ELSEVIER

Computer Aided Geometric Design 18 (2001) 359–379

COMPUTER AIDED GEOMETRIC DESIGN

www.elsevier.com/locate/comaid

Geometric fairing of irregular meshes for free-form surface design

Robert Schneider *, Leif Kobbelt

Max-Planck Institute for Computer Sciences, Stuhlsatzenhausweg 85, D-66123 Saarbrücken, Germany

Received July 2000; revised December 2000

Abstract

In this paper we present a new algorithm for smoothing arbitrary triangle meshes while satisfying G^1 boundary conditions. The algorithm is based on solving a nonlinear fourth order partial differential equation (PDE) that only depends on intrinsic surface properties instead of being derived from a particular surface parameterization. This continuous PDE has a (representation-independent) well-defined solution which we approximate by our triangle mesh. Hence, changing the mesh complexity (refinement) or the mesh connectivity (remeshing) leads to just another discretization of the same smooth surface and doesn't affect the resulting geometric shape beyond this. This is typically not true for filter-based mesh smoothing algorithms. To simplify the computation we factorize the fourth order PDE into a set of two nested second order problems thus avoiding the estimation of higher order derivatives. Further acceleration is achieved by applying multigrid techniques on a fine-to-coarse hierarchical mesh representation.

Keywords: Discrete fairing; Free-form modeling; PDE method; Parameterization independence

1. Introduction

Although piecewise polynomial patches are still the dominating free-form surface representation in engineering applications, the use of triangle meshes has become increasingly important—especially with the rising complexity of 3D models. A general approach to generate free-form surfaces that satisfy aesthetic requirements is *surface fairing* where auxiliary degrees of freedom are used to improve the global distribution of curvature (or optimize any other quality criterion). For triangle meshes this type of optimization has two different aspects. First, the triangle mesh should have *outer fairness*,

* Corresponding author.
E-mail addresses: schneider@mpi-sb.mpg.de (R. Schneider), kobbelt@mpi-sb.mpg.de (L. Kobbelt).

0167-8396/01/$ – see front matter
PII: S0167-8396(01)00036-X

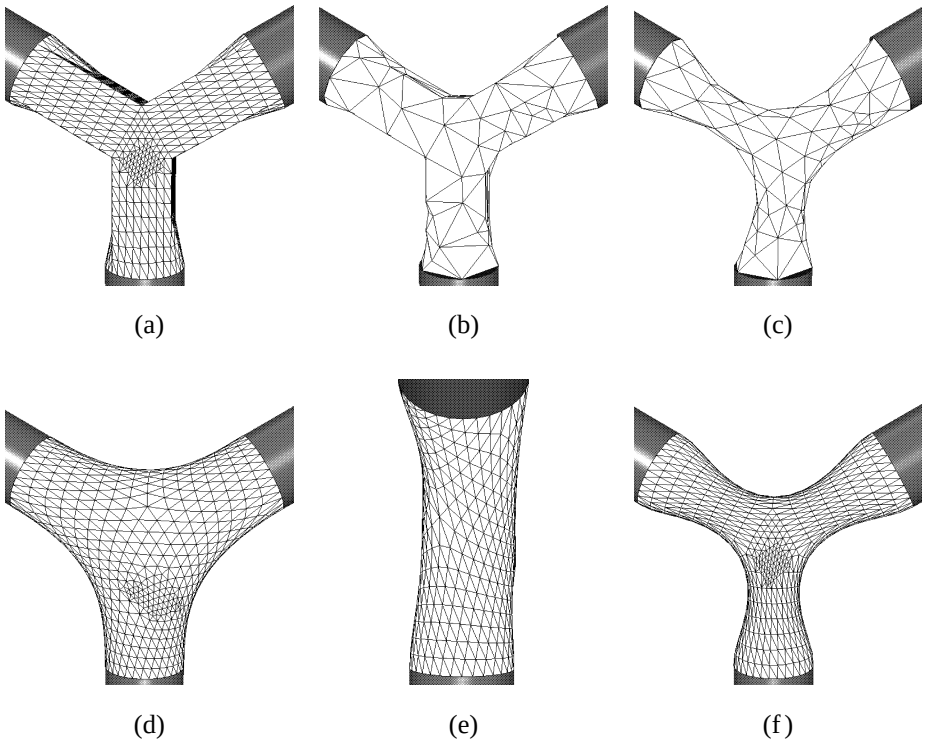


Fig. 1. Mesh fairing based on discretizing the equation $\Delta^2 f = 0$ using two different local parameterization strategies. The boundary condition is determined by 3 cylinders that are arranged symmetrically. (a) Shows the original mesh (920 vertices) and (b) a reduced version with 118 vertices. In (c), (d) and (e) we see the results if local uniform parameterization is assumed. In (f) we used a discretization of a laplacian that was derived from a discrete harmonic map to fair the original mesh. As we can see, the mesh size and the local parameterization strategy heavily influence the resulting surfaces. The rectangular patch introduced in the original mesh leads to local as well as global shape distortions and prevents a symmetric solution.

i.e., the imaginary surface that is approximated by the mesh should be optimal with respect to curvature distribution. Additionally, the mesh should have *inner fairness* which means that the distribution of mesh vertices *within* the surface and the shape of the individual faces should be good according to application dependent requirements.

The result of recent mesh fairing algorithms usually depends on the underlying mesh connectivity, thus the inner fairness influences the resulting outer fairness in some way and even for the most sophisticated schemes there is no guarantee that the surfaces are free of parameterization artifacts (Fig. 1). A solution to this shortcoming is to use a fairing algorithm that is able to separate the two fairness types (Fig. 2). This is achieved by using an outer fairness measure that is based on intrinsic surface properties only, i.e., properties that depend on the geometry alone. Unfortunately, intrinsic fairing is a nonlinear problem, and while the linear fairing operators are highly efficient and in general mathematically

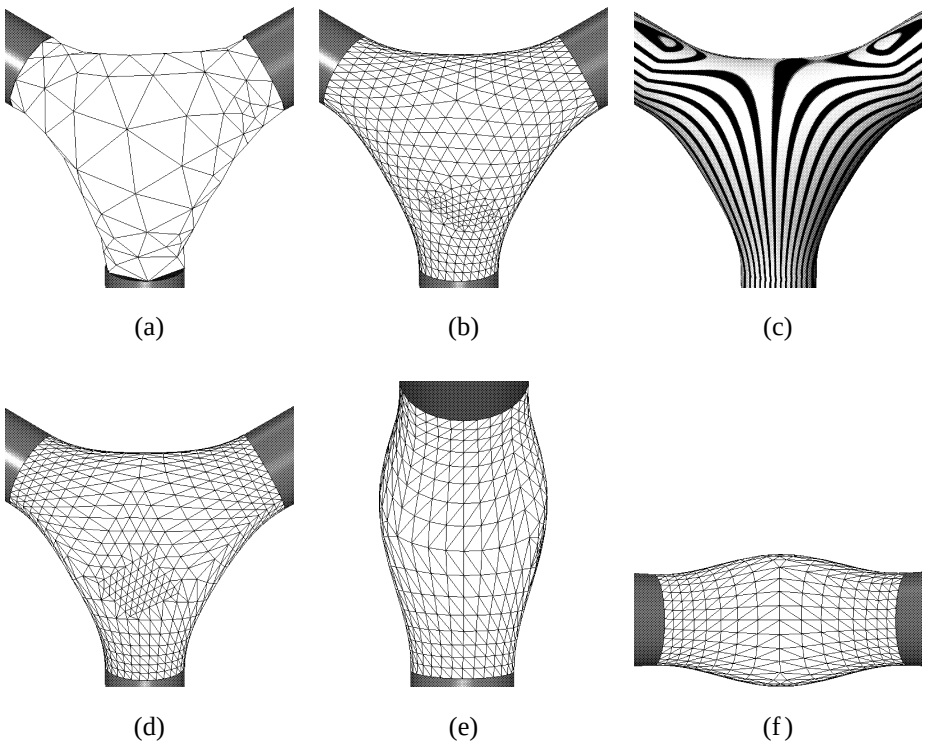


Fig. 2. Our new fairing approach applied on the example shown in Fig. 1. In (a), (b) and (c) we optimized the inner fairness with respect to a local uniform parameterization. In (d), (e) and (f) the mesh is discrete conformal to the original mesh in Fig. 1(a). We can see that the chosen inner fairness strategy and the mesh structure have only marginal influence on the shape. The influence of the mesh size is also much smaller than in the linear setting. The continuity of the refection lines in (c) indicates G^1 continuity at the boundary.

well understood, the analysis in the nonlinear case is much more difficult. To the authors knowledge, even for the rather simple fairing functional

$$\int_A \kappa_1^2 + \kappa_2^2 \, \mathrm{d}A, \tag{1}$$

leading to a minimal energy surface (MES), it is still unknown if a solution always exists within specific smoothness classes.

