our proposed learning algorithms is the initialization of the SOM. An appropriate initialization scheme must guarantee that the points corresponding to the initialized codebook vectors lie on the Riemannian manifold M, from where data are sampled.

We here present a simple randomized initialization scheme. When initializing the codebook vector $\mathbf{w}_i$, we first randomly select two samples, denoted by $\mathbf{x}$ and

$\mathbf{y}$, from dataset $\{\mathbf{x}_i\}_{i=1}^N$. Then, $\mathbf{w}_i$ is initialized by taking the mid point between $\mathbf{x}$ and $\mathbf{y}$ as follows

$$\mathbf{w}_i = \mathrm{Exp}_{\mathbf{x}}\left(\frac{1}{2} \cdot \mathrm{Log}_{\mathbf{x}}(\mathbf{y})\right). \tag{22}$$

We repeat the above procedure K times until all the codebook vectors of the SOM have been initialized.

5 Experimental Results

In this section, we compare the performance between the classic SOM and the generalized SOM by performing experiments on both synthetic data and real word 3D facial shape data.

5.1 Experiments on Synthetic Data

We use standard Swiss-roll datasets in this section. We randomly select 2000 points in a Swiss-roll dataset. These points are used as training samples. After training, the distribution of the codebooks of the SOM are plotted.

Results are shown in Fig.4. The left plot in Fig.4 is the distribution of 2000 sample points, the center and right plots are the learning results of the classical SOM and the generalized SOM, respectively. More specifically, for example, the right plot in Fig.4 is obtained by directly plotting the codebook vectors of the trained generalized SOM.

The center plot in Fig.4 clearly demonstrates that the updated codebook vectors in the classical SOM can not be guaranteed to still reside on the underlying

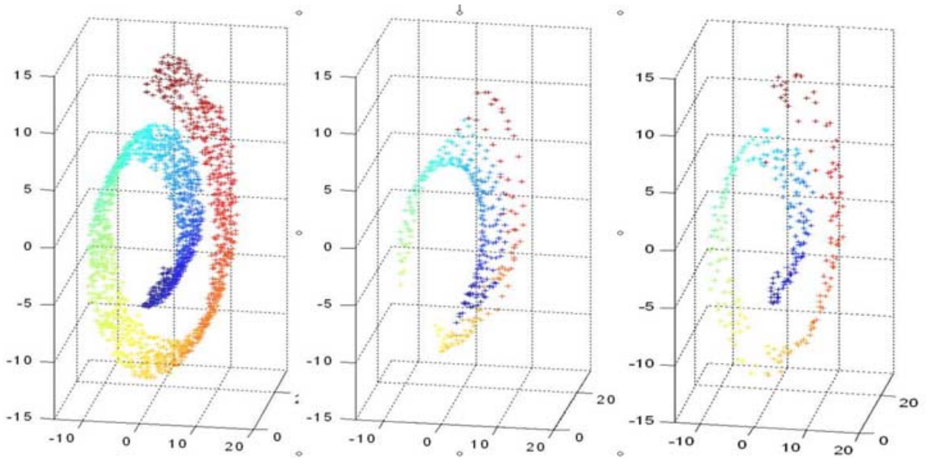


Fig. 4. Experimental results on Swiss-roll data. Left: samples of Swiss-roll data. Center: learning result of the classical SOM. Right: learning result of the generalized SOM.

manifold, as claimed in previous section. Thus, the resulting codebook vectors can not accurately reflect the structure of the underlying manifold. However, since the generalized SOM moves the codebook vectors along geodesics, they can be guaranteed still to reside on the underlying manifold, as depicted in the right plot in Fig.4.

From Fig.4, we can easily conclude that compared with the classical SOM, the generalized SOM can more accurately learn the topological neighborhood structure of the underlying manifold.

5.2 Experiments on Real Data

Our second application involves a facial needle-map (a set of facial surface normals) as the representation of a 3D facial surface shape.

To explore the effectiveness of the generalized SOM, both facial needle-maps extracted from range images, referred as Dataset-I, and the facial needle-maps recovered from 2D brightness images using shape-from-shading, referred as Dataset-II, are used in our experiments. Dataset-I contains range images obtained from the UND database [17], and Dataset-II is a set of facial needle-maps recovered from FERET database by applying the non-Lambertian shape-from-shading method, which was proposed by Smith&Hancock [18].

Dataset-I: Facial Needle-maps from UND. Dataset-I is obtained from the biometrics database from University of Notre Dame [17]. UND biometrics database provides both 3D range images and 2D face images for each subject. We select 200 range images for 200 subjects(100 females and 100 males, each subject has only one range image). Each selected 3D range image is geometrically aligned and get a corresponding facial surface height matrix $\mathbf{H}_{114\times100}$. Entry value $\mathbf{H}(i,j)$ is the facial surface height value corresponding to facial image location (i,j). The needle-map $\mathbf{N}_{114\times100\times3}$ then can be directly computed from the facial surface height matrix $\mathbf{H}_{114\times100}$ of the aligned 3D range image. More details about constructing Dataset-I can be found in Wu&Hancock [19]. Fig.5 presents 4 needle-maps extracted from Dataset-I.

Dataset-II: Recovered Facial Needle-maps from FERET. FERET [20] - the outcome of FERET program sponsored by DARPA - has become a standard face image database in the face recognition literatures. Each subject has different facial images with a variety of pose, angle of view, illumination, expression and age. Different gender and ethnicity categories are also covered by the database. In our experiments, 200 frontal facial images for 200 subjects are selected from FERET (103 females and 97 males, each subject has only one face image). Each selected image is pre-processed to a resolution of 142×124 by cropping, rotating, scaling and aligning. Face images are aligned with each other based on the central points of the left and right eyes.

After pre-processing, we use non-Lambertian shape-from-shading to recover the needle-map $\mathbf{N}_{142\times124\times3}$ for each facical image [18]. Fig.6 presents 4 example recovered needle-maps from Dataset-II.

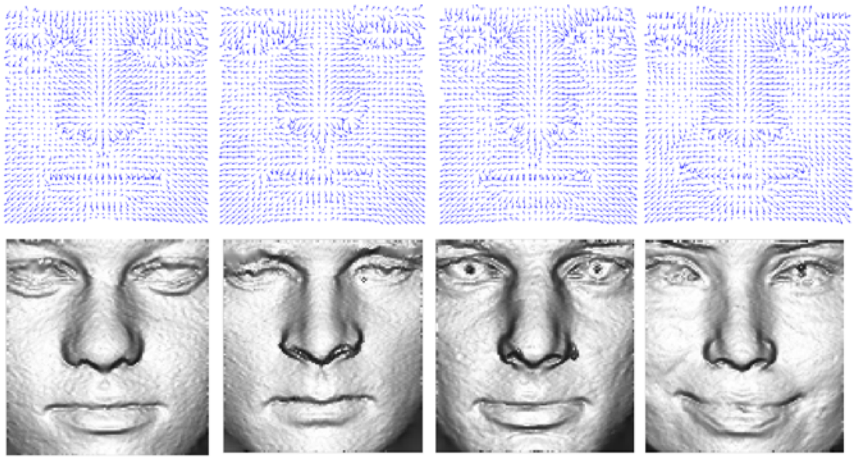


Fig. 5. Four needle-maps in Dataset-I. The first row shows the needle-maps. The second row contains images obtained by visualizing the 3rd component of each needle-map as a 2D image.

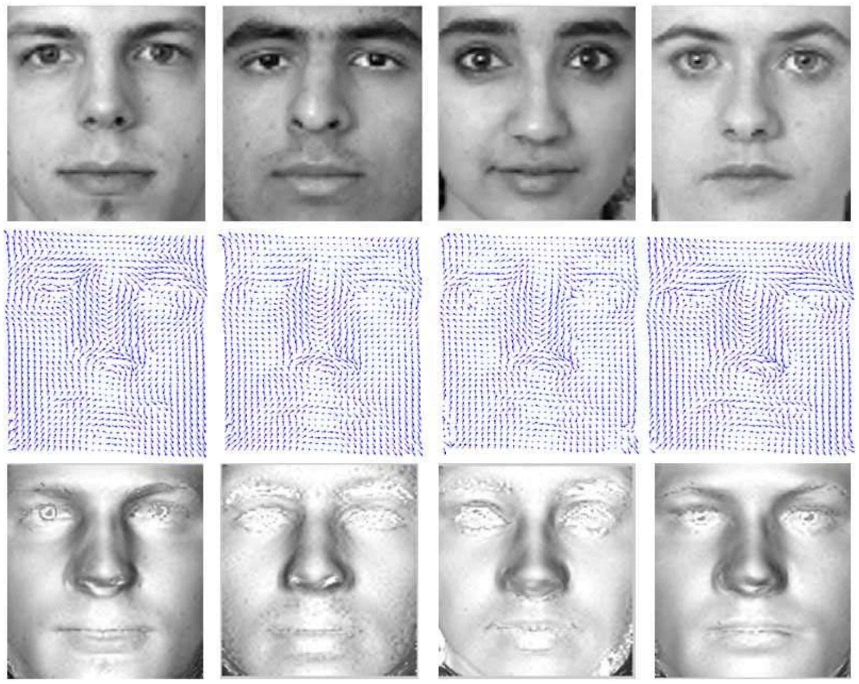


Fig. 6. Four recovered needle-maps from Dataset-II. The first row contains the 4 original 2D facial images from FERET. The second row contains the 4 recovered facial needle-maps corresponding to the 4 2D facial images in the first row. The third row contains 4 images obtained by visualizing the 3rd component of each needle-map as a 2D image.

