Anisotropic Diffusion of Surfaces and Functions on Surfaces

CHANDRAJIT L. BAJAJ
University of Texas, Austin
GUOLIANG XU
Chinese Academy of Sciences, Beijing

We present a unified anisotropic geometric diffusion PDE model for smoothing (fairing) out noise both in triangulated two-manifold surface meshes in $\mathbb{R}^3$ and functions defined on these surface meshes, while enhancing curve features on both by careful choice of an anisotropic diffusion tensor. We combine the C^1 limit representation of Loop's subdivision for triangular surface meshes and vector functions on the surface mesh with the established diffusion model to arrive at a discretized version of the diffusion problem in the spatial direction. The time direction discretization then leads to a sparse linear system of equations. Iteratively solving the sparse linear system yields a sequence of faired (smoothed) meshes as well as faired functions.

Categories and Subject Descriptors: G.1.8 [**Numerical Analysis**]: Partial Differential Equations—*finite element methods*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*splines*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*color, shading, shadowing, and texture*; I.4.3 [**Image Processing and Computer Vision**]: Enhancement

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Surface function diffusion; Loop's subdivision; Riemannian manifold, texture mapping, noise reduction

1. INTRODUCTION

Problem Considered

Given are a discretized triangular surface mesh $G_d \subset \mathbb{R}^3$ (geometric information) and a discretized function-vector $F_d \subset \mathbb{R}^{\kappa-3}$. Each of the function-vector values is attached to one and only one vertex of the surface mesh. We assume that both the geometric and surface function information suffer from noise. Our primary goal is to smooth out the noise and to obtain faired geometry as well as faired surface function data at different scales. Our secondary goal is to construct continuous (nondiscretized) representations for the smoothed geometry and surface function data. In this article, we use the terms fairing and smoothing interchangeably.

C. L. Bajaj was supported in part by NSF grants ACI-9982297, KDI-DMS-9873326, and NPACI-UCSD 1018140 (Interactive Environments & Engineering), and SANDIA/LLNL BD 4485-MOID. G. Xu was supported in part by NSF 10241004 of China and National Innovation Fund 1770900, Chinese Academy of Sciences.

Authors' addresses: C. L. Bajaj, Department of Computer Science, University of Texas, Taylor Hall 2.124, Austin, TX 78712-1188; email: bajaj@cs.utexas.edu; G. Xu, Institute of Computational Mathematics, Chinese Academy of Sciences, Beijing, China; email: xuguo@lsec.cc.ac.cn.

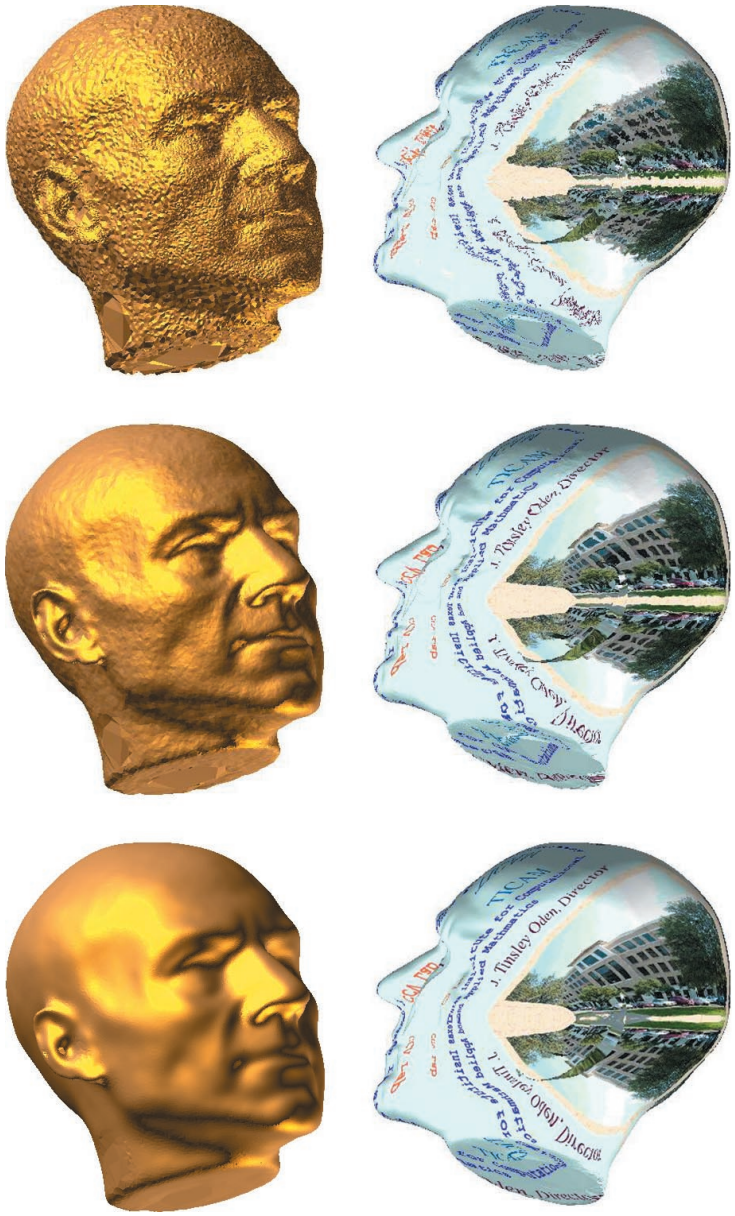

Fig. 1. First column: Fairing the geometry of the head model of Picard (146,036 triangles). The second and third figures in this column are the meshes after one and four steps of fairing. Second column: Fairing texture coordinates while the geometry is fixed. The second and third figures of this are the fairing results after one and four iterations. In all the examples in this article, the timestep τ is 0.001.

Motivation

Quite often, discretized surfaces under investigation suffer from noise or errors in geometry (see Figure 1). For surfaces and attribute functions that come from the reconstruction of physical objects, the noise comes from the sampling error of the imaging equipment, such as CT, MRI, ultrasound, or 3D

laser scanners. If the surface and function on surface (e.g., air velocity on an airfoil) are the result of numerical computation (e.g., finite element simulations), the errors come from the numerical sensitivity of the algorithm or model discretization. The use of lossy compression is prevalent in streaming geometry and textures for Internet gaming and eCommerce visualization applications. The lossy compressed geometry and texture data when decoded often suffer from noise caused by the inaccuracy in spatial distribution of the mesh density (topology) and the quantization of the numerical vertex coordinate data.

The errors of the geometric and surface function data may often be coherent. For example, the sound pressure distribution function resulting from the numerical solution of the Helmholtz equation over a surface domain is very sensitive to the perturbation in the geometry, especially for high frequencies. In another situation, the function errors could also cause geometry errors. For example, in the case of surface extraction from volumetric MRI and surface coregistration with functional-MRI volumetric data, the errors of the function could result in direct errors of the extracted surfaces. In these cases, it might be rational to combine the geometry and surface function data, and to consider the smoothing problem uniformly. Another point of view is to look at the surface function data as graphs. If we consider a greyscale image $I(x, y)$ defined on the xy-plane as a surface in $\mathbb{R}^3$, then the image is given by the graph $(x, y, I(x, y))$. Similarly, if we consider a scalar function $f(x, y, z)$ defined on a surface G as a hypersurface in $\mathbb{R}^4$, then the surface is given by the graph $(x, y, z, f(x, y, z))$ for $(x, y, z) \in G$. Finally, smoothing both the geometry and function data on the surface uniformly reduces the computational costs in the finite element discretization, as both of them use the same stiffness matrix (see Section 4.2 for details). Of course, in other cases, when the surface geometry and function on surface data errors are not coherent, the smoothing could be performed separately.