In this paper the separation of outer and inner mesh fairness and parameter independence is achieved by using an outer fairness concept that is based on a discrete solution of an intrinsic PDE. The PDE we choose is of fourth order and leads to surfaces of high quality. To speed up the construction scheme, we factorize the fourth order PDE into two second order problems and use a hierarchic mesh representation to enable multigrid techniques.

The paper is organized as follows: Section 2 reviews related work on mesh fairing. In Section 3 we present the concept of our intrinsic fairing approach. Section 4 defines the

notation we use throughout the paper. In Sections 5 and 6 we show how to discretize the necessary intrinsics. Finally, in Section 7 we present the details of our algorithm.

2. Mesh fairing—previous work

Most mesh fairing schemes are based on linear operators. While this leads to simple and fast algorithms, there is a serious consequence: The outer fairness and therefore the shape of the resulting mesh highly depends on the chosen parameterization strategy. As a consequence it is not possible to separate the outer and inner fairness concept in this case. If we change the inner fairness strategy we have to change the parameterization and will therefore also change the outer fairness (see Fig. 1(d) and (f)).

There are two types of fairing algorithms, depending on whether the fairing is based on the calculation of a well defined surface or whether it is based on filtering operations.

The standard approach for fair surface construction is based on the idea to minimize a fairness metric, punishing features that are inconsistent with the fairness principle of the simplest shape (Burchard et al., 1994). Applied to meshes this leads to the discrete fairing approach proposed by Kobbelt (1997). Besides of energy minimization, during the last years various other linear mesh fairing schemes have been developed.

A very effective method for smoothing polyhedral surfaces is the discrete diffusion flow (Taubin, 1995a). Here the idea is to iteratively update each vertex

$$q_i' = q_i + \lambda \Delta q_i \tag{2}$$

by adding a displacement vector that is a scaled discrete laplacian Δq_i. For stability reasons the scale factor λ has to satisfy $0 < \lambda < 1$. A diffusion flow that is unconditionally stable and hence enables larger scale factors was presented by Desbrun et al. (1999).

The main purpose of the diffusion flow is to remove the high frequencies in noisy meshes. Since the equilibrium surface of the flow only enables C^0 boundary conditions, (2) is of only limited use in surface design. To enable smooth boundary conditions one has to consider diffusion equations of higher order. Taubin (1995a) proposed to combine two such smoothing steps with positive and negative scale factors and developed an algorithm that enables various interpolation constraints. Another idea that enables smooth boundary conditions is to use higher powers of the laplacian in the diffusion flow. As a good trade-off between efficiency and quality one can chose the bilaplacian flow, enabling C^1 boundary conditions.

Another mesh fairing approach is based on the idea to discretize the PDE approach of Bloor and Wilson (1990). In (Kobbelt et al., 1998b) it was proposed to discretize

$$\Delta^2 f = 0, \tag{3}$$

where Δ is the laplacian operator, to create surfaces satisfying prescribed C^1 boundary conditions. This equation results if we apply variational calculus to the thin plate energy

$$\iint f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 \,\mathrm{d}x\,\mathrm{d}y$$

to describe the minimum of the functional.

Fairing based on some kind of diffusion flow and fairing based on the discretization of the PDE (3) that describes the smooth solution are tightly connected problems, since (3) is the equilibrium of the bilaplacian flow. In both cases the discretization of the laplacian plays the central role. During the last years various linear discretizations of the laplacian have been developed (Taubin, 1995a; Kobbelt et al., 1998b; Desbrun et al., 1999; Guskov et al., 1999), differing in how the geometry of the mesh affects the discretization. The chosen discretization determines the inner mesh fairness, but it also greatly influences the shape of the resulting mesh (Fig. 1).

To make the outer fairness independent of the mesh parameterization, other approaches are based on intrinsic surface properties. These approaches lead to nonlinear fairing schemes.

In (Welch and Witkin, 1994) a mesh fairing algorithm was presented that enables G^1 boundary conditions based on the idea to minimize the total curvature (1) thus punishing large curvature values. For an isometric parameterization this approach coincides with minimizing the thin plate energy (3). The necessary intrinsic curvature values were estimated using local quadratic approximations over local planar parameterizations. However, in constellations where the local quadratic approximations are not uniquely defined, their approach may lead to stability problems.

An intrinsic diffusion operator using a discrete mean curvature flow was presented by Desbrun et al. (1999), leading to an excellent noise reduction algorithm. While this fairing algorithm is mainly designed for outer fairness without tangential shift, in (Ohtake et al., 2000) this algorithm was combined with an inner fairness criterion that leads to more regular meshes. Since these algorithms converge to a discrete minimal surface satisfying $H = 0$, they only enable C^0 boundary conditions and are therefore, as their linear counterparts, of only limited use in surface design. Again, the solution is to use curvature flows of higher order (Brakke, 1992; Hsu et al., 1992; Chopp and Sethian, 1999).

In (Schneider and Kobbelt, 2000) it was proposed to approximate a PDE based on intrinsics to create fair meshes with G^1 boundary conditions in the special case where the meshes have subdivision connectivity. As we will see in the next section, our method shares the principal idea to solve an intrinsic PDE, but enables much greater flexibility.

3. Our fairing concept

We present a fairing algorithm for arbitrary triangle meshes that enables G^1 boundary conditions (prescribed vertices and unit normals) and allows us to completely separate outer and inner fairness. To achieve this, the outer fairness strategy is based on the idea to discretize an intrinsic PDE. Given G^1 information at the boundary, the PDE defines a smooth surface satisfying the constraints, altering the mesh size, the connectivity or the inner fairness condition only produces another discretization of the same smooth surface and hence leads to geometries that will be close to each other (Fig. 2).

In practice this allows us to choose our inner fairness criterion freely. In this paper we restricted ourselves to two especially important cases. One leads to a regular mesh parameterization, the other produces meshes that are conformally parameterized to a

given initial polyhedron. The latter inner fairness method plays a fundamental role for fairing of textured meshes and in mesh editing, since it minimizes local distortions. An other consequence is that a coarse mesh already approximates the shape of the smooth surface that is implicitly defined by the PDE, so increasing the mesh size mainly improves the quality of the approximation not the quality of the underlying shape. We exploit this property to improve the efficiency of our construction algorithm by using multigrid methods for arbitrary meshes. The necessary mesh hierarchies are created using the progressive mesh representation as introduced by Hoppe (1996).