Localizing the facial needle-map. Here, we represent 2D facial image by using local feature vectors ($LFVs$) [21] to lower the input dimensionality of the SOM and lessen the problems associated with the high dimensionality of facial surface needle-maps.

Let $\mathbf{N}_{h \times w \times 3}$ denote a facial needle-map matrix, where h and w are the height and the width of facial surface, respectively. The facial needle-map can be represented by several small local facial needle-map matrices ($LFNMs$). The original facial needle-map is partitioned into L non-overlapping sub-blocks with equal size of $h^{'} \times w^{'} \times 3$, where $h^{'}$ and $w^{'}$ are the height and the width of each sub-block, respectively, and $L = \lfloor \frac{h}{h'} \rfloor \times \lfloor \frac{w}{w'} \rfloor$. As an example, a facial surface needle-map $\mathbf{N}_{142 \times 124 \times 3}$ in Dataset-II can be represented by $L = \lfloor \frac{142}{4} \rfloor \times \lfloor \frac{124}{4} \rfloor = 1085$ $LFNMs$, each with a size of $4 \times 4 \times 3$.

We use the $LFNMs$, rather than the original facial needle-maps, as training patterns for the SOM. Notice that each $LFNM$ with size of $h^{'} \times w^{'} \times 3$ is actually a point lying on Riemannian manifold $S^2(h^{'} \times w^{'})$.

Classification. Let $\{T_i\}_{i=1}^{N}$ be the training facial needle-maps, L be the number of $LFNMs$ for each facial needle-map. Thus there are in total $N \times K$ $LFNMs$ used as training patterns.

After training the SOM, we can obtain the BMU matrix, denoted **BMU_Train**, by feeding N training facial needle-maps to the trained SOM:

$$\mathbf{BMU\_Train} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1L} \\ b_{21} & b_{22} & \cdots & b_{2L} \\ \cdots & \cdots & b_{ij} & \cdots \\ b_{N1} & b_{N2} & \cdots & b_{NL} \end{pmatrix}. \tag{23}$$

where b_{ij} is the BMU which is activated by feeding the j-th $LFNM$ of the i-th training facial needle-map to the trained SOM.

Given a test facial needle-map, we first partition it into L $LFNMs$, then we feed each of the L $LFNMs$ to the trained SOM in turn, and thus obtain a BMU vector, denoted $\mathbf{BMU\_Test} = (t_1, t_2, \cdots, t_L)$, where t_i is the BMU which is activated by feeding the i-th $LFNM$ of the testing needle-map to the trained SOM.

From **BMU_Test** and **BMU_Train**, we can obtain a confidence matrix using soft-kNN technique [21]

$$\mathbf{D}(\mathbf{BMU\_Test}, \mathbf{BMU\_Train}) = (c_{ij}). \tag{24}$$

where c_{ij} $(1 \le i \le N,\ 1 \le j \le L)$ is the confidence value which measures the similarity between the j-th $LFNM$ of the test facial needle-map and the corresponding $LFNM$ of the i-th training facial needle-map.

We can determine the gender by utilizing the confidence values $c_{ij}, 1 \le i \le N, 1 \le j \le L$. Let M and F be the sets of male and female training facial needle-maps, respectively. Obviously, $M \cup F = \{T_i\}_{i=1}^{N}$ and $M \cap F = \emptyset$. The *s*imilarity *c*onfidence value between the testing facial needle-map and the i-th training facial needle-map, denoted sc_i , can be obtained by aggregating the corresponding L confidence values, i.e. $sc_i = \sum_{j=1}^{L} c_{ij}$. Here, we present two different schemes as follows.

1) **Most Similar Subject:** This scheme makes a decision with the maximum rule after the similarity confidence values between the test facial needle-map and the entire set of training needle-maps are computed. We can locate the training facial needle-map, denoted T_{i^*}, which corresponds to the maximum similarity confidence value with respect to the test facial needle-map.

$$T_{i^*} = \arg \max_{T_i \in M \cup F} sc_i = \arg \max_{T_i \in M \cup F} \sum_{j=1}^{L} c_{ij}. \tag{25}$$

We then classify the gender of the test facial needle-map as the gender of T_{i^*}.

2) **Most Similar Class:** Let M_{avg} be the average similarity confidence value between the test facial needle-map and the entire set of male training facial needle-maps. F_{avg} be the average similarity confidence value between the test facial needle-map and the entire set of female training facial needle-maps.

$$M_{avg} = \frac{1}{\|\{T_i\}\|} \sum_{T_i \in M} \sum_{j=1}^{L} c_{ij}. \tag{26}$$

$$F_{avg} = \frac{1}{\|\{T_i\}\|} \sum_{T_i \in F} \sum_{j=1}^{L} c_{ij}. \tag{27}$$

If $M_{avg} > F_{avg}$, we classify the gender of the test facial needle-map as male.

Results and Analysis. In the following experiments, the size of SOM is set to be 30×16 and the output neurons of the SOM are arranged in 2D hexagonal lattice structure. Batch learning algorithms for both the classical SOM and the generalized SOM are implemented. Note that 10-fold cross validation is applied.

Table 1. Number of erroneously classified needle-maps and corresponding accuracy rate on Dataset-I

sub-block size	Method	
	classical SOM	generalized SOM
$4 \times 4 \times 3$	31 (84.5%)	25 (87.5%)
$8 \times 8 \times 3$	36 (82.0%)	27 (86.5%)
$16 \times 16 \times 3$	50 (75.0%)	39 (80.5%)
$32 \times 32 \times 3$	53 (73.5%)	48 (76.0%)

Table 2. Number of erroneously classified needle-maps and corresponding accuracy rate on Dataset-II

sub-block size	Method	
	classical SOM	generalized SOM
$4 \times 4 \times 3$	34 (83.0%)	21 (89.5%)
$8 \times 8 \times 3$	37 (81.5%)	26 (87.0%)
$16 \times 16 \times 3$	46 (77.0%)	32 (84.0%)
$32 \times 32 \times 3$	55 (72.5%)	36 (82.0%)

Table 1 shows the experimental results for both the classical SOM and the generalized SOM on Dataset-I. Table 2 presents the experimental results on Dataset-II.

From Table 1 and Table 2, it is clear that both classical SOM and the generalized SOM achieve the best accuracy rates when the size of sub-block of is $4 \times 4 \times 3$. The best accuracy rate for the generalized SOM on Dataset-I is 87.5%, which is 3% higher than that (84.5%) obtained with the classical SOM. The best accuracy rate for the generalized SOM on Dataset-II is 89.5%, which is 6.5% higher than that (83.0%) obtained using the classical SOM.

We also carried out experiments by gradually changing the size of sub-block from $4 \times 4 \times 3$ to $32 \times 32 \times 3$. From Table 1 and Table 2, the accuracy rates for both the classical SOM and the generalized SOM decrease when the size of sub-block becomes large. However, the generalized SOM consistently outperforms the classical SOM under different sub-block sizes on both datasets.

6 Conclusion

In this paper, we generalize the classical SOM from a flat Euclidean space to a curved Riemannian manifold. Both sequential and batch learning algorithms for learning a SOM on a Riemannian manifold are presented. We prove that the classic SOM learning algorithms are a special cases of the generalized learning algorithms of the generalized SOM. The generalized SOM can learn the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold on which the data resides. We compare the generalized SOM with the classical SOM on both synthetic data and real world facial surface normal data. Experimental results show that the generalized SOM outperforms the classical SOM when the data lies on a curved manifold.

The essential idea underlying the generalized SOM is to move the codebook vector closer to the learning sample along a geodesic. Thus, as long as we can *approximately* compute the geodesic, the generalized SOM can be applied. Many studies concerning fast algorithms for approximating geodesics [22,23,24] have been conducted. These algorithms can be incorporated into the generalized SOM.

References

1. Kohonen, T.: Self-Organization and Associative Memory, 2nd edn. Springer, Berlin (1988)
2. Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., Saarela, A.: Self Organization of a Massive Document Collection. IEEE Transactions on Neural Networks 11(3), 574–585 (2000)
3. Mavroudi, S., Papadimitriou, S., Bezerianos, A.: Gene Expression Data Analysis with a Dynamically Extended Self-organized Map That Exploits Class Information. Bioinformatics 18(11), 1446–1453 (2000)
4. Joutsiniemi, S.L., Kaski, S., Larsen, T.A.: Self-organizing Map in Recognition of Topographic Patterns of EEG Spectra. Bioinformatics 42(11), 1062–1068 (1995)
5. Laaksonen, J., Koskela, M., Oja, E.: Class Distributions on SOM Surfaces for Feature Extraction and Object Retrieval. Neural Networks 17(8), 1121–1133 (2004)