Previous Work.
The existing approaches for surface fairing can be classified roughly into two categories: optimization and evolution. In the first category, one constructs an optimization problem that minimizes certain objective functions [Greiner 1994; Hoppe et al. 1993; Hubeli and Gross 2000; Moreton and Sequin 1992; Sapidis 1994; Welch and Witkin 1992], such as thin plate energy, membrane energy [Kobbelt et al. 1998], total curvature [Kobbelt et al. 1997; Welch and Witkin 1994], or sum of distances [Mallet 1992]. Using local interpolation or fitting, or replacing differential operators with divided difference operators, the optimization problems are discretized to arrive at finite-dimensional linear or nonlinear systems. Approximate solutions are then obtained by solving the constructed systems.

The main solution idea of the evolution category is borrowed from the solution of the linear heat equation $\partial_t \rho - \Delta\rho = 0$ for equilibrating spatial variation in concentration, where $\Delta := \mathrm{div} \cdot \nabla$ is the Laplace operator. This PDE- (partial differential equation) based evolution technique was originally transplanted to image processing. (See Perona and Malik [1987], Preusser and Rumpf [1999], and Weickert [1998]. In the last-mentioned, 453 relevant references are listed.) This was extended to smoothing or fairing noisy surfaces (see Clarenz et al. [2000], and Desbrun et al. [2000a,c]). For a surface M, the counterpart of the Laplacian Δ is the Laplace–Beltrami operator Δ_M (see do Carmo [1992]). One then obtains the geometric diffusion equation $\partial_t x - \Delta_M x = 0$ for surface point $x(t)$ on the surface $M(t)$.

Taubin [1995] discussed the discretized operator of the Laplacian and related approaches in the context of generalized frequencies on meshes. Kobbelt [1996] considered discrete approximations of the Laplacian in the construction of fair interpolatory subdivision schemes. This work was extended to arbitrary connectivity for purposes of multiresolution interactive editing [Kobbelt et al. 1998]. Desbrun et al. [2000a] used an implicit discretization of geometric diffusion to obtain a strongly stable numerical smoothing scheme. Clarenz et al. [2000] introduced anisotropic geometric diffusion to enhance features while smoothing. All these were based on a discretized surface model. Hence the

first- and second-order derivative information, such as normals, tangents, and curvatures, were estimated using some local averaging or fitting scheme. Computational methods of normals and curvatures for discrete data were carefully studied recently by Desbrun et al. [2000c]. They used the proposed methods to mesh smoothing and enhancement.

Similar to surface diffusion using the Laplacian, another class of PDE-based methods called *flow surface techniques* have been developed that simulate different kinds of flows on surfaces (see Westermann et al. [2000] for references) using the equation $\partial_t x - v(x,t) = 0$, where $v(x,t)$ represents the instantaneous stationary velocity field.

Recently, level-set methods were also used in surface fairing and surface reconstruction (see Bertalmio et al. [2000a,b], Chopp and Sethian [1999], Osher and Fedkiw [2000], Whitaker and Breen [1998], and Zhao et al. [2001, 2000]). In these methods, surfaces are formulated as isosurfaces (level surfaces) of 3D functions, which are usually defined from the signed distance over Cartesian grids of a volume. An evolution PDE on the volume governs the behavior of the level surface. These level-set methods have several attractive features including ease of implementation, arbitrary topology, and a growing body of theoretical results. Often, fine surface structures are not captured by level sets, although it is possible to use adaptive (see Bansch and Mikula [2001] and Preusser and Rumpf [1999]) and triangulated grids. To reduce the computational complexity, Bertalmio et al. [2000a,b] solve the PDE in a narrow band for deforming vectorial functions on surfaces (with a fixed surface represented by the level surface). In their approach, anisotropic diffusion is also considered.

In 2D image processing, Sochen, Kimmel, Malladi (see Kimmel et al. [1998], Kimmel and Sochen [1999], and Sochen et al. [1998]), and Yezzi [1998] treat images as high-dimensional surfaces and process them based on projected curvature motion flows. A similar treatment was adopted by Desbrun et al. [2000b] for denoising bivariate data embedded in high-dimensional spaces while preserving edges. Curvature flows were also used in Sethian [1999, Chapter 16] for image enhancement and noise removal.

For fairing functions on surfaces, Kimmel [1997] used geodesic curvature flow to smooth images painted on a surface. We should point out that many of the above surface fairing methods can be extended to the problem of fairing functions on surfaces if each component of the vector function is independently smoothed. For example, the signal processing approach for meshes proposed in Guskov et al. [1999] has been used to smooth the coordinates of texture maps. In this article, we provide a new approach when vector-function data on a surface are treated simultaneously, both together and independent of the surface geometry.

The aim of *anisotropic* diffusion is to smooth the two-manifold in a certain direction and enhance sharp features in another direction. There is plentiful use in 2D image processing [Action 1998; Harr Romeny 1994; Kawohl and Kutev 1998; Perona and Malik 1987; Weickert 1996, 1998]. We review a relevant few that relate to vector-valued data. Sapiro and Ringach [1996] determine the directions of maximal and minimal rate of change of vector-valued functions by eigenvectors and eigenvalues of the first fundamental form in the given image metric. Weickert [1997] uses a structure tensor for anisotropic diffusion. The structure tensor is constructed as the sum of structure tensors (second-moment matrix) of each channel of the vector-valued function. In Tang et al. [2000b], the authors propose a framework to isotropically and anisotropically diffuse directional data, using a harmonic map. Different from the general vector-valued functional data mentioned above, the directional data satisfy a unit norm constraint. The harmonic map allows both the domain and directional data to be nonflat. Hence the approach could be used for denoising data on surfaces in 3D. This approach is also used to enhance color images [Tang et al. 2000a]. Similar to Tang et al.'s direction diffusion, but developed independently, is Chan and Shen's [1999] research. They develop both models and algorithms for denoising and restoration of nonflat image features that live on Riemannian manifolds. For surface smoothing, Desbrun et al. [2000c] use weighted mean curvature flow to achieve the effect of anisotropic

diffusion. The weighted mean curvature flow penalizes the vertices that have a large ratio between their two principal curvatures. Clarenz et al. [2000] use a diffusion tensor defined from the principal directions and principal curvatures of the deformed surface. Our anisotropic diffusion tensor generalizes the work of Clarenz et al. to higher dimensions.

Our Approach and Contributions

(*a*) *Establishing a Unified Diffusion Model*. In this article, we call a triangular surface mesh with function values on each of the vertices of the mesh, simply an *attributed triangular mesh*. We treat three-dimensional discrete surface data and $(\kappa - 3)$-dimensional function data on the surface as a discretized version of a two-dimensional Riemannian manifold embedded in $\mathbb{R}^{\kappa}$. We establish a PDE diffusion model for such a manifold. Although the derivation of the model involves Riemannian geometry, the final outcome is simple and easy to understand. An alternative formulation was taken in Sochen et al. [1998] for color image processing. Our formulation is more straightforward and simpler.

(*b*) *Discretizing the Diffusion Model in a Smooth Function Space*. We combine the limit function representation of Loop's subdivision for triangular meshes with our diffusion model to arrive at a discretized version of the diffusion problem. The input attributed triangular mesh serves as the control mesh of Loop's subdivision. Solving the discretized problem, a sequence of smoothed attributed triangular meshes as well as smoothed functions is obtained. What makes our discretization distinct from previous work is that we diffuse *globally smooth functions* instead of *discrete functions*. Working with smooth (higher-order) function models of finite dimension (instead of linear elements), related quantities, such as gradients, tangents, normals, and curvatures, can be computed exactly and directly at all timesteps. This framework allows us to trade off accuracy and speed with computational complexity.