Following the set-up presented in (Schneider and Kobbelt, 2000) to define fair surfaces satisfying G^1 boundary constraints, the PDE that determines our outer fairness concept in this paper is defined as

$$\Delta_B H = 0, \tag{4}$$

which can be interpreted as surface analogon to the planar equation $\kappa'' = 0$ (the derivative of the curvature κ is with respect to arc length) leading to clothoid splines. Here Δ_B is the Laplace–Beltrami operator and H the mean curvature. The PDE only depends on geometric intrinsics and is comparatively simple for a forth order equation. Because of the mean value property of the laplacian, it is guaranteed that the extremal mean curvature values of a solution of (4) will be reached at the border and that there are no local extrema in the interior. Since constant mean curvature surfaces satisfy this equation, important basic shapes as spheres, cylinders and minimal surfaces with $H = 0$ can be reconstructed. Although the PDE we propose can be derived as a simplification of the Euler–Lagrange equation resulting from MESs (1), this is a fairing technique in its own right, which therefore doesn't have to be inferior to the MES approach. In fact, this approach even has some advantages over MESs, e.g., it reproduces cylinders, while it follows from the Euler–Lagrange equation of (1) that MESs do not reproduce that surface class. This again is completely analogous to the planar case. It is well known, that minimal energy curves do not reproduce circles, while a clothoid spline obviously does.

In (Schneider and Kobbelt, 2000) solutions of (4) were approximated by meshes in the special case where the meshes have subdivision connectivity and the boundary vertices could be regularly sampled on a smooth curve. The construction scheme was based on the idea to design an inner fairness criterion that partitions the surface in regular regions. Exploiting this regularity knowledge in advance, it was possible to assign a local planar parameterization to each vertex. Using these domains, the intrinsic values could be approximated by local quadratic approximations, thus the quality of the discretization depends on the quality of the estimated local planar parameterizations.

The algorithm presented in this paper doesn't have such limitations. There are also no restrictions concerning the mesh structure and the boundary vertices and we are free to choose an inner fairness criteria. Nevertheless, the resulting construction algorithm is fast and can be implemented compactly. Instead of trying to simulate the continuous case using local quadratic approximations, we use the discrete data of our minimization process directly.

4. Notation

We partition the vertices of a mesh M into two classes, denoting the set of all border vertices with $V_B(M)$ and the set of all vertices in the interior of M with $V_I(M)$. For each vertex q_i of M let $N(q_i)$ be the set of vertices q_j that are adjacent to q_i and let $D(q_i) = N(q_i) \cup \{q_i\}$ be the according 1-disk. Let $H_i = H(q_i)$ denote the discrete mean curvature and $\vec{n}_i = \vec{n}(q_i)$ the discrete unit normal vector at the vertex q_i. When it is clear which mean curvature value or normal vector is meant, in some cases we omit the index to increase the readability.

5. Discretization of the Laplace–Beltrami operator

The discretization relies on the fact that there is a tight connection between the Laplace–Beltrami operator Δ_B and the mean curvature normal of a surface. In (Desbrun et al., 2000) also an improved discretization can be found that uses a more detailed area calculation based on voronoi regions, but the following discretization is sufficient for our needs. Let $f : \mathbb{R}^2 \to \mathbb{R}^3$ be a parameterization of a surface, then it is well known that the following equation holds

$$\Delta_B f = 2H\vec{n}.$$

Exploiting this relation, a discretization of Δ_B follows directly from the mean curvature flow approach for arbitrary meshes that was presented by Desbrun et al. (1999). They showed that the mean curvature normal at a vertex q_i of a triangular mesh M can be discretized using its 1-neighborhood by

$$H\vec{n} = \frac{3}{4A} \sum_{q_j \in N(q_i)} (\cot\alpha_j + \cot\beta_j)(q_j - q_i), \tag{5}$$

where A is the sum of the triangle areas of the 1-disk at q_i and α_j and β_j are the triangle angles as shown in Fig. 3. Since the vertices can be interpreted as sample points of a smooth

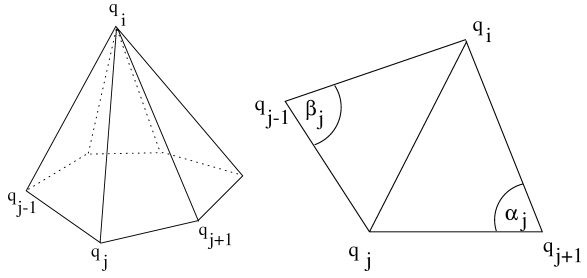


Fig. 3. The Laplace–Beltrami operator at the vertex q_i can be discretized using $D(q_i)$.

surface parameterized by f and exploiting the fact that a scaling factor does not influence the result, we can discretize the equation $\Delta_B H = 0$ at a vertex q_i as

$$\sum_{q_j \in N(q_i)} (\cot\alpha_j + \cot\beta_j)(H_i - H_j) = 0.$$

If this equation is satisfied at all inner vertices $q_i \in V_I(M)$ and if we further know all mean curvature values for the boundary vertices $V_B(M)$, this leads us to a sparse linear system in the unknowns $H_i \in V_I(M)$, whose matrix S has the coefficients

$$S_{ii} = \sum_{q_j \in N(q_i)} (\cot\alpha_j + \cot\beta_j), \tag{6}$$

$$S_{ij} = \begin{cases} -(\cot\alpha_j + \cot\beta_j), & q_j \in N(q_i) \cap V_I(M), \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The matrix S is symmetric and—as long as no triangle areas of the mesh M vanish—positive definite. To see this, we note that S also appears in a paper by Pinkall and Polthier (1993), where a stable construction algorithm for discrete minimal surfaces based on the idea to minimize the discrete Dirichlet energy of a mesh was presented. Hence, for an elegant proof of the mathematical structure of this matrix we can refer to this paper. It should be mentioned that S further appears in a paper about piecewise linear harmonic functions (Duchamp et al., 1997).

6. Discretization of the mean curvature

In this section we present a discretization of the mean curvature H_i at a vertex q_i that depends on the vertices in a local neighborhood. There are various techniques to discretize surface curvatures, but to be applicable for our construction algorithm, it is important that-for a given mesh connectivity-the discretization of H_i is a continuous function of those vertices.

Not all curvature discretization schemes satisfy this property. In (Welch and Witkin, 1994) it was proposed to discretize curvature information using local quadratic least square approximation over local planar parameterizations. However, it is well known that least square approximation fails, if the points in the parameterization plane lie on a curve of degree 2. To be able to detect such cases, the authors estimated the condition number of the least square problem in the Frobenius Norm and reduced the number of the basis functions if the problem was ill-conditioned. This approach is not only costly, but makes the discretization process discontinuous and thus leads to a potential instability in the mesh fairing algorithm.

To avoid analogous stability problems, in the following we propose a mean curvature discretization technique that satisfies the continuity criterion. Moreover, as we will show in Section 7.2, the presented scheme has other favorable properties that simplify the construction process.

At first glance, it seems tempting to use Eq. (5) to discretize the mean curvature, but this would only be applicable for inner vertices. However, in our construction algorithm

presented later in Section 7, we have to be able to discretize the mean curvature not only for inner vertices, but also for boundary vertices, where we have a completely different situation. Due to the boundary constraints, here we already know the tangent plane of the final surface, but we don't have a complete 1-disk. To avoid having mean curvature discretizations of different accuracy, we choose one technique that is able to handle both cases, so our method expects that a tangent plane is known at every vertex. At interior vertices where no normal vector is known in advance, we define it to be the normalized sum of the vector crossproducts of the incident triangle faces, in order to minimize square root operations.

6.1. Moreton and Séquin's curvature discretization algorithm

A curvature discretization algorithm that seems ideal for our needs was presented by Moreton and Séquin (1992). The idea of their approach is to use the fact that the normal curvature distribution cannot be arbitrary, but is determined by Euler's theorem (do Carmo, 1993).