6. Barreto, G.A., Arajo, A.F.R.: Identification and Control of Dynamical Using the Self-organizing Map. IEEE Transactions on Neural Networks 15(5), 1244–1259 (2004)
7. Wu, S.T., Chow, T.W.S.: PRSOM: A New Visualization Method by Hybridizing Multidimensional Scaling and Self-organizing Map. IEEE Transactions on Neural Networks 16(6), 1362–1380 (2005)
8. Andras, P.: Kernel-Kohonen Networks. Int. J. Neural. Syst. 12(2), 117–135 (2002)
9. Manuel, A.M., Alberto, M.: Extending the SOM algorithm to Non-Euclidean Distances via the Kernel Trick. In: 11th International Conference on Neural-Information-Processing, pp. 150–157. Springer Press, Calcutta (2004)
10. Ritter, H.: Self-organiring Maps on non-Euclidean Spaces. In: WSOM 1999, pp. 1321–1344. IEEE Press, Espoo (1999)
11. Shi, C.Q., Zhang, S.L., Shi, Z.Z.: Geodesic Distance based SOM for Image Clustering. In: International Conference on Sensing, Computing and Automation, pp. 2483–2488. Watam Press, Chongqing (2006)
12. Simila, T.: Self-organizing Map Learning Nonlinearly Embedded Manifolds. J. Information Visualization 4, 22–31 (2005)
13. Pennec, X.: Probabilities and Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. In: IEEE Workshop on Nonlinear Signal and Image Processing, pp. 194–198. IEEE Press, Antalya (1999)
14. Karcher, H.: Riemannian Center of Mass and Mollifier Smoothing. Commun. Pure Appl. Math. 30(5), 509–541 (1977)
15. Jost, J.: Riemannian Geometry and Geometric Analysis, 3rd edn. Springer, Berlin (2002)
16. Fletcher, P.T., Lu, C., Pizer, S.M., Joshi, S.: Principal Geodesic Analysis for The Study of Nonlinear Statistics of Shape. IEEE Transactions on Medical Imaging 23(8), 995–1005 (2004)
17. Chang, K., Bowyer, K.W., Flynn, P.J.: Face Recognition using 2D and 3D Facial Data. In: ACM Workshop on Multimodal User Authentication, pp. 25–32. ACM Press, Santa Barbara (2003)
18. Smith, W.A.P., Hancock, E.R.: A New Framework for Grayscale and Colour Non-lambertian Shape-from-Shading. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 869–880. Springer, Heidelberg (2007)
19. Wu, J., Smith, W.A.P., Hancock, E.R.: Gender Classification using Shape from Shading. In: BMVC, UK, pp. 499–508 (2007)
20. Phillips, P.J., Moon, H., Rauss, P.J., Rizvi, S.: The FERET Evaluation Methodology for Face Recognition Algorithms. Transactions on Pattern Analysis and Machine Intelligence 22(10), 1090–1104 (2000)
21. Tan, X.Y., Chen, S.C., Zhou, Z.H., Zhang, F.Y.: Recognizing Partially Occluded, Expression Variant Faces from Single Training Image per Person with SOM and Soft kNN Ensemble. IEEE Transactions on Neural Networks 16(4), 875–886 (2005)
22. Ruggeri, M.R., Darom, T., Saupe, D., Kiryati, N.: Approximating Geodesics on Point Set Surface. In: EG/IEEE Symposium on Point-Based Graphics, pp. 85–93. IEEE Press, Los Alamitos (2006)
23. Stawiaski, J., Decencifre, E., Bidault, F.: Computing Approximate Geodesics and Minimal Surfaces using Watershed and Graph-cuts. In: 8th International Symposium on Mathematical Morphology, Brazil, pp. 349–360 (2007)
24. Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J., Hoppe, H.: Fast Exact and Approximate Geodesics on Meshes. ACM Transactions on Graphics 24(3), 553–560 (2005)

Surface Quasi-Conformal Mapping by Solving Beltrami Equations

W. Zeng[1], F. Luo[2], S.-T. Yau[3], and X.D. Gu[1]

[1] Computer Science Department, State University of New York at Stony Brook, USA
[2] Mathematics Department, Rutgers University, USA
[3] Mathematics Department, Harvard University, USA

Abstract. We consider the problem of constructing quasi-conformal mappings between surfaces by solving Beltrami equations. This is of great importance for shape registration.

In the physical world, most surface deformations can be rigorously modeled as quasi-conformal maps. The local deformation is characterized by a complex-value function, *Beltrami coefficient*, which describes the deviation from conformality of the deformation at each point.

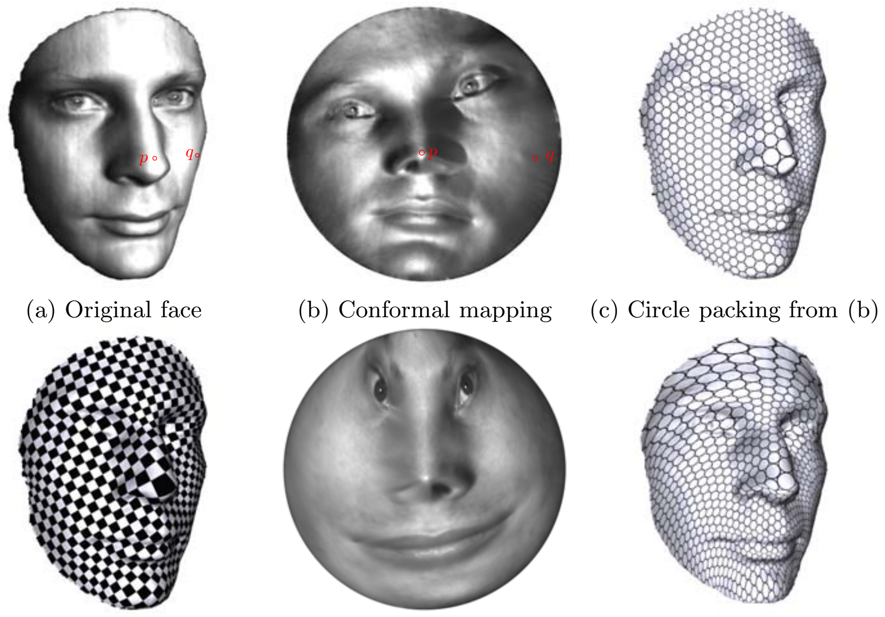
We propose an effective algorithm to solve the quasi-conformal map from the Beltrami coefficient. The major strategy is to deform the conformal structure of the original surface to a new conformal structure by the Beltrami coefficient, such that the quasi-conformal map becomes a conformal map. By using holomorphic differential forms, conformal maps under the new conformal structure are calculated, which are the desired quasi-conformal maps.

The efficiency and efficacy of the algorithms are demonstrated by experimental results. Furthermore, the algorithms are robust for surfaces scanned from real life, and general for surfaces with different topologies.

Keywords: Quasic-Conformal Map, Beltrami Equation, Riemannian Metric, Uniformization.

1 Introduction

Computing the mappings between surfaces is of fundamental importance in many fields in science and engineering. Mappings will introduce distortions on surfaces, which can be measured by area distortion and angle distortion. Mappings without area or angle distortions are isometric, those without angle distortions are conformal. Isometric and conformal mappings are extremely rare in reality. Most mappings in the physical world have bounded angle distortion, which can be categorized as quasi-conformal mappings. Conformal mappings are fully determined by boundary conditions. Quasi-conformal mappings are determined by both boundary conditions and a function, the so-called Beltrami coefficient, defined on the source surface, therefore it gives point-wise control to the users. The detailed control of the mapping is crucial for many practical applications. This work focuses on how to construct quasi-conformal mappings from the given



(a) Original face (b) Conformal mapping (c) Circle packing from (b)

(d) Checker-board texture (e) Quasi-conformal mapping (f) Circle packing from (e)

Fig. 1. Conformal and Quasi-Conformal mappings for a topological disk

Beltrami coefficients. Although the discussions are mainly on genus zero surfaces with an arbitrary number of boundaries, the method can be directly applied to surfaces with general topologies.

Suppose two surfaces S_1, S_2 have Riemannian metrics $\mathbf{g}_1$ and $\mathbf{g}_2$. A homeomorphism ϕ maps S_1 to S_2. We say ϕ is conformal, if it is angle-preserving. Mathematically, the pull back metric $\phi^*\mathbf{g}_2 = e^{2u}\mathbf{g}_1$. Locally, conformal mapping is just scaling, therefore the local shapes are well preserved. Figure 1 shows a conformal map from a human face surface (a) to the planar disk (b). From the planar image (b), it is obvious that the major facial features are well preserved. If we put a checker-board on the disk, and pull back the texture onto the face surface, all the right-angled corners of checkers are preserved (d). Geometrically, a conformal mapping maps infinitesimal circles to infinitesimal circles. As shown in the figure, the regular circle packing on the texture is pulled back to the face, and the shape of the circles is well preserved as shown in (c).

In general, conformal mappings are rare. Most mappings in physical world are quasi-conformal. Conformal mappings have no angle distortions, while quasi-conformal mappings introduce bounded angle distortion. Geometrically, a quasi-conformal map transforms infinitesimal circles on the source surface to infinitesimal ellipses on the target surface with bounded eccentricity. Figure 1(e) shows a quasi-conformal map from the face surface to the unit disk. Frames (c) and (f) show that the circles on the texture are mapped to the ellipses.