(*c*) *Anisotropic Diffusion*. We construct an anisotropic diffusion tensor in the diffusion model which makes the diffusion process have the effect of enhancing sharp features while filtering out noise. If $k = 3$, this diffusion tensor reduces to the one given in Clarenz et al. [2000]. Figure 8 shows the difference between applying and not applying an anisotropic diffusion tensor.

The function on a surface defined by Loop's subdivision is in a finite-dimensional space. The basis functions of this space have compact support (within 2-rings of the vertices). This support is bigger than the support (within 1-ring of the vertices) of hat basis functions that are used for the discrete surface model. Such a difference in the size of support of basis functions makes our evolution more efficient than those previously reported, due to the increased bandwidth of affected frequencies. Compared to prior approaches, the reduction speed of high-frequency noise in our iterative solution approach is not overly drastic (leading to overfairing), and the reduction speed of lower-frequency noise is not that slow (leading to underfairing). Hence, the bandwidth of affected frequencies is much wider. Figure 12 provides an example to illustrate this difference. The figures are the fairing results of a noisy input mesh (Figure 10) and one and three fairing steps (timestep 0.001) using linear finite element implementation (the first row) and Loop's finite element implementation (the second row) with identity diffusion tensors. The figures of the first row smooth out more detailed features (see the ears, eyes, lips, and nose) than the figures of the second row, and at the same time the large-scale features (see the head) of the top are less smooth than that of the bottom. Figure 2 shows another example that exhibits the same features of the two approaches. It should be pointed out that the larger support of basis functions leads to more nonzero (five times more on average) elements in the stiffness matrix of the finite element discretization. This implies more computations are required in both forming the matrix and solving the linear system. However, test results show that the numerical condition of the discretized linear system of our higher-order approach is often better than that of the linear element approach, yielding faster convergence rates. For the example mentioned above (Figure 12), our approach needs 23, 18, 16, and 17 iterations

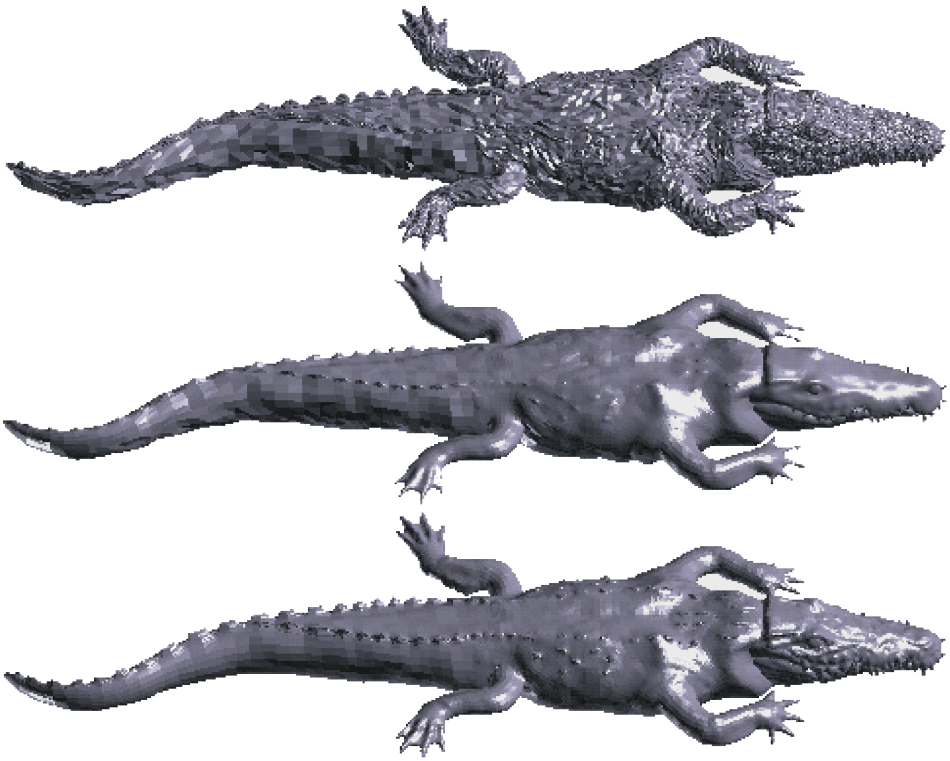

Fig. 2. The top figure shows the initial geometry mesh. The second and the third figures show the faired meshes after two fairing iterations with timestep $\tau = 0.001$ (isotropic diffusion) and using the linear element approach and our high-order smooth element approach, respectively.

for solving the linear systems by Gauss–Seidel methods for the timesteps $1, \ldots, 4$, within the L_∞ error $9 * 10^{-6}$. The linear element approach needs $57, 67, 73$, and 77 iterations, respectively. This behavior is very explainable. Because the support of the basis functions of the linear element is small, tiny triangles will cause very small elements in the matrix of the discretized linear system, which worsens the numerical condition of the system.

Very recently, Arden [2001] in his thesis reported on the approximation power of Loop's subdivision limit functions. His results show that Loop's subdivision limit functions have one order higher approximation power than linear elements. This provides at least one quantifiable measure of their superiority over linear elements.

The evolution process produces not only a sequence of attributed triangular meshes at different timesteps, but also a sequence of smooth functions. By sampling these smooth functions, new attributed triangular meshes at a resolution higher than that of the original mesh can be produced. Furthermore, gradient and curvature at any point can be easily computed.

2. THE DIFFUSION MODEL

The diffusion model that we use is a generalization of the heat equation $\partial_t \rho - \Delta \rho = 0$ in Euclidean space to a two-dimensional manifold embedded in $\mathbb{R}^\kappa$. Such a generalization to surface in 3D has been given by Clarenz et al. [2000]. The natural extension to a two-dimensional manifold embedded in $\mathbb{R}^\kappa$ is what we now present. First, we establish the diffusion model for continuous geometry $G \subset \mathbb{R}^3$ and

continuous surface functions $F \subset \mathbb{R}^{\kappa-3}$. The discretization of the continuous model is discussed in Section 4. Suppose we are given $\kappa - 3$ ($\kappa \geq 3$) functions $f(x) = (f_1(x), f_2(x), \ldots, f_{\kappa-3}(x)) \in F$, $x \in G$. We assume that surface G is a two-dimensional manifold embedded in $\mathbb{R}^3$. We combine the geometric position x and function $f(x)$ to form a κ-dimensional vector $(x, f(x))$. We use M to indicate the graph $\{(x, f(x)) \in \mathbb{R}^\kappa : x \in G\}$. Therefore, we may consider M as a two-dimensional manifold embedded in $\mathbb{R}^\kappa$. Working with such a manifold for establishing the diffusion model requires proper extensions of expressions, such as tangents, gradients, Laplacian, curvatures, and integrations. Fortunately, several of these are well developed in the field of *Riemannian Geometry* (see do Carmo [1992], Rosenberg [1997], and Willmore [1982]).

Tangent Space of Differential Manifold

Let $M \subset \mathbb{R}^\kappa$ be a two-dimensional manifold, and $\{U_\alpha, x_\alpha\}$ be the differentiable structure. The mapping x_α with $x \in x_\alpha(U_\alpha)$ is called a parameterization of M at x. Denoting the coordinate U_α as (ξ_1, ξ_2), the tangent space T_xM at $x \in M$ is spanned by $\{(\partial/\partial\xi_1), (\partial/\partial\xi_2)\}$. For a given point $x \in x_\alpha(U_\alpha) \subset M$, the tangent vector components $\partial/\partial\xi_1$ and $\partial/\partial\xi_2$ depend upon α, but T_xM does not. The set $TM = \{(x, v);\ x \in M,\ v \in T_xM\}$ is called a tangent bundle.