Let $\vec{b}_x$ and $\vec{b}_y$ be an arbitrary orthonormal basis of the plane defined by the normal $\vec{n}$. To each vertex $q_j \in N(q_i)$ we can assign a unit direction vector $\vec{t}_j$ by projecting q_j into the plane and scaling this projection to unit length. For each q_j we can now estimate a normal curvature $\tilde{\kappa}_j$ as the inverse of the circle radius defined by q_i, q_j and t_j (Fig. 4)

$$\tilde{\kappa}_j = 2\frac{\langle q_j - q_i | \vec{n} \rangle}{\langle q_j - q_i | q_j - q_i \rangle}. \tag{8}$$

Using Euler's theorem, we can express the normal curvature κ_n for a direction $\vec{t}$ by the principal curvatures κ_1 and κ_2 and the principal curvature directions $\vec{e}_1$ and $\vec{e}_2$. Let t_x and t_y be the coordinates of $\vec{t}$ in the basis $\vec{b}_x, \vec{b}_y$ and let e_x and e_y be the coordinates of $\vec{e}_1$, then the normal curvature can be expressed as

$$\kappa_n = \begin{pmatrix} t_x \\ t_y \end{pmatrix}^{\mathrm{t}} \cdot K \cdot \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \tag{9}$$

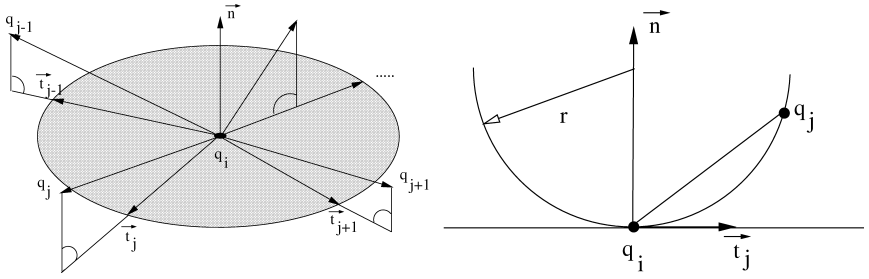


Fig. 4. Left: Projecting the neighborhood of q_i onto the plane defined by $\vec{n}$ and normalizing the results we get the normal curvature directions $\vec{t}_j$. Right: The normal curvature $\tilde{\kappa}_j$ along the direction $\vec{t}_j$ is discretized by interpolating q_i and q_j with a circle and using the inverse of the circle radius r as normal curvature. The center of the circle lies on the line defined by q_i and $\vec{n}$.

with

$$K = \begin{bmatrix} e_x & e_y \\ -e_y & e_x \end{bmatrix} \cdot \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix} \cdot \begin{bmatrix} e_x & e_y \\ -e_y & e_x \end{bmatrix}^{-1}.$$

The idea of Moreton and Séquin is to use the normal curvatures $\tilde{\kappa}_j$ to create a linear system and find estimates for the unknown principal curvature values by determining the least square solution. Let $t_{j,x}$ and $t_{j,y}$ denote the coordinates of $\vec{t}_j$ and let m be the valence of q_i, then we get by evaluating (9)

$$A\vec{x} = \vec{b},$$

where

$$A = \begin{bmatrix} t_{1,x}^2 & t_{1,x}t_{1,y} & t_{1,y}^2 \\ t_{2,x}^2 & t_{2,x}t_{2,y} & t_{2,y}^2 \\ \cdot & \cdot & \cdot \\ t_{m,x}^2 & t_{m,x}t_{m,y} & t_{m,y}^2 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} \tilde{\kappa}_1 \\ \tilde{\kappa}_2 \\ \cdot \\ \tilde{\kappa}_m \end{bmatrix}$$

and

$$\vec{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} e_x^2\kappa_1 + e_y^2\kappa_2 \\ 2e_xe_y(\kappa_1 - \kappa_2) \\ e_x^2\kappa_2 + e_y^2\kappa_1 \end{pmatrix}.$$

Since $x_0 + x_2 = \kappa_1 + \kappa_2$ this means the mean curvature is determined by

$$H = \frac{1}{2}(x_0 + x_2). \tag{10}$$

In our case the most efficient method to solve the least square problem is to use the normal equations approach (Golub and Van Loan, 1989), since this mainly involves to calculate the inverse of a symmetric 3×3 matrix. Using this approach, the least square solution $\vec{x}$ can be expressed as

$$\vec{x} = \left(A^{\mathrm{t}}A\right)^{-1}A^{\mathrm{t}}\vec{b}. \tag{11}$$

The cases when the matrix $A^{\mathrm{t}}A$ becomes singular can be detected by a simple criterion:

Lemma 1. *The matrix $A^{\mathrm{t}}A$ is singular if and only if all points $(t_{i,x}, t_{i,y})$ are intersection points of the unit circle with two straight lines through the origin.*

Proof. Since singularity of $A^{\mathrm{t}}A$ is equivalent with $\operatorname{rank} A < 3$, singularity occurs iff any 3 row vectors of A are linear dependent. Therefore, the proof is complete if we show that a matrix of type

$$\begin{bmatrix} t_{1,x}^2 & t_{1,x}t_{1,y} & t_{1,y}^2 \\ t_{2,x}^2 & t_{2,x}t_{2,y} & t_{2,y}^2 \\ t_{3,x}^2 & t_{3,x}t_{3,y} & t_{3,y}^2 \end{bmatrix}$$

with $t_{i,x}^2 + t_{i,y}^2 = 1$ is singular if and only if the tree points lie on two lines through the origin. This matrix is singular, if there are coefficients a_i which are not all zero such that

$$a_1 \begin{pmatrix} t_{1,x}^2 \\ t_{2,x}^2 \\ t_{3,x}^2 \end{pmatrix} + a_2 \begin{pmatrix} t_{1,x}t_{1,y} \\ t_{2,x}t_{2,y} \\ t_{3,x}t_{3,y} \end{pmatrix} + a_3 \begin{pmatrix} t_{1,y}^2 \\ t_{2,y}^2 \\ t_{3,y}^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

but this means each point $(t_{i,x}, t_{i,y})$ lies on a curve that satisfies $a_1x^2 + a_2xy + a_3y^2 = 0$. Depending on the coefficients a_i this equation characterizes a single point (the origin) or two lines through the origin. □

6.2. Singularity handling

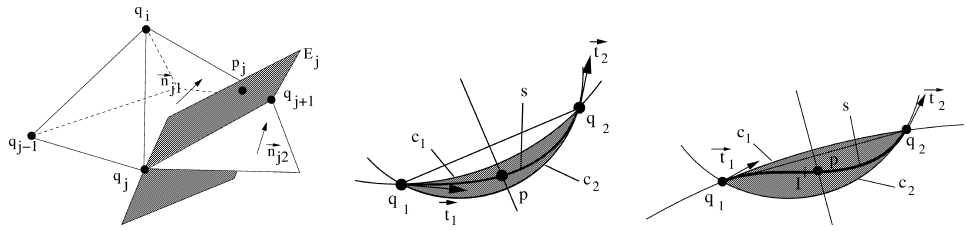
Perhaps the most obvious strategy to avoid singularities of A^tA is to simply check whether the criterion presented in Lemma 1 is satisfied and to apply a special method only to such cases near singularity. Such an approach is straightforward, but it can lead to instabilities. A small perturbation of one vertex can trigger a special case handling and thus it could switch the mean curvature discretization method, making the process discontinuous.

An elegant solution to this discontinuity problem is to exploit the connection between possible singularity of A^tA and the vertex valence. Assuming that all points $(t_{i,x}, t_{i,y})$ are distinct, a simple consequence of Lemma 1 is that the matrix A^tA can only become singular, if the valence of q_i is 3 or 4.

For all vertices of valence 3 or 4 we increase the data quantity that serves as input for our algorithm. Instead of enlarging the vertex neighborhood—which lacks symmetry if the valence of the neighbor vertices varies largely—we increase the local input data by estimating new vertices p_j between adjacent $q_j \in N(q_i)$. The p_j and q_j then serve as input for the mean curvature discretization as described in Section 6.1, making the problem well posed.