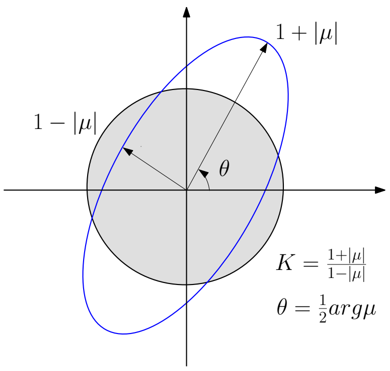


Fig. 2. Illustration of how Beltrami coefficient μ measures the distortion by a quasi-conformal mapping that maps a small circle to an ellipse with dilatation K

Quasi-conformal maps are controlled by both boundary condition and the so-called Beltrami coefficient μ. The characteristics of the infinitesimal ellipses, the orientation of the axis and the ratio between the longer axis and shorter axis are encoded in μ, but the scale factor is absent. In detail, let $\phi : S_1 \to S_2$ be the map, z and w local conformal parameters of $(S_1, \mathbf{g}_1)$ and $(S_2, \mathbf{g}_2)$, such that $\mathbf{g}_1 = e^{2u_1}dzd\bar{z}$, $\mathbf{g}_2 = e^{2u_2}dwd\bar{w}$, then ϕ has a local representation. The Beltrami coefficient is defined as

$$\frac{\partial \phi}{\partial \bar{z}} = \mu(z)\frac{\partial \phi}{\partial z}. \tag{1}$$

The ratio between the two axis of the ellipse is given by $K = \frac{1+|\mu(z)|}{1-|\mu(z)|}$ and the orientation of the axis is related to $arg\mu(z)$. As shown in Figure 2, two orthogonal lines associated with the circle are the principal distortion directions and the angle is measured between corresponding principal distortion directions.

The fundamental problem is to find a quasi-conformal map ϕ, which satisfies the given Beltrami coefficient μ, namely, solving Beltrami equations (see Eqn. 1). The major strategy is as follows. First, we compute conformal maps, $\phi_1 : S_1 \to \mathbb{D}_1$, $\phi_2 : S_2 \to \mathbb{D}_2$, where $\mathbb{D}_1$ and $\mathbb{D}_2$ are domains on the complex plane, with the canonical Euclidean metric $\mathbf{g}_0 = dzd\bar{z}$. Then we construct a quasi-conformal map $\tau : (\mathbb{D}_1, \mathbf{g}_0) \to (\mathbb{D}_2, \mathbf{g}_0)$, such that the Beltrami coefficient of τ equals to μ. Then the pullback metric on $\mathbb{D}_1$ induced by τ is

$$\tau^*(\mathbf{g}_0) = |\frac{\partial \tau}{\partial z}|^2 |dz + \mu d\bar{z}|^2,$$

which is conformal to the metric $\mathbf{g}_1 = |dz + \mu d\bar{z}|^2$, and then $\tau : (\mathbb{D}_1, \mathbf{g}_1) \to (\mathbb{D}_2, \mathbf{g}_0)$ is a conformal map. By changing the metric on $\mathbb{D}_1$ from $\mathbf{g}_0$ to $\mathbf{g}_1$, the quasi-conformal map τ becomes conformal and can be computed by using mature methods for conformal mappings, such as using Gu-Yau's method based on holomorphic differential forms. Finally, the desired quasi-conformal mapping satisfying the Beltrami equation (see Eqn. 1) is given by $\phi = \phi_2^{-1} \circ \tau \circ \phi_1$.

Figure 1 (e) and (f) show a quasi-conformal map, whose Beltrami coefficient is given by $\mu(z) = z$, where z is the conformal parameter as shown in (b). Near the nose tip, $\mu(z)$ is close to zero, therefore, the mapping there is close to be conformal, and the ellipses are rounder as illustrated in (f).

The paper is organized in the following way: Section 2 briefly review the most related works in the field; Section 3 introduces the theoretic background; Section 4 focuses on the computational methodologies; Section 5 reports the experimental results; the paper is concluded in Section 6.

2 Previous Work

Recently, with the development of digital scanning technology, computing conformal mappings between surfaces becomes more and more important. In computer graphics and discrete mathematics, much sound research has focused on discrete conformal mappings.

The computational method of current work is mainly based on harmonic maps and holomorphic differential forms. Here, we briefly overview most related work, and refer readers to [1,2] for thorough surveys. For example, curvature flow is another important method for computing conformal mappings. In current work, we skip the curvature flow methods, such as discrete Ricci flow [3,4].

Discrete harmonic maps were constructed in [5], where the cotan formula was introduced. First order finite element approximations of the Cauchy-Riemann equations were introduced by Levy et al. [6]. Discrete intrinsic parameterization by minimizing Dirichlet energy was introduced by [7]. Mean value coordinates were introduced in [8] to compute generalized harmonic maps; Discrete spherical conformal mappings were used in [9] and [10].

Discrete holomorphic forms were introduced by Gu and Yau [11] to compute global conformal surface parameterizations for high genus surfaces. Another approach of discrete holomorphy was introduced in [12] using discrete exterior calculus [13]. The problem of computing optimal holomorphic 1-forms to reduce area distortion was considered in [14]. Gortler et al. [15] generalized 1-forms to the discrete case, using them to parameterize genus one meshes. Tong et al. [16] generalized the 1-form method to incorporate cone singularities. Discrete one-forms have been applied for meshing point clouds in [17], surface tiling [18], surface quadrangulation [19]. The holomorphic 1-form method has been applied in virtual colonoscopy [20]. The colon surface is reconstructed from MRI images, and conformally mapped to the planar rectangle. This improves the efficiency and accuracy for detecting polyps. Conformal mapping is used for brain cortex surface morphology study in [10]. By mapping brain surfaces to spheres, cortex surface registration and comparison become straightforward. Holomorphic 1-form method has also been applied in computer vision [21,22] for 3D shape matching, recognition and stitching. In geometric modeling field, constructing splines on general surfaces is one of the most fundamental problems. It is proven in [23] that if the surface has an affine structure, then splines can be generalized to it directly. Holomorphic 1-forms can be applied for computing the affine structures of general surfaces.

3 Theoretical Background

In this section, we briefly introduce the major concepts in differential geometry and Riemann surface theory, which are necessary to explain the quasi-conformal maps. We refer readers to [24,25] for detailed information.

3.1 Conformal Structure and Riemann Surface

A Riemann surface is a surface with a complex structure, such that complex analysis can be defined on the surface.

Suppose $f : \mathbb{C} \to \mathbb{C}$ is a complex function. $f(x,y) = (u(x,y), v(x,y))$, f is *holomorphic*, if it satisfies the following *Cauchy-Riemann equations*,

$$\begin{cases} \frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \end{cases}$$

If a holomorphic function f is a bijection, and the inverse f^{-1} is also holomorphic, then f is *biholomorphic*.

As shown in Fig.3, suppose S is a surface covered by a collection of open sets $\{U_\alpha\}$, $S \subset \bigcup_\alpha U_\alpha$. A chart is (U_α, ϕ_α), where $\phi_\alpha : U_\alpha \to \mathbb{C}$ is a homeomorphism. The chart transition function $\phi_{\alpha\beta} : \phi_\alpha(U_\alpha \cap U_\beta) \to \phi_\beta(U_\alpha \cap U_\beta)$, $\phi_{\alpha\beta} = \phi_\beta \circ \phi_\alpha^{-1}$. The collection of the charts $\mathcal{A} = \{(U_\alpha, \phi_\alpha)\}$ is called the *atlas* of S. If all chart transition functions are biholomorphic, then the atlas is called a *conformal atlas* of the surface. Two conformal atlases are compatible, if their union is still a conformal atlas. The union of all compatible conformal atlases is called the *conformal structure* of the surface. A surface with a conformal structure is called a *Riemann surface*.

Suppose S has a Riemannian metric $\mathbf{g}$, the local coordinates z_α is called *isothermal*, if the metric has local representation

$$\mathbf{g} = e^{2\lambda(z)} dz d\bar{z}.$$

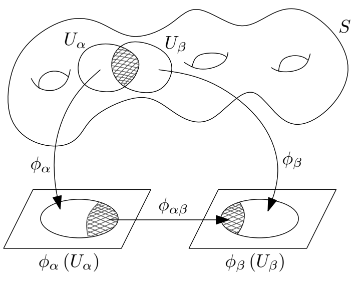


Fig. 3. A surface is covered by an atlas. If all chart transitions are holomorphic, the atlas is a conformal atlas. If all local coordinates are isothermal, the surface is a Riemann surface.

A conformal structure is compatible with the Riemannian metric, if all its local coordinates are isothermal. In practice, the surfaces of interest are embedded in $\mathbb{R}^3$, therefore with the induced Euclidean metric. The conformal structure compatible to the induced metric is our major focus.

Suppose ω is a complex differential form, such that on each chart (U_α, ϕ_α) with local complex parameter z_α, $\omega = f_\alpha(z_\alpha)dz_\alpha$. Suppose two charts overlap $U_\alpha \cap U_\beta$, then

$$f_\beta(z_\beta) = f_\alpha(z_\alpha(z_\beta))\frac{dz_\alpha}{dz_\beta}.$$

If f_α is conformal for arbitrary local coordinates, then ω is called a *holomorhic 1-form.* All holomorphic 1-forms form a group, which is isomorphic to the first cohomology group of the surface. The holomorphic 1-form plays a crucial role in computing conformal mappings.

3.2 Quasi-Conformal Mapping

Suppose $(S_1, \mathcal{A}_1)$ and $(S_2, \mathcal{A}_2)$ are two Riemann surfaces and $\mathcal{A}_i$'s are their conformal structures. Suppose (U_α, ϕ_α) is a local chart of $\mathcal{A}_1$, and (V_β, ψ_β), is a local chart of $\mathcal{A}_2$. $\phi : S_1 \to S_2$ is a *conformal map* if and only if

$$\psi_\beta \circ \phi \circ \phi_\alpha^{-1} : \phi_\alpha(U_\alpha) \to \psi_\beta(V_\beta)$$

is biholomorphic. A conformal map preserves angles.