Riemannian Manifold

To define integration on M, a *Riemannian metric* (inner product) is required. A differentiable manifold with a given Riemannian metric is called a *Riemannian Manifold*. A Riemannian metric $\langle , \rangle_x$ of M is a symmetric, bilinear and positive-definite form on the tangent space T_xM. Since M is a submanifold of Euclidean space $\mathbb{R}^\kappa$, we use the *induced metric*:

$$\langle u, v\rangle_x = u^T v, \quad u, v \in T_xM.$$

Integration

Let f be a function on M, and let $\{\phi_\alpha\}_\alpha$ be a finite partition of unity on M with support $\phi_\alpha \subset U_\alpha$. Then define

$$\int_M f\, dx := \sum_\alpha \int_{U_\alpha} \phi_\alpha\ f(x_\alpha)\sqrt{det(g_{ij})}d\xi_1\, d\xi_2, \tag{1}$$

where

$$g_{ij} = \left\langle \frac{\partial}{\partial\xi_i}, \frac{\partial}{\partial\xi_j} \right\rangle_x.$$

Then we can define the inner product of two functions on M and two vector fields on TM as

$$\begin{aligned} (f, g)_M &= \int_M fg\, dx, \qquad f, g \in C^0(M), \\ (\phi, \psi)_{TM} &= \int_M \langle \phi, \psi\rangle\, dx, \quad \phi, \psi \in TM. \end{aligned}$$

Gradient

Suppose $f \in C^1(M)$. The gradient $\nabla_M f \in T_xM$ of f is defined by the following conditions:

$$t_i^T \nabla_M f = \frac{\partial(f \circ x)}{\partial\xi_i}, \quad i = 1, 2, \tag{2}$$

where $t_i = \partial x/\partial\xi_i$ are the tangent vectors. Note that $\nabla_M f$ is invariant under the surface local reparameterization. From (2), we have

$$\nabla_M f = [\, t_1,\ t_2\,]G^{-1}\left[\ \frac{\partial(f \circ x)}{\partial\xi_1},\ \frac{\partial(f \circ x)}{\partial\xi_2}\ \right]^T, \tag{3}$$

where

$$G^{-1} = \frac{1}{\det G}\begin{bmatrix} g_{22} & -g_{12} \\ -g_{21} & g_{11} \end{bmatrix}, \quad G = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix},$$

and G is known as the first fundamental form.

Divergence
The divergence $\mathrm{div}_M \psi$ for a vector field $\psi \in TM$ is defined as the dual operator of the gradient (see Rosenberg [1997]).

$$(\mathrm{div}_M v, \phi)_M = -(v, \nabla_M \phi)_{TM}, \quad \forall \phi \in C_0^\infty(M), \tag{4}$$

where $C_0^\infty(M)$ is a subspace of $C^\infty(M)$, whose elements have compact support.

Diffusion Model
Using the notation introduced above, we formulate the geometric diffusion model as the nonlinear system of parabolic differential equations:

$$\partial_t x(t) - \Delta_{M(t)} x(t) = 0, \tag{5}$$

where $M(t)$ is the solution manifold at time t, $x(t) = (x_1(t), \ldots, x_\kappa(t))$ is a point on the manifold, and $\Delta_{M(t)} = \mathrm{div} \circ \nabla_{M(t)}$ is known as the Laplace–Beltrami operator on $M(t)$. Because $\Delta_M x = 2H(x)$ (see Willmore [1993, page 151]), Equation (5) could be written as $\partial_t x(t) = 2H(x)$, where $H(x)$ is the mean curvature vector at x. Hence the equation describes the mean curvature motion, whose regularization effect could be seen, for $\kappa = 3$, from the following equations (see Clarenz [2000] and Sapiro [2001]).

$$\frac{d}{dt}\mathrm{Area}(M(t)) = -\int_{M(t)} h^2\, dx, \quad \frac{d}{dt}\mathrm{Volume}(M(t)) = -\int_{M(t)} h dx, \tag{6}$$

where $\mathrm{Area}(M(t))$ and $\mathrm{Volume}(M(t))$ are the area of $M(t)$ and volume enclosed by $M(t)$, respectively, and h is the mean curvature (i.e., $H = hn$, where n is the unit surface normal). However, to be able to enhance sharp features, a *diffusion tensor* D, acting on the gradient $\nabla_{M(t)}$, is introduced. Furthermore, a term $r \in \mathbb{R}^\kappa$ on the right-hand side of the equation, which represents an external force, is imposed. Hence the final model we use is

$$\partial_t x(t) - \mathrm{div}_{M(t)}\big(D \nabla_{M(t)} x(t)\big) = r(x(t)), \tag{7}$$

$$M(0) = M, \tag{8}$$

where the diffusion tensor $D := D(x)$ is a symmetric and positive definite operator from TM to TM. The diffusion tensor $D(x)$ has a significant influence on the shape of the diffused surface and functions on the surface. The details of the discussion for choosing the diffusion tensor are in Section 5. The function r is chosen in the form:

$$r(x(t)) = \omega(x(0) - x(t)), \quad \omega \geq 0. \tag{9}$$

This term is used to approximate the initial mesh in the smoothing process, so that the smoothed surfaces do not evolve too much from the initial surface $M(0)$. ω is a user-specified parameter. If $D(x) = I$, an identity operator, then (7) becomes $\partial_t x(t) = 2H(x) + \omega(x(0) - x(t))$. Hence the equation described is a motion that is decomposed into the mean curvature motion, caused by $2H(x)$ and in the direction of the mean curvature vector, and a motion towards the original surface, caused by $\omega(x(0) - x(t))$ and in the direction of $x(0) - x(t)$. The magnitude of ω determines which portion of the two motions dominates the composite motion. In most of the examples in this article, we choose $\omega = 0$. We do give an example

(see Figure 15) that which uses a nonzero ω and shows the effect of ω on the evolved surface. Using (4), the diffusion problem (7) and (8) can be reformulated as the following variational form.

$$\begin{cases} \text{Find a smooth } x(t) \text{ such that} \\ (\partial_t x(t), \theta)_{M(t)} + \left(D\nabla_{M(t)} x(t), \nabla_{M(t)}\theta\right)_{TM(t)} = (r, \theta)_{M(t)}, \quad \forall \theta \in C_0^\infty(M(t)) \\ M(0) = M. \end{cases} \tag{10}$$

Since $x = (x_1, \ldots, x_\kappa)$ is a vector-valued function, the inner product $(\partial_t x(t), \theta)_{M(t)}$ is understood as $((\partial_t x_1(t), \theta)_{M(t)}, \ldots, (\partial_t x_\kappa(t), \theta)_{M(t)})$.