Simply setting $p_j = (q_j + q_{j+1})/2$ would be fast and convenient, but such an approach distorts the resulting mean curvature discretization considerably and should not be used if high quality results have to be generated. A scheme that has proven to be adequate during our numerical experiments is based on the idea to determine p_j by sampling a planar curve with monotone curvature, that interpolates the vertices and normals at q_j and q_{j+1} (Fig. 5(a)). Since it is not obvious what type of spiral (or pair of spirals, if the curve has an inflection point) is most promising and the computation of such an interpolating curve can be expensive, we exploit a nice approximation property of spirals (Marciniak and Putz, 1984):

Given two planar points q_1 and q_2 with tangent vectors $\vec{t}_1$ and $\vec{t}_2$, let s be an arbitrary spiral that satisfies this G^1 interpolation problem. Let c_1 be the circle defined by q_1, q_2 and $\vec{t}_1$ and c_2 be the circle defined by q_1, q_2 and $\vec{t}_2$ (such that $\vec{t}_1$ and $\vec{t}_2$ are circle tangents). Then the spiral s can be approximated by the area enclosed by the circular arcs of c_1 and c_2 between q_1 and q_2, if the tangent angle between $\vec{t}_1$ and $\vec{t}_2$ does not change exceedingly (Fig. 5(b)). This argument is also reasonable for planar curves with monotone curvature and an inflection point (Fig. 5(c)).



(a) (b) (c)

Fig. 5. (a) At vertices q_i with valence 3 or 4, between all $q_j, q_{j+1} \in N(q_i)$ that share a common edge, new vertices p_j are introduced. The p_j lie on a plane E_j determined by q_j, q_{j+1} and the vector $\vec{n}_{j1} + \vec{n}_{j2}$, where $\vec{n}_{j1}$ and $\vec{n}_{j2}$ are the triangle normals of the faces adjacent to the edge $q_j q_{j+1}$. (b) Approximation of a spiral (planar curve with monotone nonzero curvature) by the area enclosed by two arcs. (c) This approximation also produces reasonable results for planar curves with monotone curvature and an inflection point (denoted as I).

The tangent vectors $\vec{t}_1$ and $\vec{t}_2$ are computed by intersecting the tangent planes at the vertices q_j and q_{j+1} (defined by their normal vectors) with the plane E_j (Fig. 5(a)), where we choose in each case the direction that has the smaller angle to the vector $q_{j+1} - q_j$. After intersecting the enclosed area with the perpendicular bisector of q_1 and q_2, we set $p_j = p$ to be the center point of the intersection interval. Each circle has two intersection points with the perpendicular bisector, here we choose only that intersection point that is closer to the line defined by q_1 and q_2.

7. Construction algorithm

In this section we finally present the construction algorithm for a mesh M_S that is a discrete solution of Eq. (4), i.e., it satisfies

$$\Delta_B H(q_i) = 0 \quad \forall q_i \in V_I(M_S) \tag{12}$$

plus an additional inner fairness criterion. The input data for our algorithm consists of vertices and unit normals that form the G^1 boundary condition and an initial mesh M^0 that interpolates the boundary vertices. The idea of the construction algorithm is to create a mesh sequence $M^k,\ k = 0, 1, 2, \ldots,$ by iteratively updating the vertices, until the outer and inner fairness conditions are sufficiently satisfied. Although in our implementation the mesh connectivity varies during the construction algorithm (see Section 7.4), let us first assume that only the position of the vertices changes, while the connectivity remains constant.

One possible strategy to transform the initial mesh M^0 into the solution M_S would be to use motion by intrinsic laplacian of curvature (Chopp and Sethian, 1999), exploiting the fact that (4) describes the equilibrium of that flow. Here, for fine meshes very small time steps are needed to achieve stability and it is hard to decide what concrete time step values should be used in an application.

To avoid such problems, in this paper we do not use curvature flow by the laplacian of curvature directly, instead we adapt the ideas presented in (Schneider and Kobbelt, 1999) to our case. In that paper an algorithm for the fast construction of discrete planar clothoid splines—the planar analogon to our problem—was presented. The key idea of that algorithm was to factorize the fourth order problem into two problems of second order. In combination with a multigrid scheme and an iteration step that alternates between local and global update strategies, an efficient and reliable algorithm for the discretization of planar clothoid splines was presented.

7.1. Factorization

Instead of solving a fourth order problem directly, we factorize it into two second order problems which are solved sequentially. The factorization idea is inspired by the following observation: Given a fixed Laplace–Beltrami operator and fixed mean curvature values at the boundary vertices $V_B(M^k)$ of a mesh M^k, (12) can be interpreted as a Dirichlet problem for the H_i, where the unknown scalar mean curvature values at the inner vertices are determined by a nonsingular linear system with a symmetric and positive definite matrix S whose coefficients are defined in (6) and (7). Solving the resulting nonsingular linear system yields scalar values $\widetilde{H}_i$ at all inner vertices $q_i \in V_I(M^k)$, that represent a discrete harmonic function. The idea is now to use this calculated scalar values $\widetilde{H}_i$ to update each inner vertex q_i such that $H(q_i^{k+1}) = \widetilde{H}_i$, which is again a second order problem. Expressed in two formulas, this factorization of $M^k \to M^{k+1}$ becomes

$$\left.\begin{array}{lll} \text{(I)} & \Delta_B \widetilde{H}_i & = 0 \\ \text{(II)} & H(q_i^{k+1}) & = \widetilde{H}_i \end{array}\right\} \quad \forall q_i^k \in V_I\big(M^k\big).$$

We determine the Laplace–Beltrami operator and the boundary mean curvature values by calculating the according values of the current mesh M^k. In practice, it is not necessary to solve the Dirichlet problem exactly. When we have determined the linear system, we apply some iteration steps of an iterative linear solver, using the current mean curvature values as starting values. So one step to update a mesh $M^k \to M^{k+1}$ becomes:

- All mean curvature values $H(q_i^k)$ at all vertices of the current mesh M^k are calculated, the mean curvature discretization technique depends on the vertex valence as described in Section 6. Also the discrete Laplace–Beltrami operator is determined, that means we calculate the cotangent weights (6) and (7) for every inner vertex.
- The mean curvature values at the boundary and the discretized Laplace–Beltrami operator determine a Dirichlet problem that can be formulated as a linear system for the interior mean curvature values. Use the mean curvature values at the interior vertices as initial values and iterate the linear system n times using an iterative linear system solver.
- The second step results in improved scalar values $\widetilde{H}_i$ for the interior vertices and this scalar values are used to update all $q_i^k \in V_I(M^k)$. The update is done in a Gauss–Seidel like manner, that means adjacent vertices that have already been updated are used with their new position.

While discrete harmonic functions do not always have to share all the mathematical properties of their continuous counterparts, e.g. the convex hull property (Pinkall and Polthier, 1993), they will nevertheless approximate continuous harmonic functions. This means our scalar values $\widetilde{H}_i$ will approximate a function that does not have local extrema and whose maximal values occur at the boundary, so the $\widetilde{H}_i$ will behave well and can be approximately bounded by the current mean curvature values at the boundary vertices.

As in (Schneider and Kobbelt, 1999), we noticed that we can improve the convergence rate, if we mix high and low frequency smoothing steps. For that purpose, in our implementation we used Gauss–Seidel and conjugate gradient (Golub and Van Loan, 1989) iterations. Both schemes enable especially efficient coding if applied on meshes and both would converge for $n \to \infty$, since the matrix S is positive definite. We alternate between a mesh update $M^k \to M^{k+1}$ based on Gauss–Seidel with $n = 1$ and a conjugate gradient update where n is chosen larger. As a noninteractive criterion to terminate the fairing algorithm, we can iterate until $\|\Delta_B H\| < \varepsilon$ at every vertex for a prescribed ε.