A generalization of the conformal map is called the *quasi-conformal* map which is an orientation-preserving homeomorphism between Riemann surfaces with bounded conformality distortion, in the sense that the first order approximation of the quasi-conformal homeomorphism takes small circles to small ellipses of bounded eccentricity. Thus, a conformal homeomorphism that maps a small circle to a small circle can also be regarded as quasi-conformal.

Mathematically, ϕ is quasi-conformal provided that it satisfies the Beltrami equation in Eqn. 1 on a local chart for some complex valued Lebesgue measurable μ satisfying $|\mu|_\infty < 1$. μ is called the *Beltrami coefficient*, which is a measure of conformality. In particular, the map ϕ is conformal around a small neighborhood of p when $\mu(p) = 0$. In general, ϕ maps an infinitesimal circle to a infinitesimal ellipse. From $\mu(p)$, we can determine the angles of the directions of maximal magnification as well as the amount of maximal magnification and maximal shrinking. Specifically, the angle of maximal magnification is $arg\mu(p)/2$ with magnifying factor $1 + |\mu(p)|$; the angle of maximal shrinking is the orthogonal angle $(arg\mu(p) - \pi)/2$ with shrinking factor $1 - |\mu(p)|$. The distortion or dilation is given by:

$$K = \frac{1 + |\mu(p)|}{1 - |\mu(p)|} \tag{2}$$

Thus, the Beltrami coefficient μ gives us all the information about the conformality of the map (See Fig. 2).

In terms of the metric tensor, considering the effect of the pullback under ϕ of the canonical Euclidean metric $\mathbf{g}_0$, the resulting metric is given by:

$$\phi^*(\mathbf{g}_0) = |\frac{\partial\phi}{\partial z}|^2|dz + \mu(z)d\overline{z})|^2 \tag{3}$$

4 Computational Algorithm

This section introduces the computational algorithms for computing conformal mappings for surfaces based on holomorphic 1-form method.

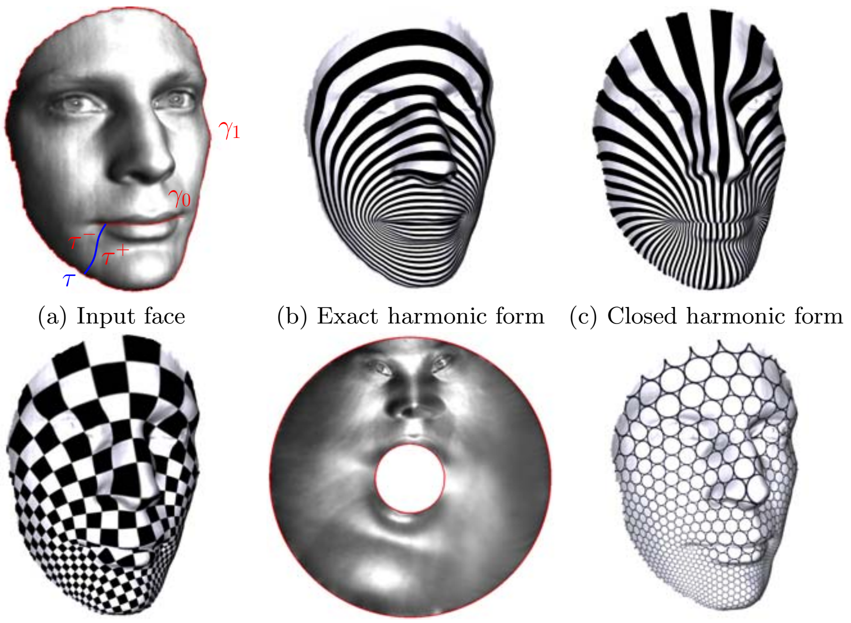
4.1 Conformal Mapping

Doubly Connected Domain. Figure 4 shows a human face surface, with the mouth sliced open, therefore it is a doubly connected domain (a topological annulus). We denote the surface as S, its outer boundary as γ_1 and the inner boundary as γ_0, namely $\partial S = \gamma_1 - \gamma_0$. The conformal mapping $\phi : S \to \mathbb{C}$ is constructed as follows:

1. Compute a holomorphic 1-form ω, such that

$$Im(\int_{\gamma_1} \omega) = 2\pi,$$

   where $Im()$ denotes the imaginary part.



(a) Input face (b) Exact harmonic form (c) Closed harmonic form

(d) Holomorphic form (e) Conformal image (f) Circle packing from (e)

Fig. 4. Conformal mapping for doubly connected domain

2. Choose a base point $p \in S$, $Im(\phi(p)) = 0$. For any point $q \in S$, find an arbitrary path γ connecting p and q, then

$$\phi(q) = exp(\int_\gamma \omega).$$

We applied Gu-Yau's algorithm for computing the holomorphic 1-form, which is similar to that introduced in [11].

1. Compute a harmonic function $f : S \to \mathbb{R}$, such that

$$\begin{cases} \Delta f &= 0 \\ f|_{\gamma_0} &= 0 \\ f|_{\gamma_1} &= 1 \end{cases}$$

Let $\omega_1 = df$. Figure 4 (b) shows the exact harmonic 1-form ω_1.
2. Find a path τ connecting the two boundaries as shown in Fig. 4 (a). Slice the surface along the path to get a topological quadrilateral $\tilde{S}$, with boundaries $\gamma_0, \tau^+, \gamma_1, \tau^-$.
3. Randomly set a function $g : \tilde{S} \to \mathbb{R}$, such that

$$\begin{cases} g_{\tau^+} &= 2\pi \\ g_{\tau^-} &= \ \ 0 \end{cases}$$

g is random on other points.
4. The gradient of g is a closed 1-form on S, denoted as $\tilde{\omega}_2 = dg$. Compute a function $h : S \to \mathbb{R}$, such that

$$\nabla \cdot (\tilde{\omega_2} + dh) = 0,$$

with Neuman boundary condition

$$< \tilde{\omega}_2 + dh, \mathbf{n} >= 0,$$

where $\mathbf{n}$ is the normal to the boundaries γ_0, γ_1. Let $\omega_2 = \tilde{\omega}_2 + dh$, Fig. 4 (c) shows the closed harmonic 1-form ω_2.
5. Find a constant c, such that

$$\int_\tau * \omega_2 = c \int_\tau \omega_1,$$

where $*$ is the Hodge star operator. Then $\omega = c\omega_1 + \sqrt{-1}\omega_2$ is the desired holomorphic 1-form. Figure 4 (d) shows the holomorphic 1-form ω.

Simply Connected Domain. The construction of a conformal mapping for a simply connected domain (a topological disk) to the planar unit disk is very straight forward. Given a surface which is a topological disk, as shown in Fig. 1 (a), a small hole is punched at the point p. Then we map the punched disk, which is a topological annulus, to the planar annulus with the unit outer boundary. We choose a point q different from p, and map q to the real axis. We then shrink the size of the hole, and get another conformal mapping, still the outer boundary is with radius one, q is mapped to the real axis. If the size of the hole shrinks to zero, the conformal mapping of the annulus converges to the conformal mapping, such that p is mapped to the origin, q is mapped to the real axis. Figure 5 shows the conformal mapping result for a semi-cortex surface to the unit planar disk.

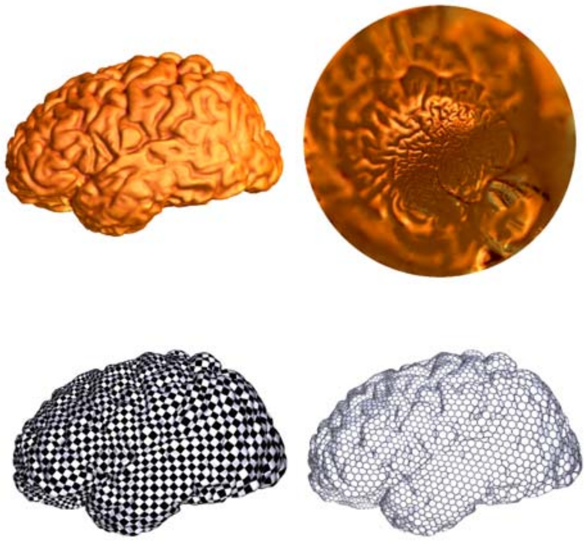


Fig. 5. Conformal mapping for simply connected domain

Multiply-Connected Domains. The construction of conformal mapping from a multiply connected domain to a planar disk with circular holes can be deduced to the doubly connected case by using Koebe's method [26]. As shown in Fig. 9, the human face surface in (a) is segmented to D_0, D_1, D_2, D_3, where D_0 is a multiply connected domain. The domain is conformally mapped to a planar disk with circular holes as shown in (b). The conformality is illustrated by texture mapping a circle packing pattern in (c), where all the small circles on the texture are mapped to circles on the surface.