Other Alternatives of the Diffusion Model

In establishing our diffusion model, we have combined the geometry and physical data on the geometry. This combination is under the assumption that both the geometric and physical data have errors and the two errors are coherent. In practice, this assumption may not always be valid. Considering the two aspects of having errors or not, and whether the errors are coherent or not, we have several possibilities: (a) *both the geometry and physical data have errors and the errors are coherent*; (b) *both the geometry and physical data have errors and the errors are not coherent*; (c) *only the physical data have errors*; (d) *only the geometric data have errors*; and (e) *none of them have errors*. Case (a) is what we previously assumed. If the errors are not coherent as in case (b), then the smoothing process should be conducted separately. Let $G(t) \subset \mathbb{R}^3$ and $F(t) \subset \mathbb{R}^{\kappa-3}$ denote the geometry and the physics information at time t, respectively. Then (10) becomes the following two problems.

$$\begin{cases} \text{Find a smooth } g(t) \in \mathbb{R}^3 \text{ such that} \\ (\partial_t g(t), \theta)_{G(t)} + \left(D\nabla_{G(t)} g(t), \nabla_{G(t)}\theta\right)_{TG(t)} = (r_g(t), \theta)_{G(t)}, G(0) = G, \end{cases} \tag{11}$$

for all $\theta \in C_0^\infty(G(t))$, and

$$\begin{cases} \text{Find a smooth } f(t) \in \mathbb{R}^{\kappa-3} \text{ such that} \\ (\partial_t f(t), \theta)_{G(t)} + \left(D\nabla_{G(t)} f(t), \nabla_{G(t)}\theta\right)_{TG(t)} = (r_f(t), \theta)_{G(t)}, F(0) = F, \end{cases} \tag{12}$$

for all $\theta \in C_0^\infty(G(t))$, where $G(t)$ is the solution of (11) at time t, $r_g(t) \in \mathbb{R}^3$ and $r_f(t) \in \mathbb{R}^{\kappa-3}$ are the decomposition of r. Here we assume the time t in both problems (11) and (12) goes at the same speed. Obviously, this is not necessary if we assume the two types of errors are not related in any way. However, there is some difficulty in determining which geometry mesh should be used for smoothing the functional data. Furthermore, using different evolution speeds for the two problems will cause the dual stiffness matrices. This will duplicate the computational costs.

In case (c), we separate the geometry and physics. We use the notation $G = G(t)$ to denote the geometry, and again use $F(t) \subset \mathbb{R}^{\kappa-3}$ to denote the physics information. Then (10) becomes

$$\begin{cases} \text{Find a smooth } f(t) \in \mathbb{R}^{\kappa-3} \text{ such that} \\ (\partial_t f(t), \theta)_G + (D\nabla_G f(t), \nabla_G\theta)_{TG} = (r_f(t), \theta)_G, \quad \forall \theta \in C_0^\infty(G), \\ F(0) = F, \end{cases} \tag{13}$$

where $f(t) \in F(t)$ is the function of $F(t)$. Since G is fixed, the system (13) is linear. In case (d), we need only to solve problem (11). Case (e) does not need to be considered.

3. SUBDIVISION SURFACES

We discretize the proposed diffusion problem in a function space defined by the limit of Loop's subdivision. This section describes only the relevant results on surface subdivision. We show clearly that these results are valid on the subdivision of functions defined on surfaces.

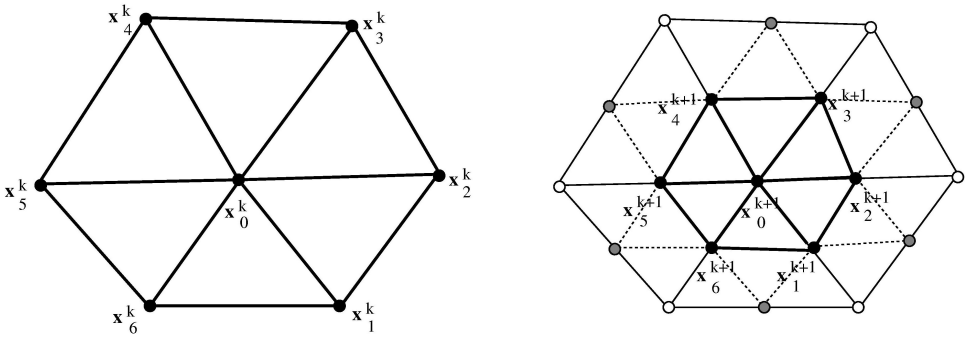

Fig. 3. Refinement of triangular mesh around a vertex.

Subdivision schemes generate smooth surfaces via a limit procedure of an iterative refinement starting from an initial mesh that serves as the control mesh of the limit surface. Several subdivision schemes for generating smooth surfaces have been proposed. Some of them are interpolatory, that is, the vertex positions of the coarse mesh are fixed, and only the newly added vertex positions need to be computed (see, e.g., Kobbelt et al. [1997] for quadrilateral meshes, and Dyn et al. [1990] and Zorin et al. [1996] for triangular meshes), whereas others are approximatory (see, e.g., Catmull and Clark [1978] and Doo and Sabin [1978] for quadrilateral meshes, Loop [1978] for triangular meshes, and Peters and Reif [1997] for general polyhedra). These approximatory subdivision schemes compute both the old and new vertex positions at each refinement step. Generally speaking, approximatory schemes produce better quality surfaces than those produced by interpolatory schemes. Hence, in this work, we use an approximating scheme for triangular meshes proposed by Loop [1978]. This scheme produces C^2 limit surfaces except at a finite number of isolated points where the surface is C^1 (see Schweitzer [1996]).

For Loop's scheme, a fast method exists for evaluating the limit surfaces at any parameter value (see Stam [1988]), especially needed for the numerical computation of the area-integral. Loop's subdivision surfaces in the past have been successfully used in smooth surface reconstruction from scattered data (see Hoppe et al. [1993, 1994]) and in thin-shell finite element analysis (see Cirak et al. [2000]) for describing both the geometry and associated displacement fields.

3.1 Loop's Subdivision Scheme

In Loop's subdivision scheme, the initial control mesh and the subsequent refined meshes consist of only triangles. In a refinement step, each triangle is subdivided linearly into four subtriangles. Then all the vertex positions of the refined mesh are computed as the weighted average of the vertex positions of the unrefined mesh. Consider a vertex x_0^k at level k with neighbor vertices x_i^k for $i = 1, \ldots, n$ (see Figure 3), where n is the valence of vertex x_0^k. The coordinates of the newly generated vertices x_i^{k+1} on the edges of the previous mesh are computed as

$$x_i^{k+1} = \frac{3x_0^k + 3x_i^k + x_{i-1}^k + x_{i+1}^k}{8}, \quad i = 1, \ldots, n, \tag{14}$$

where index i is to be understood modulo n. The old vertices get new positions according to

$$x_0^{k+1} = (1 - na)x_0^k + a\left(x_1^k + x_2^k + \cdots + x_n^k\right), \tag{15}$$

where $a = (1/n)[\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}\cos 2\pi/n)^2]$. Note that all newly generated vertices have a valence of 6, and the vertices inherited from the original mesh at level zero may have a valence other than 6. The former

case is refered to as *ordinary* and the latter case is referred to as *extraordinary*. The limit surface of Loop's subdivision is C^2 everywhere except at the extraordinary points where it is C^1.