7.2. Vertex updates

The inner vertices are updated $q_i^k \to q_i^{k+1}$, using the scalar mean curvature values $\widetilde{H}_i$ resulting from the iterative solver described above. The aim of the update step is to produce a new mesh M^{k+1} whose mean curvature values $H(q_i^{k+1})$ at the vertices $q_i \in V_I(M^{k+1})$ are closer to the calculated $\widetilde{H}_i$ values than those of the previous mesh M^k. In order to be able to separate between inner and outer fairness, we only allow the vertex to move along the surface normal vector. This means we search a scalar value t such that

$$H\big(q_i^{k+1}\big) = \widetilde{H}_i \quad \text{with } q_i^{k+1} = q_i^k + t\vec{n}. \tag{13}$$

If we take a look at the mean curvature discretization algorithm presented in Section 6.1, we find that the matrix A does not change if we move q_i along the normal vector. So only the right side of Eq. (11) is influenced by such a motion and for the normal curvatures (8) we obtain

$$\tilde{\kappa}_j = 2\frac{\langle q_j - q_i - t\vec{n}|\vec{n}\rangle}{\langle q_j - q_i - t\vec{n}|q_j - q_i - t\vec{n}\rangle}.$$

This normal curvature discretization is nonlinear in t, but since the position of q_i will not change much during the update step, we assume the distance between q_i and q_j to remain constant. With this linearization technique we get

$$\tilde{\kappa}_j \approx 2\frac{\langle q_j - q_i|\vec{n}\rangle}{\langle q_j - q_i|q_j - q_i\rangle} - 2t\frac{1}{\langle q_j - q_i|q_j - q_i\rangle}.$$

Using this assumption, the discretization of $H(q_i^{k+1})$ determined by (10) and (11), becomes a linear function in t. Solving this linear equation for t, we finally update $q_i^{k+1} = q_i^k + t\vec{n}$ which approximately solves (13).

7.3. Inner fairness condition

To control how the vertices are distributed on the surface of the solution, for every inner vertex $q_i \in V_I$ we assign a generalized discrete laplacian Δ that determines the local

parameterization of the surface. Assigning a scalar weight λ_{ij} to every $q_j \in N(q_i)$ with the constraint $\sum_j \lambda_{ij} = 1$, the discrete laplacian Δ is defined as

$$\Delta(q_i) = -q_i + \sum_{q_j \in N(q_i)} \lambda_{ij} q_j.$$

A mesh M is said to satisfy the inner fairness condition, if all $\Delta(q_i)$ have vanishing tangential components, so that there are scalar values t_i such that $\Delta(q_i) = t_i \vec{n}_i$ for all $q_i \in V_I(M)$.

We integrate the inner fairness condition into the construction algorithm by including it into the update step described in Section 7.2. Instead of updating $q_i^k \to q_i^{k+1}$ along the line $q_i^k + t\vec{n}_i$, we update along the line $\tilde{q}_i^k + t\vec{n}_i$, where $\tilde{q}_i^k$ is the projection of $\sum \lambda_{ij} q_j$ onto the plane defined by q_i^k and $\vec{n}_i$

$$\tilde{q}_i^k = \sum_{q_j \in N(q_i)} \lambda_{ij} q_j + \langle \Delta(q_i) | \vec{n}_i \rangle \vec{n}_i.$$

In our examples we used two different generalized laplacians Δ. If we assume local uniform parameterization, we arrive at a laplacian with weights $\lambda_{ij} = \frac{1}{m}$, where m is the valence of the vertex q_i. This laplacian leads to meshes that have a regular vertex distribution on the surface. However, in some important cases such a kind of mesh parameterization is not wanted. For example in multiresolution mesh modeling (Kobbelt et al., 1998b; Guskov et al., 1999), one would like to have a faired mesh that minimizes local distortions when compared to the original mesh M^0.

For meshes it is usually not possible to get a map that is conformal in the sense that angles are preserved, but we can use the fact that the discretization of $\Delta_B f = 0$ can be interpreted as a discrete harmonic map (Pinkall and Polthier, 1993; Duchamp et al., 1997) and thus approximates a continuous conformal map. So to minimize local distortions, we use the weights resulting from the discretization of the Laplace–Beltrami operator of the original mesh M^0. The weights λ_{ij} at the vertex q_i are then given by

$$\lambda_{ij} = \frac{\cot\alpha_j + \cot\beta_j}{\sum_{q_j \in N(q_i)} (\cot\alpha_j + \cot\beta_j)},$$

where α_j and β_j are the triangle angles as shown in Fig. 3.

Other promising parameterizations are, e.g., the *weighted least square* and the *shape-preserving* parameterizations presented by Floater (1997). Compared to the discrete conformal parameterization they have the advantage to be based on convex combinations only.

7.4. Multigrid approach

It is well known, that the convergence of mesh fairing algorithms can be dramatically accelerated, if multigrid techniques are integrated into the construction process (Kobbelt, 1997; Guskov et al., 1999). In our implementation we followed this idea. An overview of the algorithm is shown in Fig. 6. The necessary hierarchy levels are constructed using the progressive mesh approach (Hoppe, 1996) with half-edge collapses (Kobbelt et al., 1998a).

- Create a progressive mesh representation of the given mesh and subdivide the vertex splits into hierarchies
- Solve the fairing problem on the base mesh
  - Create a smooth initial mesh using an algorithm that can handle noisy meshes
  - Solve intrinsic fairing problem on the base mesh
- For each hierarchy level
  - Add vertices of the progressive mesh representation until the next hierarchy level is reached as described in Section 7.4
  - Smooth the complete mesh on the current hierarchy level. When determining the estimated new mean curvature distribution, alternate between local and global iterative schemes
    - local: Determine the $\widetilde{H}_i$ values using one Gauss–Seidel iteration and update the vertices
    - global: Determine the $\widetilde{H}_i$ values using n conjugate gradient iterations (with diagonal preconditioner) and update the vertices

Fig. 6. The multigrid fairing algorithm.

However, instead of reducing a mesh while trying to keep the details, we are more interested in creating a mesh whose smallest edge length is maximal while avoiding distorted triangles (long triangles with small inner circle). The number of hierarchy levels can be specified by the user. A reasonable strategy to fix the number of vertices per hierarchy is exponential growth.

Our multigrid algorithm exploits the fact that a coarse mesh already approximates the shape of the smooth surface that is implicitly defined by the PDE, increasing the mesh size mainly improves the smoothness of the approximation (Figs. 2 and 8). Therefore, we start with the construction of a discrete solution on the coarsest level of the progressive mesh representation and then each solution on a coarse level serves as starting point for the iteration algorithm on the next finer hierarchy level. Between two hierarchy levels we need a prolongation operator that introduces new vertices using the vertex split information of the progressive mesh. When adding a new vertex q_i, we have to take care that the outer fairness is not destroyed at that position. This is achieved in three steps, where the first two steps are similar to the prolongation operator used in (Guskov et al., 1999):

- First we update the mesh topology and introduce q_i at the position given by its inner fairness criterion

$$q_i = \sum_{q_j \in N(q_i)} \lambda_{ij} q_j .$$

- In some cases the first step is not enough to avoid triangle distortions, therefore in the second step we further update the complete 1-ring of q_i. This means we solve the local linear problem $\Delta q_l = 0$ for all $q_l \in D(q_i)$.
- Since the second step disturbs the outer fairness, we finally solve $\Delta_B H_l = 0$ for all $q_l \in D(q_i)$ by applying the construction algorithm locally on the 1-disk $D(q_i)$.