The algorithm is as follows. The face surface is denoted as S. First, D_1 is removed from the surface S as shown in (d). Then the doubly connected domain $S - D_1$ is conformally mapped to a planar annulus as shown in (e). Then D_1 is glued back to the planar annulus by solving a harmonic map, such that the boundary of D_1 is mapped to the inner circle of the planar annulus. By using a scaling and a Möbius transformation, the conformal mapping is normalized, such that the point $p \in S$ is mapped to the origin, and the boundary point $q \in S$ is mapped to $+1$. The result is shown in (f). A similar procedure is repeated for segment D_2 shown in (g), (h) and (i), and for segment D_3 in (j), (k) and (l). By repeating this procedure, the images of the inner holes are getting rounder and rounder. Eventually, the images converge to a planar disk with circular holes where p is mapped to the origin, q is mapped to $+1$, as shown in (b).

The detailed proof for the convergence of Koebe's algorithm can be found in [26]. The convergence is very fast. In the example, the procedure only goes through each segment $D_k, k = 1, 2, 3$ once, and the images of the inner boundaries are very close to circles as shown in Fig. 9 (l).

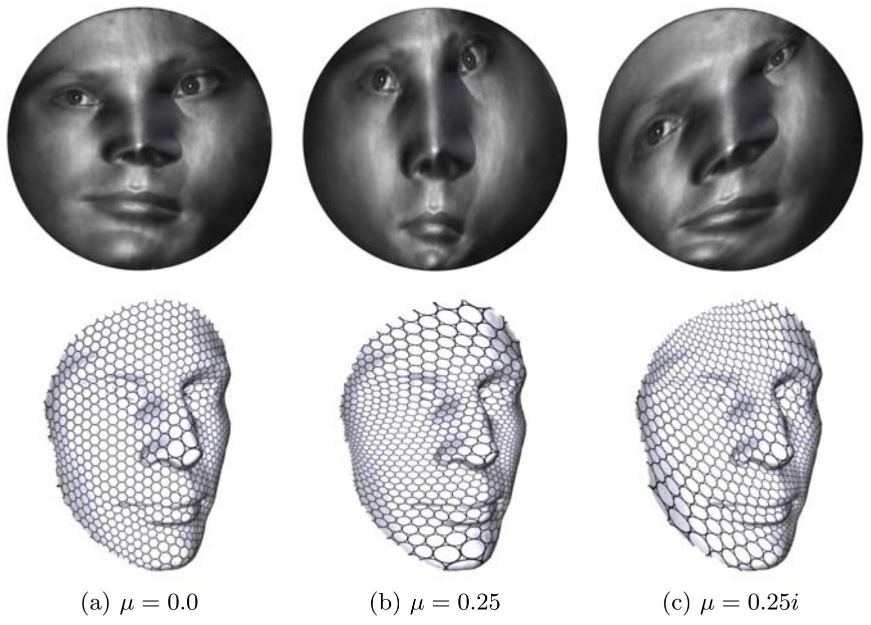


Fig. 6. Quasi-Conformal mapping for a simply connected domain

4.2 Quasi-Conformal Maps

Simply Connected Domain. Given a surface S, which is a topological disk, two points $p, q \in S$, we want to compute a quasi-conformal map $\phi : S \to \mathbb{D}$, such that ϕ satisfies the Beltrami equation

$$\frac{\partial \phi}{\partial z} = \mu(z) \frac{\partial \phi}{\partial \overline{z}},$$

where z is the isothermal coordinate of S. Furthermore

$$\phi(p) = 0, \phi(q) \in \mathbb{R}^+.$$

First, we compute a conformal map $\phi_1 : S \to \mathbb{D}$, where $\mathbb{D}$ is the planar unit disk with the canonical Euclidean metric:

$$\mathbf{g}_0 = dz d\overline{z}.$$

Assume the Beltrami coefficient is defined on $\mathbb{D}$, $\mu : \mathbb{D} \to \mathbb{C}$, then we construct a new Riemannian metric $\mathbf{g}$ for $\mathbb{D}$,

$$\mathbf{g}(z) = |dz + \mu(z) d\overline{z}|^2. \tag{4}$$

We compute a conformal map $\phi_2 : (\mathbb{D}, \mathbf{g}) \to (\mathbb{D}, \mathbf{g}_0)$, such that $\phi_2(\phi_1(p)) = 0, \phi_2(\phi_1(q)) \in \mathbb{R}^+$. Then the quasi-conformal map is given by $\phi = \phi_2 \circ \phi_1$, $\phi : S \to (\mathbb{D}, \mathbf{g}_0)$, which satisfies the Beltrami equation.

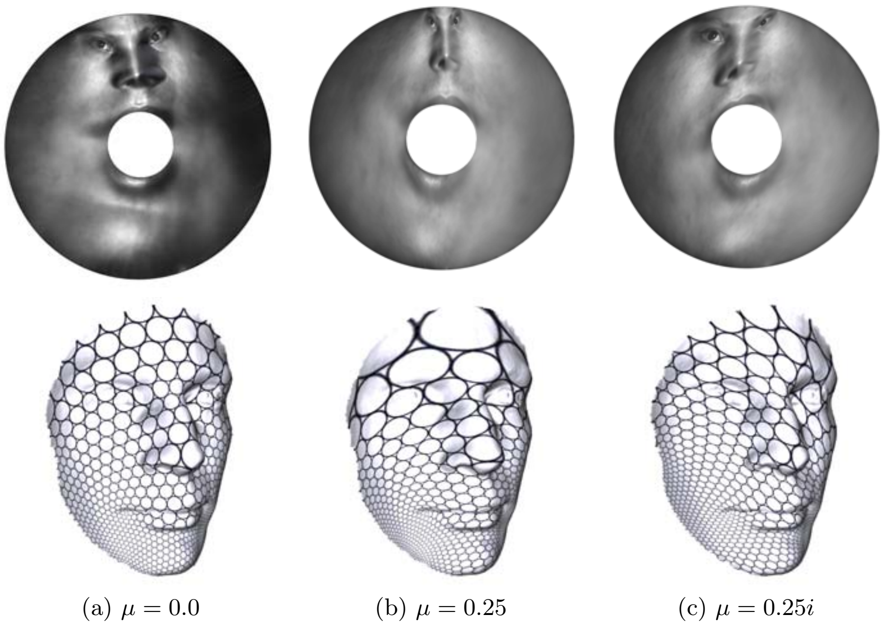


Fig. 7. Quasi-Conformal mapping for a doubly connected domain

Doubly Connected Domain. Similarly, if S is a topological annulus, first a conformal map $\phi_1 : S \to A^2$ is computed using the algorithm discussed above, where A^2 is a planar annulus with canonical Euclidean metric $\mathbf{g}_0$, whose boundaries are concentric circles. Similarly, a new metric $\mathbf{g}$ is constructed for T^2 using Formula 4. Then a conformal map $\phi_2 : (A^2, \mathbf{g}) \to (\tilde{A}^2, \mathbf{g}_0)$ is computed, where $\tilde{A}^2$ is another planar annulus with canonical Euclidean metric and concentric boundary circles. The shape of $\tilde{A}^2$ is determined automatically by $(T^2, \mathbf{g})$. The quasi-conformal map ϕ is given by the composition of ϕ_1 and ϕ_2, $\phi : \phi_2 \circ \phi_1 : S \to (\tilde{A}^2, \mathbf{g}_0)$.

Multiply Connected Domain. Suppose S is a multiply connected domain with boundaries

$$\partial S = \gamma_0 - \gamma_1 - \cdots - \gamma_n$$

First, we fill all the holes with topological disks $D_1, D_2, \cdots D_n$, such that $\partial D_k = \gamma_k$. Then $\tilde{S} = S \cup \{\bigcup_k D_k\}$ is a simply connected domain. Then we compute a conformal map $\phi_1 : \tilde{S} \to \mathbb{D}$. We can extend the Beltrami coefficient μ to $\tilde{\mu}$ on $\mathbb{D}$ from $\phi_1(S)$ to $\phi_1(D_k)$'s by solving harmonic functions using Dirichlet boundary condition,

$$\tilde{\mu}(p) = \mu(p), \forall p \in S; \Delta \tilde{\mu}(p) = 0, p \in \bigcup_k D_k.$$

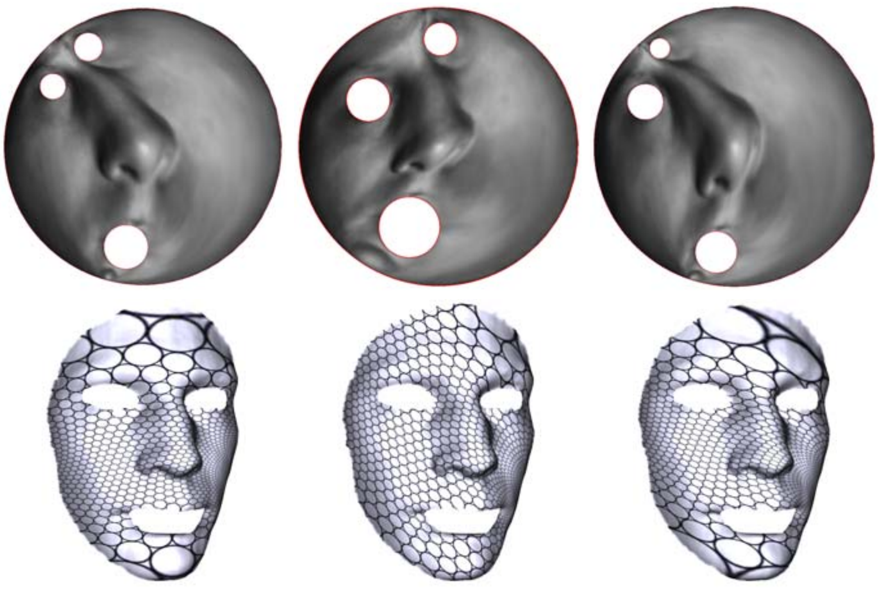


Fig. 8. Quasi-Conformal mapping for a multiply connected domain

Then we can compute a quasi-conformal map $\phi_2 : (\mathbb{D}, \mathbf{g}_0) \to (\mathbb{D}, \mathbf{g}_0)$, such that

$$\frac{\partial \phi_2}{\partial \bar{z}} = \tilde{\mu}(z) \frac{\partial \phi_2}{\partial z}$$

Then we can compute a conformal map from a planar multiply connected domain $(\phi_2 \circ \phi_1(S), \mathbf{g}_0)$ with canonical Euclidean metric to a planar disk with circular holes, $\phi_3 : \phi_2 \circ \phi_1(S) \to (\mathbb{D}, \mathbf{g}_0)$. The desired quasi-conformal map ϕ is the composition

$$\phi = \phi_3 \circ \phi_2 \circ \phi_1 : S \to (\mathbb{D}, \mathbf{g}_0),$$

which satisfies the Beltrami equation (see Eqn. 1).