3.2 Evaluation of Regular Surface Patches

To obtain a local parameterization of the limit surface for each of the triangles in the initial control mesh, we choose (ξ_1, ξ_2) as two of the barycentric coordinates (ξ_0, ξ_1, ξ_2) and define T as

$$T = \{(\xi_1, \xi_2) \in I\!R^2 : \xi_1 \geq 0, \xi_2 \geq 0, \xi_1 + \xi_2 \leq 1\}. \tag{16}$$

The triangle T in the (ξ_1, ξ_2)-plane may be used as a master element domain. Consider a generic triangle in the mesh and introduce a local numbering of vertices lying in its immediate 1-ring neighborhood (see Figure 6). If all its vertices have a valence of 6, the resulting patch of the limit surface is exactly described by a single quartic box-spline patch, for which an explicit closed form exists Stam [1998]. We refer to such a patch as *regular*. A regular patch is controlled by 12 basis functions:

$$x(\xi_1, \xi_2) = \sum_{i=1}^{12} N_i(\xi_1, \xi_2) x_i, \tag{17}$$

where the label i refers to the local numbering of the vertices that is shown in Figure 6. The surface within the shaded triangle in this figure is defined by the 12 local control vertices. The bases N_i are given as follows (see Stam [1998]).

$$\begin{aligned}
N_1 &= \frac{1}{12}\big(\xi_0^4 + 2\xi_0^3\xi_1\big),\\
N_2 &= \frac{1}{12}\big(\xi_0^4 + 2\xi_0^3\xi_2\big),\\
N_3 &= \frac{1}{12}\big[\xi_0^4 + \xi_1^4 + 6\xi_0^3\xi_1 + 6\xi_0\xi_1^3 + 12\xi_0^2\xi_1^2 + \big(2\xi_0^3 + 2\xi_1^3 + 6\xi_0^2\xi_1 + 6\xi_0\xi_1^2\big)\xi_2\big],\\
N_4 &= \frac{1}{12}\big[6\xi_0^4 + 24\xi_0^3(\xi_1 + \xi_2) + \xi_0^2\big(24\xi_1^2 + 60\xi_1\xi_2 + 24\xi_2^2\big)\\
&\quad + \xi_0\big(8\xi_1^3 + 36\xi_1^2\xi_2 + 36\xi_1\xi_2^2 + 8\xi_2^3\big) + \big(\xi_1^4 + 6\xi_1^3\xi_2 + 12\xi_1^2\xi_2^2 + 6\xi_1\xi_2^3 + \xi_2^4\big)\big],
\end{aligned} \tag{18}$$

where (ξ_0, ξ_1, ξ_2) are barycentric coordinates of the triangle with vertices numbered as 4, 7, 8, and $\xi_0 = 1 - \xi_1 - \xi_2$. Other basis functions are similarly defined. For example, replacing (ξ_0, ξ_1, ξ_2) by (ξ_1, ξ_2, ξ_0) in N_1, N_2, N_3, N_4, we get N_{10}, N_6, N_{11}, N_7. Replacing (ξ_0, ξ_1, ξ_2) by (ξ_2, ξ_0, ξ_1) we get N_9, N_{12}, N_5, N_8.

3.3 Evaluation of Irregular Surface Patches

If a triangle is irregular, that is, at least one of its vertices has a valence other than six, the resulting patch is not a quartic box spline. We assume extraordinary vertices are isolated; that is, there is no edge in the control mesh such that both its vertices are extraordinary. This assumption could be fulfilled by subdividing the mesh once. Under this assumption, any irregular patch has only one extraordinary vertex. For the evaluation of irregular patches, we use the scheme proposed by Stam [1988]. In this scheme the mesh needs to be subdivided repeatedly until the parameter values of interest are interior to a regular patch. We now summarize the central idea of Stam's scheme. First, it is easy to see that

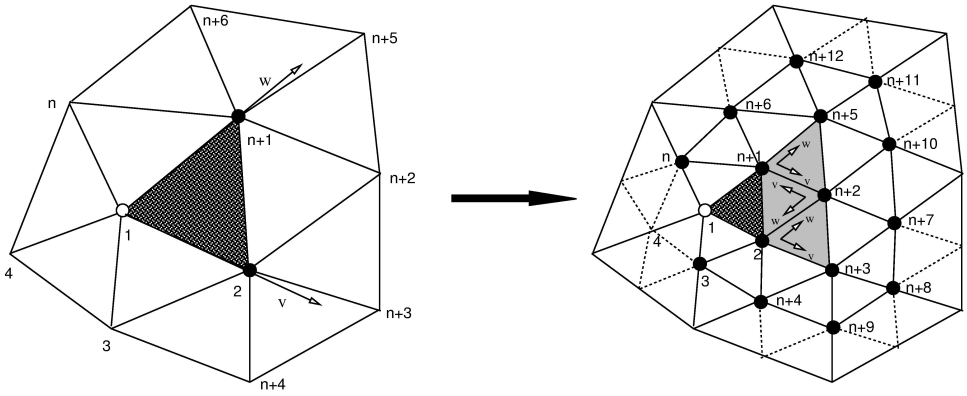

Fig. 4. The vertex with empty circle is extraordinary. After one subdivision, the irregular patch (dark shaded part) is split into one irregular patch (dark shaded part) and three regular patches (light shaded parts).

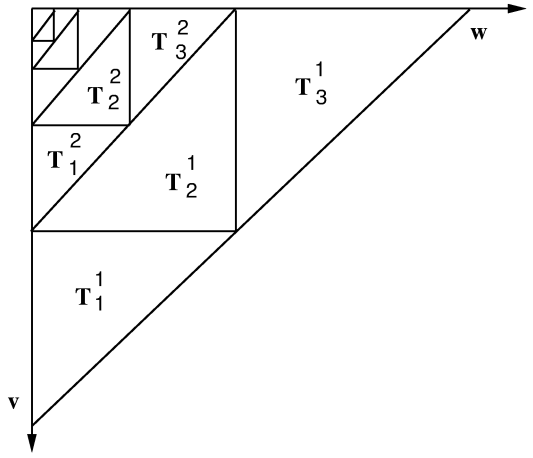

Fig. 5. Refinement in the parametric space, where $(u, v, w) = (\xi_0, \xi_1, \xi_2)$ is the barycentric coordinate of the triangle.

each subdivision of an irregular patch produces three regular and one irregular patch (see Figure 4). Repeated subdivision of the irregular patch produces a sequence of regular patches. The surface patch is piecewise parameterized as shown in Figure 5. The subdomains T_j^k are given as follows.

$$\begin{aligned} T_1^k &= \{(\xi_1, \xi_2) : \xi_1 \in [2^{-k}, 2^{-k+1}], \quad \xi_2 \in [0, 2^{-k+1} - \xi_1]\}, \\ T_2^k &= \{(\xi_1, \xi_2) : \xi_1 \in [0, 2^{-k}], \qquad \xi_2 \in [2^{-k} - \xi_1, 2^{-k}]\}, \\ T_3^k &= \{(\xi_1, \xi_2) : \xi_1 \in [0, 2^{-k}], \qquad \xi_2 \in [2^{-k}, 2^{-k+1} - \xi_1]\}. \end{aligned} \tag{19}$$

These subdomains are mapped onto T by the transform

$$\begin{aligned} t_{k,1}(\xi_1, \xi_2) &= (2^k \xi_1 - 1, 2^n \xi_2), \qquad (\xi_1, \xi_2) \in T_1^k, \\ t_{k,2}(\xi_1, \xi_2) &= (1 - 2^k \xi_1, 1 - 2^k \xi_2), \quad (\xi_1, \xi_2) \in T_2^k, \\ t_{k,3}(\xi_1, \xi_2) &= (2^k \xi_1, 2^k \xi_2 - 1), \qquad (\xi_1, \xi_2) \in T_3^k. \end{aligned}$$

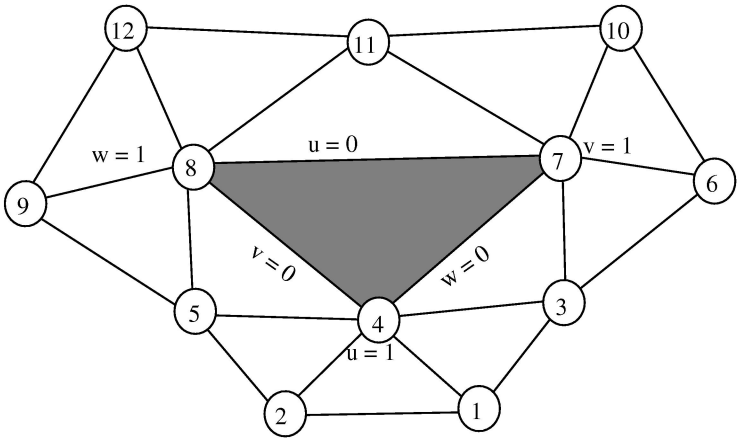

Fig. 6. The vertex numbering of a regular patch with 12 control points. A regular patch is defined over the shaded triangle.