Our mean curvature discretization (Section 6) as well as the update step (Section 7.2) assume that the mesh is not a noisy surface. Therefore, before starting the multigrid algorithm we first construct at the coarsest level the linear solution of the problem $\Delta^2 f = 0$ using the laplacian defined by the chosen inner fairness criterion (see Fig. 1(c)). We further

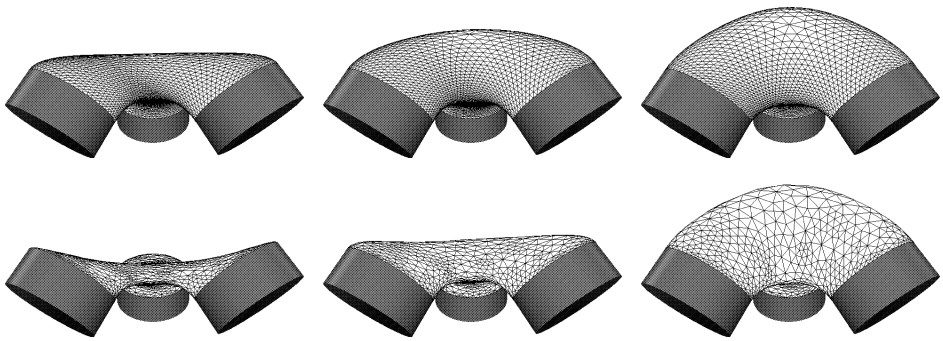


Fig. 7. Comparison of the results from a linear bilaplacian (left), our improved bilaplacian (middle) and our intrinsic method. In the upper row the mesh has a regular subdivision connectivity structure and in the lower row it is sparse and has a very irregular connectivity.

Table 1
Running times of the intrinsic fairing algorithm

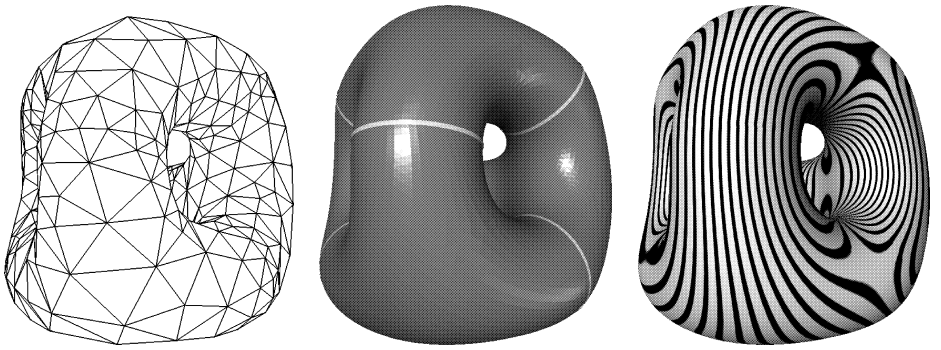
Dataset	Vertices complete mesh	Triangles complete mesh	Vertices base mesh	Time sec.
Cylinder blend	920	1746	140	1.9
Bust face	2926	5720	256	12
Human ear	4665	9220	248	21
Tetra thing	13308	26624	308	57

developed an improved bilaplacian mesh fairing algorithm (Schneider et al., 2000) that produces an initial mesh with superior quality than those resulting from linear schemes. For sparse meshes with very irregular structure, this scheme is much better suited to create an initial mesh (Fig. 7). Later at each hierarchy level our mesh is already presmoothed by the multigrid fairing concept.

7.5. Examples and remarks

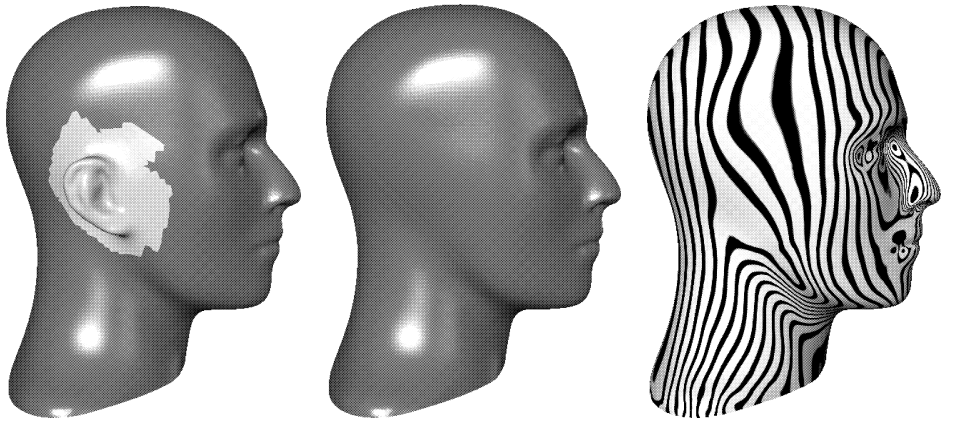
Because of the importance of the discrete mean curvature at the boundary vertices for the construction algorithm and the fact that at such points we only have one-sided vertex information, in our examples we calculated the mean curvature at every boundary vertex using the special case discretization as for inner vertices of valence 3 or 4. For coarse meshes, we also used the special case handling for interior vertices of higher valence.

Instead of updating $q_i^{k+1} = q_i^k + st\vec{n}$ with $s = 1.0$ as described in Section 7.2, in practice s has to lie in the range $0 < s < 1.0$. While $s = 1.0$ usually will work well, we noticed that in some rare constellations it is possible that vertices can alternate between two positions. All of our examples presented in this paper were constructed using $s = 0.9$.



(a) (b) (c)

Fig. 8. 6 circles are used to define a tetra thing. (a) and (b) show the solution of a mesh with 500 respectively 13308 vertices. (c) Shows the reflection lines of the mesh (b). Due to the symmetric constellation the final smooth solution would be G^2 continuous.



(a) (b) (c)

Fig. 9. Feature removal of a mesh with user defined boundary curve.

The conformal inner fairness is only hardly more time consuming to construct than the uniform parameterization, but requires much more memory. Here the original mesh has to be reconstructed from the progressive mesh representation parallel to the faired mesh to be able to update the necessary coefficients of the laplacian—which also have to be stored in memory—at every hierarchy level.

The lines of reflection in our example pictures indicate G^1 continuity at the boundary (continuous lines) and at least G^2 continuity in the interior (smooth lines). As can be seen in Examples 9 and 10 the boundary does not have to be smooth. In such cases the fixed unit normal information at the boundary is derived by averaging the adjacent triangle normal vectors before the construction.