4.3 Discrete Algorithm

In practice, all the surfaces are approximated by simplicial-complexes embedded in $\mathbb{R}^3$, denoted as $M = (V, E, F)$, where V, E, F are the sets of vertices, edges and faces respectively. We use v_i to denote the i-th vertex, $[v_i, v_j]$ the halfedge from v_i to v_j, $[v_i, v_j, v_k]$ the face with vertices v_i, v_j, v_k, sorted counter clockwisely. A discrete 0-form $f : V \to \mathbb{R}$ is a real valued function defined on the vertices. A discrete 1-form $\omega : E \to \mathbb{R}$, defined on edges. The exterior differentiation operator is defined as

$$df([v_i, v_j]) = f(v_j) - f(v_i).$$

and

$$d\omega([v_i, v_j, v_k]) = \omega([v_i, v_j]) + \omega([v_j, v_k]) + \omega([v_k, v_i]).$$

ω is a closed 1-form, if $d\omega = 0$.

Suppose two faces $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$ share an edge $[v_i, v_j]$. The weight on edge $[v_i, v_j]$ is defined as

$$w_{ij} = \cot\theta_k^{ij} + \cot\theta_l^{ij},$$

where θ_k^{ij} is the corner angle at the vertex v_k in the face $[v_i, v_j, v_k]$: θ_l^{ij} is defined in the similar way. If the edge $[v_i, v_j]$ is on the boundary and only attached to $[v_i, v_j, v_k]$ then the edge weight is defined as

$$w_{ij} = \cot\theta_k^{ij}.$$

The discrete harmonic energy of a 0-form $f : V \to \mathbb{R}$ is defined as

$$E(f) = \sum_{[v_i, v_j]} w_{ij}(f(v_j) - f(v_i))^2.$$

The *divergence operator* is defined as

$$\nabla \cdot \omega(v_j) = \sum_j w_{ij}\omega([v_i, v_j]),$$

where v_j are all the vertices connecting to v_i. A discrete harmonic 1-form satisfies $\nabla \cdot \omega = 0$.

Let ω be a closed 1-form. A face $[v_i, v_j, v_k]$ can be isometrically embedded on the plane, and then ω has local representation as $\omega = adx + bdy$. The *Hodge star operator* is defined as ${}^*\omega = -bdx + ady$, where a, b are two real numbers and the *wedge product* between two closed 1-forms $\omega_k = a_k dx + b_k dy,\ k = 1, 2$ is given by

$$\omega_1 \wedge \omega_2 = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} dx \wedge dy \tag{5}$$

The inner product between ω_1 and ω_2 is defined as

$$< \omega_1, \omega_2 >= \int_M \omega_1 \wedge {}^*\omega_2. \tag{6}$$

Let M be a triangular mesh of genus g. We first compute its first homology group $H_1(M, \mathbb{Z})$ basis by CW-cell decomposition, which we denote by $\{\gamma_1, \gamma_2, \cdots, \gamma_{2g}\}$. Then we compute the dual basis for the cohomology group $H^1(M, \mathbb{R})$, $\{\tau_1, \tau_2, \cdots, \tau_{2g}\}$, such that

$$\int_{\gamma_i} \tau_j = \delta_{ij}.$$

Then we find 0-forms $g_i : M \to \mathbb{R}$, such that

$$\nabla \cdot (\tau_i + dg_i) = 0, i = 1, 2, \cdots, 2g,$$

then $\omega_i = \tau_i + dg_i$'s are harmonic 1-forms. The Hodge star of a harmonic 1-form is also harmonic, therefore

$$^*\omega_i = \sum_j \lambda_{ij}\omega_j,$$

the coefficient of λ_{ij} can be computed by solving the linear system

$$< \omega_k, \omega_i >= \sum_j \lambda_{ij} \int_M \omega_k \wedge \omega_j, k = 1, 2, \cdots, 2g,$$

the left hand side is computed using formula 6, the right hand side by Formula 5. The holomorphic 1-form basis is given by

$$\{\omega_1 + \sqrt{-1}^*\omega_1, \omega_2 + \sqrt{-1}^*\omega_2, \cdots, \omega_{2g} + \sqrt{-1}^*\omega_{2g}\}.$$

Surfaces with boundaries can be converted to symmetric closed surfaces by double covering and therefore their holomorphic 1-form group basis can be computed in the similar way. For details, we refer readers to Gu and Yau's work [11]. Conformal mappings between genus zero surfaces with boundaries can be carried out using holomorphic 1-forms, which can be approximated in the discrete setting.

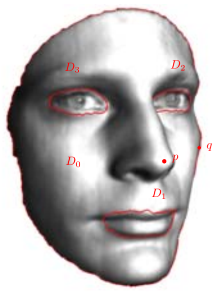
By the above discrete forms and the operators, the algorithms described in previous two subsections can be carried out in the discrete setting.

5 Experimental Results

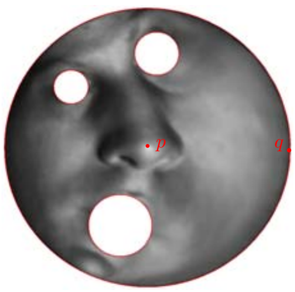
Most of the surfaces are captured using 3D scanner based on phase shifting principle [27,28]. The scanned resolution is 640×480, at 60 frames per second. The triangulations are directly determined by the pixel grid structure of the scanned images. We did not perform preprocessing operations, such as smoothing, denoising and mesh simplification. The raw data sets are computed using the algorithms described above. This demonstrates the robustness of our method.

All the algorithms are implemented using generic C++ on a Windows XP platform with Dual 2.33GHz CPU and 3.98 GB of RAM. The linear systems are in general symmetric and positive definite. We use the Matlab C++ library as the linear solver.

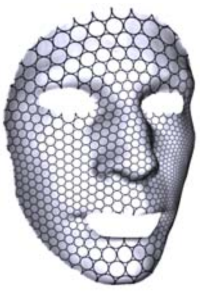
In our experiment, the surfaces are described as triangular meshes. The Beltrami coefficient μ is a function over the whole surface. In Fig. 1(e), the μ is set to be $x + iy$ for each vertex. The quasi-conformal mappings are constructed using holomorphic differential forms, which is general for surfaces with different topologies. In other cases, it is set to be a constant complex number. Table 1 lists the computational time for each case with different Beltrami coefficient μ. For large meshes with $160k$ faces, the processing time for Koebe's method [26] with multiple iterations is about 2.5 minutes. This demonstrates the efficiency of our algorithm.