Hence T_j^k form a tiling of T except for the point $(\xi_1, \xi_2) = (0, 0)$. The surface patch is then defined by its restriction to each triangle

$$x(\xi_1, \xi_2)\big|_{T_j^k} = \sum_{i=1}^{12} x_i^{k,j} N_i(t_{k,j}(\xi_1, \xi_2)), \quad j = 1, 2, 3; \quad k = 1, 2, \ldots, \tag{20}$$

where $x_i^{k,j}$ are the properly chosen 12 control vertices (see Figure 6) around the irregular patch at the level k that define a regular surface patch. Using the vertex numbering and local coordinate system shown in Figure 4, it is easy to see that the three sets of control vertices are

$$\begin{aligned}
\{x_i^{k,1}\}_{i=1}^{12} &= \left[x_3^k, x_1^k, x_{n+4}^k, x_2^k, x_{n+1}^k, x_{n+9}^k, x_{n+3}^k, x_{n+2}^k, x_{n+5}^k, x_{n+8}^k, x_{n+7}^k, x_{n+10}^k\right],\\
\{x_i^{k,2}\}_{i=1}^{12} &= \left[x_{n+7}^k, x_{n+10}^k, x_{n+3}^k, x_{n+2}^k, x_{n+5}^k, x_{n+4}^k, x_2^k, x_{n+1}^k, x_{n+6}^k, x_3^k, x_1^k, x_n^k\right],\\
\{x_i^{k,3}\}_{i=1}^{12} &= \left[x_1^k, x_n^k, x_2^k, x_{n+1}^k, x_{n+6}^k, x_{n+3}^k, x_{n+2}^k, x_{n+5}^k, x_{n+12}^k, x_{n+7}^k, x_{n+10}^k, x_{n+11}^k\right].
\end{aligned}$$

Hence, the main task is to compute these control vertices. As usual, the subdivision around an irregular patch is formulated as a linear transform from the level $(k-1)$, 1-ring vertices of the irregular patch to the related level k vertices, that is,

$$X^k = AX^{k-1} = \cdots = A^k X^0, \quad \tilde{X}^{k+1} = \tilde{A}X^k = \tilde{A}A^k X^0,$$

where $X^k = [x_1^k, \ldots, x_{n+6}^k]^T$, $\tilde{X}^k = [x_1^k, \ldots, x_{n+7}^k, x_{n+7}^k, \ldots, x_{n+12}^k]^T$, and A and $\tilde{A}$ are defined by the subdivision rules. Hence, $k+1$ subdivisions lead to the computation of A^k. When k is large, the computation can be very time consuming. A novel idea proposed by Stam is to use the Jordan canonical form $A = SJS^{-1}$. The computation of the A^k amounts to computing J^k, which makes the cost of the computation nearly independent of k and hence very efficient. The beauty of the scheme is explicit forms of S and J exist. We refer to Stam [1998] for details.

3.4 Basis Functions and Classification of Patches

For each vertex x_i of a control mesh M_d, we associate it with a basis function ϕ_i, where ϕ_i is defined by the limit of Loop's subdivision for the zero control values everywhere except at x_i, where it is one (see Figure 7(a)). Hence the support of ϕ_i is local and it covers the 2-ring neighborhood of vertex x_i.

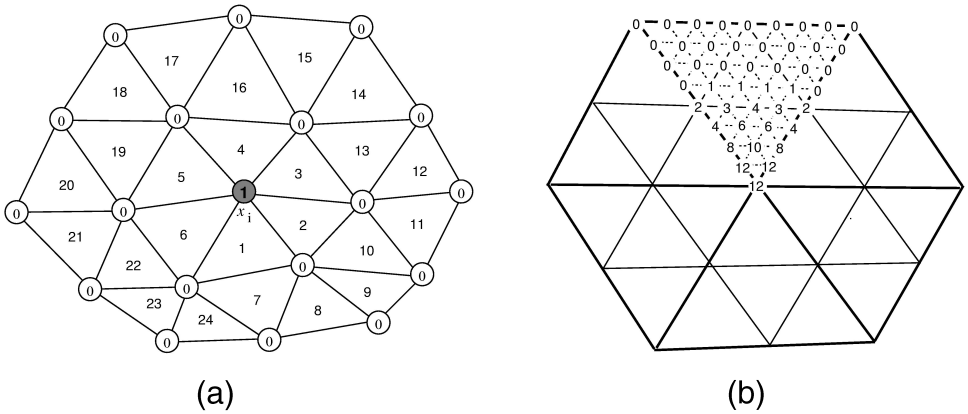

Fig. 7. (a) Numbered 2-ring neighborhood elements of vertex x_i. The vertex numbers in circles are the control coefficients for defining the basis ϕ_i; (b) the quartic BB-form coefficients (each has a factor 1/24) of the basis function. The coefficients on the other five macrotriangles are obtained by rotating the top macrotriangle around the center to the other five positions.

Let e_j, $j = 1, \ldots, m_i$ be the 2-ring neighborhood elements. Then if e_j is regular, the explicit box-spline expression as in (17) exists for ϕ_i on e_j. Using Equations (18), we could derive the BB-form (Bernstein–Bezier form) coefficients for basis ϕ_i (see Figure 7(b)). All these coefficients have a factor 1/24. Hence, the function value at x_i is $\frac{1}{2}$. Note that the basis ϕ_i derived is the same as the triangular C^2 quartic basis given by Sabin [1976]. These expressions could be used to evaluate ϕ_i in forming the linear system (26). If e_i is irregular, local subdivision, as described in Section 3.3, is needed around e_i until the parameter values of interest are interior to a regular patch.

Note the difference of the basis ϕ_i from the basis N_j in (17). ϕ_i is a piecewise function whose support covers 2-ring neighbor triangles, whereas N_j in (17) is defined on one triangle only.

Using the basis $\{\phi_i\}$, the limit surface of Loop's subdivision is expressed as $M = \sum x_i \phi_i(x)$. However, each triangular surface patch of M is defined locally by only a few related basis functions, since the supports of the basis are compact. For a triangle $[x_i x_i x_k]$, the related bases that define the surface patch over the triangle are uniquely determined by the valences n_i, n_j, and n_k; here n_i, n_j, and n_k are the valences of vertex x_i, x_j, and x_k, respectively. Hence, two triangles that have the same valence for each of the three vertices will have the same set-related basis functions. To reduce the computation costs of evaluating these functions in the numerical integration, triangles are classified into categories according their vertex valences. All members in one category will have the same vertex valences, hence the same set-related basis functions. The surface evolution in time t does not change the vertex valences of the mesh. Hence, for one category of patches, we only need to evaluate the basis functions once. Using tree structures, the classification could be conducted within a linear time.