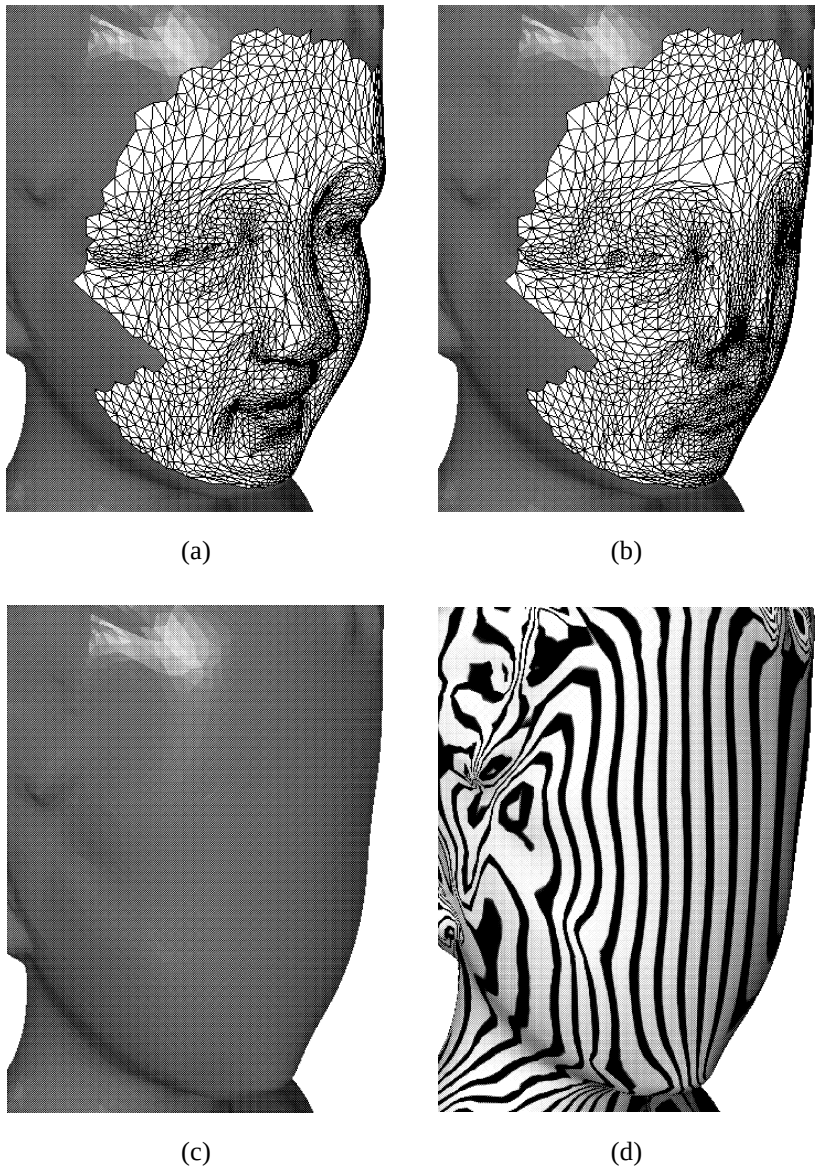


Fig. 10. Here we smoothed the face of a bust model shown in (a). The triangulation of the solution in this example is discrete conformal to the initial mesh.

The concept behind our construction algorithm works best for $\Delta_B H = 0$, but it is not restricted to that. We also constructed MESs by solving their Euler–Lagrange equation $\Delta_B H = -2H(H^2 - K)$ and surfaces satisfying $\Delta_B H = \text{const}$. However, these inhomogen problems are more costly to solve and a stable construction algorithm is more involved.

In this paper we assumed that the final mesh should have a prescribed vertex connectivity. If this is not the case, flipping of edges should be enabled to avoid long triangles, new vertices may be introduced where necessary or removed where the vertex density becomes too high (see (Hsu et al., 1992; Welch and Witkin, 1994)).

All shaded pictures are flat shaded to enhance the faceting effect. The construction algorithm was implemented in Java 1.3 and Java 3D for Windows NT running on a Pentium III with 500 MHz. We noticed that our fairing algorithm is very fast for a fairing approach based on intrinsics. The running times for the examples shown in this paper are presented in Table 1, they do not include the time needed to build the progressive mesh representation. An optimized C implementation would improve these timings without doubt.

8. Conclusion and future work

The technique we presented in this paper shows that it is possible to use higher order intrinsic PDEs in modeling based on irregular meshes. Since the surface is completely defined by G^1 boundary conditions and a PDE, the designer does not have to take surface parameterization or the actual mesh representation into account and can freely distribute the vertices on the surface.

One of our future plans is to integrate the presented technique in a multiresolution mesh modeling tool as those described in (Kobbelt et al., 1998b). Although the new construction algorithm is obviously slower than the simple linear fairing algorithm used there, based on the results we got from our current implementation, it seems possible to achieve interactive frame rates with a decent mesh complexity. In combination with an inner fairness criterion that enables local shape preservation, this would lead to a much simpler detail coding. Because of the intrinsic surface definition, this would also simplify a mesh modeling that does not fix the connectivity or the mesh size during the modifications.

We will also explore other curvature and update algorithms and study different intrinsic PDEs to introduce new design freedoms. Furthermore, it will be necessary to integrate surface constraints in the interior of the surface such as interpolation of points, which might require PDEs of higher order.

Acknowledgements

We thank Jens Vorsatz and Mario Botsch for their substantial contributions to the creation of our examples and also Kolja Kähler for providing the progressive mesh algorithm.

References

Bloor, M.I.G., Wilson, M.J., 1990. Using partial differential equations to generate free-form surfaces. Computer-Aided Design 22, 202–212.

Brakke, K.A., 1992. The surface evolver. Experimental Mathematics 1 (2), 141–165.

Burchard, H.G., Ayers, J.A., Frey, W.H., Sapidis, N.S., 1994. Approximation with aesthetic constraints, in: Sapidis, N.S. (Ed.), Designing Fair Curves and Surfaces, SIAM, Philadelphia.

do Carmo, M.P., 1993. Differential Geometry of Curves and Surfaces. Prentice Hall, Englewood Cliffs, NJ.

Chopp, D.L., Sethian, J.A., 1999. Motion by intrinsic laplacian of curvature. Interfaces and Free Boundaries 1, 1–18.

Desbrun, M., Meyer, M., Schröder, P., Barr, A.H., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow, in: SIGGRAPH '99 Conference Proceedings, pp. 317–324.

Desbrun, M., Meyer, M., Schröder, P., Barr, A.H., 2000. Discrete differential-geometry operators in nD, submitted for publication.

Duchamp, T., Certain, A., DeRose, T., Stuetzle, W., 1997. Hierarchical computation of PL harmonic embeddings. Technical Report, University of Washington.

Floater, M.S., 1997. Parametrization and smooth approximation of surface triangulations. Computer Aided Geometric Design 14, 231–250.

Golub, G.H., Van Loan, C.F., 1989. Matrix Computations. Johns Hopkins University Press, Baltimore.

Guskov, I., Sweldens, W., Schröder, P., 1999. Multiresolution signal processing for meshes, in: SIGGRAPH '99 Conference Proceedings, pp. 325–334.

Hoppe, H., 1996. Progressive meshes, in: SIGGRAPH '96 Conference Proceedings, pp. 99–108.

Hsu, L., Kusner, R., Sullivan, J., 1992. Minimizing the squared mean curvature integral for surfaces in space forms. Experimental Mathematics 1 (3), 191–207.

Kobbelt, L., 1997. Discrete fairing, in: Proceedings 7th IMA Conference on the Mathematics of Surfaces, pp. 101–131.

Kobbelt, L., Campagna, S., Seidel, H.-P., 1998a. A general framework for mesh decimation, in: Proceedings of the Graphics Interface Conference, pp. 43–50.

Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.-P., 1998b. Interactive multi-resolution modeling on arbitrary meshes, in: SIGGRAPH '98 Conference Proceedings, pp. 105–114.

Marciniak, K., Putz, B., 1984. Approximation of spirals by piecewise curves of fewest circular arc segments. Computer-Aided Design 16, 87–90.

Moreton, H.P., Séquin, C.H., 1992. Functional optimization for fair surface design, in: SIGGRAPH '92 Conference Proceedings, pp. 167–176.

Ohtake, Y., Belyaev, A.G., Bogaevski, I.A., 2000. Polyhedral surface smoothing with simultaneous mesh regularization, in: Geometric Modeling and Processing 2000 Proceedings, pp. 229–237.

Pinkall, U., Polthier, K., 1993. Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 2 (1), 15–36.

Schneider, R., Kobbelt, L., 1999. Discrete fairing of curves and surfaces based on linear curvature distribution, in: Curve and Surface Design—Saint-Malo '99 Proceedings, pp. 371–380.

Schneider, R., Kobbelt, L., 2000. Generating fair meshes with G^1 boundary conditions, in: Geometric Modeling and Processing 2000 Proceedings, pp. 251–261.

Schneider, R., Kobbelt, L., Seidel, H.-P., 2000. Improved bilaplacian mesh fairing. Submitted for publication.

Taubin, G., 1995a. A signal processing approach to fair surface design, in: SIGGRAPH '95 Conference Proceedings, pp. 351–358.

Welch, W., Witkin, A., 1994. Free-form shape design using triangulated surfaces, in: SIGGRAPH '94 Conference Proceedings, pp. 247–256.