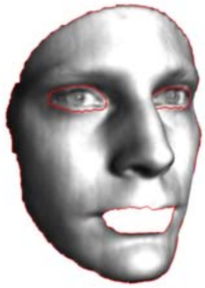
(a) Original surface



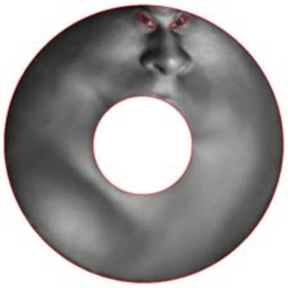
(b) Planar domain



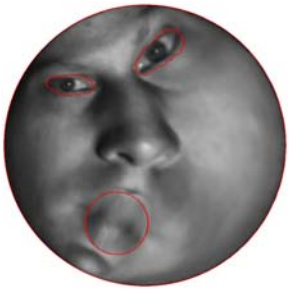
(c) Texture mapping



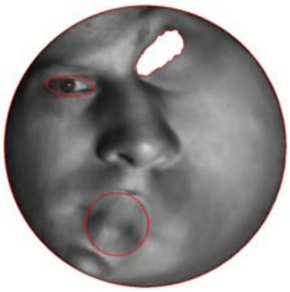
(d) D_1 is removed



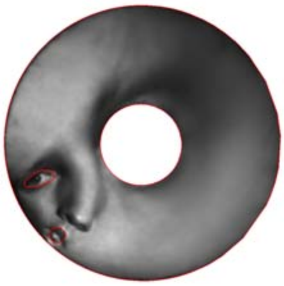
(e) Conformal mapping for $S - D_1$



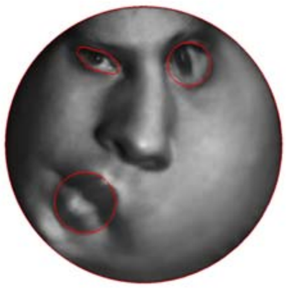
(f) Möbius transform D_1 is glued back



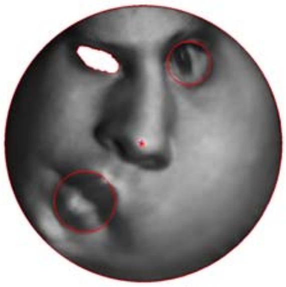
(g) D_2 is removed



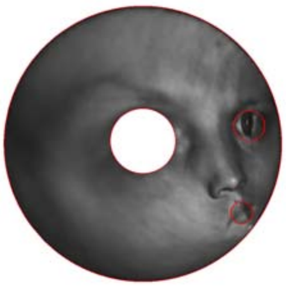
(h) Conformal mapping for $S - D_2$



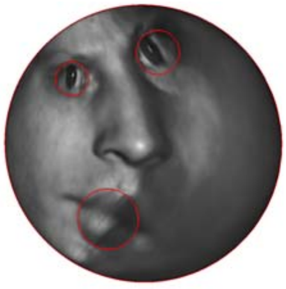
(i) Möbius transform D_2 is glued back



(j) D_3 is removed



(k) Conformal mapping for $S - D_3$



(l) Möbius transform D_3 is glued back

Fig. 9. Conformal mapping for a multiply connected domain using Koebe's algorithm

Table 1. Computational Time

Figure	#Vertex	#Face	#Boundary	Beltrami Coefficient μ	Iterations	Time
Fig. 1(e)	80593	160054	1	$x + iy$	1	102s
Fig. 6(a)	80593	160054	1	$0.00 + 0.00i$	1	73s
Fig. 6(b)	80593	160054	1	$0.25 + 0.00i$	1	110s
Fig. 6(c)	80593	160054	1	$0.00 + 0.25i$	1	105s
Fig. 7(a)	80724	160054	2	$0.00 + 0.00i$	1	78s
Fig. 7(b)	80724	160054	2	$0.25 + 0.00i$	1	110s
Fig. 7(c)	80724	160054	2	$0.00 + 0.25i$	1	112s
Fig. 9(c)	15160	29974	4	$0.00 + 0.0i$	2	156s
Fig. 8(a)	15160	29974	4	$0.25 + 0.0i$	2	160s
Fig. 8(b)	15160	29974	4	$0.00 + 0.25i$	2	156s
Fig. 8(c)	15160	29974	4	$0.25 + 0.25i$	2	157s

Besides the scanned data sets, we also test human cortex surface as shown in Fig. 5. The surface is reconstructed from MRI images. Furthermore, we tested some synthetic data to verify the accuracy of our method. The algorithm recovers the correct solution with high accuracy.

6 Conclusion and Future Works

This work introduces a rigorous method for computing quasi-conformal mappings by solving Beltrami equations. The method is efficient and robust. The major point is to deform the Riemannian metric by the Beltrami coefficient and convert a quasi-conformal mapping to a conformal mapping. In the current work, the method is based on holomorphic differentials, and it can be directly generalized to discrete Ricci flow [29,3,4] and to the Yamabe flow method [30,31,32].

In the future, we plan to apply quasi-conformal mappings to shape registration, surface comparison, shape recognition, and many other applications in computer graphics, computer vision, medical imaging and geometric modeling. Also, we will explore rigorous algorithms for computing extremal quasi-conformal maps.

Acknowledgement

The authors would like to thank Prof. Ralph Martin, Prof.Arie Kaufman and Prof. Hong Qin for the discussion. This work is partially supported by NSF 0448399 career: Conformal Geometry Applied to Shape Analysis and Geometric Modeling and NSF 0626223: Discrete Curvature Flow.

References

1. Floater, M.S., Hormann, K.: Surface parameterization: a tutorial and survey. In: Advances in Multiresolution for Geometric Modelling, pp. 157–186. Springer, Heidelberg (2005)

2. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3d models. ACM Transactions on Graphics 23(3), 861–869 (2004)
3. Jin, M., Kim, J., Gu, X.D.: Discrete surface ricci flow: Theory and applications. In: Martin, R., Sabin, M.A., Winkler, J.R. (eds.) Mathematics of Surfaces 2007. LNCS, vol. 4647, pp. 209–232. Springer, Heidelberg (2007)
4. Jin, M., Kim, J., Luo, F., Gu, X.: Discrete surface ricci flow. IEEE Transaction on Visualization and Computer Graphics 14(5), 1030–1043 (2008)
5. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 2(1), 15–36 (1993)
6. Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: SIGGRAPH 2002, pp. 362–371 (2002)
7. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic parameterizations of surface meshes. Computer Graphics Forum (Proc. Eurographics 2002) 21(3), 209–218 (2002)
8. Floater, M.S.: Mean value coordinates. Computer Aided Geometric Design 20(1), 19–27 (2003)
9. Gotsman, C., Gu, X., Sheffer, A.: Fundamentals of spherical parameterization for 3D meshes. ACM Transactions on Graphics 22(3), 358–363 (2003)
10. Gu, X., Wang, Y., Chan, T.F., Thompson, P.M., Yau, S.-T.: Genus zero surface conformal mapping and its application to brain surface mapping. IEEE Trans. Med. Imaging 23(8), 949–958 (2004)
11. Gu, X., Yau, S.-T.: Global conformal parameterization. In: Symposium on Geometry Processing, pp. 127–137 (2003)
12. Mercat, C.: Discrete riemann surfaces and the ising model. Communications in Mathematical Physics 218(1), 177–216 (2004)
13. Hirani, A.N.: Discrete exterior calculus. PhD thesis, California Institute of Technology (2003)
14. Jin, M., Wang, Y., Yau, S.-T., Gu, X.: Optimal global conformal surface parameterization. In: IEEE Visualization 2004, pp. 267–274 (2004)
15. Gortler, S.J., Gotsman, C., Thurston, D.: Discrete one-forms on meshes and applications to 3D mesh parameterization. Computer Aided Geometric Design 23(2), 83–112 (2005)
16. Tong, Y., Alliez, P., Cohen-Steiner, D., Desbrun, M.: Designing quadrangulations with discrete harmonic forms. In: Symposium on Geometry Processing, pp. 201–210 (2006)
17. Tewari, G., Gotsman, C., Gortler, S.J.: Meshing genus-1 point clouds using discrete one-forms. Comput. Graph. 30(6), 917–926 (2006)
18. Desbrun, M.: Discrete differential forms and applications to surface tiling. In: SCG 2006: Proceedings of the twenty-second annual symposium on Computational geometry, p. 40. ACM, New York (2006)
19. Tong, Y., Alliez, P., Cohen-Steiner, D., Desbrun, M.: Designing quadrangulations with discrete harmonic forms. In: SGP 2006: Proceedings of the fourth Eurographics symposium on Geometry processing, pp. 201–210 (2006)
20. Hong, W., Gu, X., Qiu, F., Jin, M., Kaufman, A.E.: Conformal virtual colon flattening. In: Symposium on Solid and Physical Modeling, pp. 85–93 (2006)
21. Wang, S., Wang, Y., Jin, M., Gu, X.D., Samaras, D.: Conformal geometry and its applications on 3d shape matching, recognition, and stitching. IEEE Trans. Pattern Anal. Mach. Intell. 29(7), 1209–1220 (2007)
22. Zeng, W., Zeng, Y., Wang, Y., Yin, X., Gu, X., Samaras, D.: 3d non-rigid surface matching and registration based on holomorphic differentials. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 1–14. Springer, Heidelberg (2008)

23. Gu, X., He, Y., Qin, H.: Manifold splines. Graphical Models 68(3), 237–254 (2006)
24. Guggenheimer, H.W.: Differential Geometry. Dover Publications (1977)
25. Farkas, H.M., Kra, I.: Riemann Surfaces. Springer, Heidelberg (2004)
26. Henrici, P.: Applide and Computational Complex Analysis, Discrete Fourier Analysis, Cauchy Integrals, Construction of Conformal Maps, Univalent Functions, vol. 3. Wiley-Interscience, Hoboken (1993)
27. Wang, Y., Gupta, M., Zhang, S., Samaras, D., Huang, P.: High resolution tracking of non-rigid 3D motion of densely sampled data using harmonic maps. In: Proc. International Conference on Computer Vision, pp. 388–395 (2005)
28. Gu, X., Zhang, S., Huang, P., Zhang, L., Yau, S.-T., Martin, R.: Holoimages. In: SPM 2006: Proceedings of the 2006 ACM symposium on Solid and physical modeling, pp. 129–138 (2006)
29. Chow, B., Luo, F.: Combinatorial ricci flows on surfaces. Journal Differential Geometry 63(1), 97–129 (2003)
30. Luo, F.: Combinatorial yamabe flow on surfaces. Commun. Contemp. Math. 6(5), 765–780 (2004)
31. Springborn, B., Schröder, P., Pinkall, U.: Conformal equivalence of triangle meshes. ACM Trans. Graph. 27(3), 1–11 (2008)
32. Zeng, W., Jin, M., Luo, F., Gu, X.: Computing canonical homotopy class representative using hyperbolic structure. In: IEEE International Conference on Shape Modeling and Applications (SMI 2009) (2009)

Author Index