4. DISCRETIZATION

In Riemannian geometry, differentiable functions are smooth and C^∞. However, our discretized version of the diffusion problem is in the class C^1. As we mentioned earlier, the functions are defined by the limit of Loop's subdivision. Such a function is C^2 smooth everywhere except at the extraordinary vertices, where it is C^1. The function is locally parameterized as the image of the unit triangle defined by $T = \{(\xi_1, \xi_2) \in I\!R^2 : \xi_1 \geq 0, \xi_2 \geq 0, \xi_1 + \xi_2 \leq 1\}$. That is, $(1 - \xi_1 - \xi_2, \xi_1, \xi_2)$ is the barycentric coordinate of the triangle. Using this parameterization, our discretized representation of M is $M = \bigcup_{\alpha=1}^{k} \mathcal{T}_\alpha$, $\mathring{\mathcal{T}}_\alpha \cap \mathring{\mathcal{T}}_\beta = \emptyset$ for $\alpha \neq \beta$, where $\mathring{\mathcal{T}}_\alpha$ is the interior of the *triangular function patch* $\mathcal{T}_\alpha$. Each

triangular function patch is assumed to be parameterized locally as

$$x_\alpha : T \to \mathcal{T}_\alpha;\ \ (\xi_1, \xi_2) \ \mapsto\ x_\alpha(\xi_1, \xi_2), \tag{21}$$

where $x_\alpha(\xi_1, \xi_2)$ is defined by Equations (17) through (20). Unlike the differentiable structure of a manifold, our parameterization has no overlap. Each point $p \in M$ has unique parameter coordinates, except at the boundary of the patches. Under this parameterization, tangents and gradients can be computed directly. The integration (1) is replaced by

$$\int_M f\, dx := \sum_\alpha \int_T f(x_\alpha(\xi_1, \xi_2))\sqrt{det(g_{ij})}d\xi_1\, d\xi_2. \tag{22}$$

The integration on the triangle T is computed adaptively by numerical methods.

4.1 Spatial Discretization

Let M be the limit function of the initial control mesh M_d. Then, instead of solving the problem (10), we solve the following alternative problem

$$\begin{cases} \text{Find}\ \ x(t) \in V^k_{M(t)}, \ \ \text{such that} \\ (\partial_t x(t), \theta)_{M(t)} + \big(D\nabla_{M(t)}x(t), \nabla_{M(t)}\theta\big)_{TM(t)} = (r(t), \theta)_{M(t)}, \quad \forall\theta \in V_{M(t)} \\ M(0) = M, \end{cases} \tag{23}$$

where $V_{M(t)} \subset C^1(M(t))$ is a finite-dimensional space spanned by the basis functions $\{\phi_i\}_{i=1}^m$. Let $x(t) = \sum_{i=1}^m x_i(t)\phi_i$, $x_i(t) \in I\!R^\kappa$, and $\theta = \phi_j$. Then (23) may be written as

$$\begin{cases} \sum_{i=1}^m x_i'(t)(\phi_i, \phi_j)_{M(t)} + \\ \sum_{i=1}^m x_i(t)\big(D\nabla_{M(t)}\phi_i, \nabla_{M(t)}\phi_j\big)_{TM(t)} = \omega \sum_{i=1}^m (x_j - x_i(t))(\phi_i, \phi_j)_{M(t)}, \\ x_j(0) = x_j, \end{cases} \tag{24}$$

for $j = 1, \ldots, m$, where x_j is the jth vertex of the initial mesh M_d. (24) is a set of nonlinear ordinary equations for the unknown functions $x_i(t)$, $i = 1, \ldots, m$. The system is nonlinear because the domain $M(t)$, over which the integrations are taken, is also unknown. It should be noted that ϕ_i depends upon the topological structure of the mesh around x_i, but not on the positions of the vertices. Hence for any time t, ϕ_i is kept the same.

4.2 Time Discretization

Given a timestep $\tau > 0$, suppose we have an approximate solution at $t = n\tau$. Now we construct an approximate solution at the next timestep $t = (n+1)\tau$ by a semi-implicit Euler scheme. Let X^n be an approximation of $x(n\tau)$. Then the semi-implicit discretization of (24) is

$$(X^{n+1} - X^n, \phi_i)_{M(n\tau)} + \tau\big(D^n\nabla_{M(n\tau)}X^{n+1}, \nabla_{M(n\tau)}\phi_i\big)_{TM(n\tau)} = \tau\omega(X^0 - X^{n+1}, \phi_i)_{M(n\tau)}, \tag{25}$$

for $i = 1, \ldots, m$. We call this a semi-implicit discretization of (24) because the integration domain is chosen to be the solution of the previous step. Let $x(t) = \sum_{i=1}^m x_i(t)\phi_i$. Then (25) can be written as a linear system:

$$((1 + \tau\omega)M^n + \tau L^n(D^n))X((n+1)\tau) = M^n(X(n\tau) + \tau\omega X(0)), \tag{26}$$

where

$$\begin{aligned} X(t) &= [x_1(t), \ldots, x_m(t)]^T, \quad X(0) = [x_1, \ldots, x_m]^T, \\ M^n &= \big((\phi_i, \phi_j)_{M(n\tau)}\big)_{i,j=1}^m, \\ L^n(D^n) &= \big((D^n\nabla_{M(n\tau)}\phi_i, \nabla_{M(n\tau)}\phi_j)_{TM(n\tau)}\big)_{i,j=1}^m. \end{aligned}$$

Note that both M^n and $L^n(D^n)$ are symmetric. Since $\phi_1, \phi_2, \ldots, \phi_m$ are linearly independent and have compact support, M^n is sparse and positive definite. Similarly, $L^n(D^n)$ is symmetric and nonnegative definite. Hence, $(1+\tau\omega)M^n + \tau L^n(D^n)$ is symmetric and positive definite. The computation of the coefficient matrix requires the computation of the integrations $(\phi_i, \phi_j)_{M(n\tau)}$ and $(D^n\nabla_{M(n\tau)}\phi_i, \nabla_{M(n\tau)}\phi_j)_{TM(n\tau)}$. These are computed by Gauss quadrature formulas over each triangle and then summed. For each timestep, the system needs to be newly generated, because M is changed. However, the values of the basis functions do not need to be recomputed.

Inasmuch as the support of ϕ_i is the 2-ring neighbor triangles, $(\phi_i, \phi_j)_{M(n\tau)} = 0$ if x_j is not a 3-ring neighbor vertex of x_i. Hence the coefficient matrix of the system (26) is highly sparse (each row has about 37 nonzero elements on average), and hence an iterative method for solving such a system is desirable. We use Gauss–Seidel iterations if the timestep $\tau < 350/N$, and otherwise use a conjugate gradient method with a diagonal preconditioning. Here N is the number of triangles of the mesh. The choice of the switch point $350/N$ is based on the experiment.

It should be mentioned that the derived system (26) is valid for solving problems (10) through (13), although it is derived for (10) only. Note that $X(t)$ is an $m \times k$ matrix. If the Riemannian metric is defined by the scalar product in $\mathbb{R}^3$, then the first 3 columns of $X(t)$ are the solution of (11), and the last $\kappa - 3$ columns are the solution of (12) and (13). For all the cases we mentioned in Section 2, the coefficient matrix of the system as well as the left-hand side are computed in the same way. The only difference is we do not need to compute the first three columns of the left-hand side for problem (13), since the geometry is fixed.

Stopping Criteria

We need to determine a time moment T ($T > 0$), where the evolution procedure stops. Since the evolution procedure is based on a mean curvature motion, we determine T by examining the reduction rate of the mean curvature. For a given mesh, deciding which portion is noisy and should be smoothed out is subjective. Furthermore, the information at time t is not enough to judge whether the smoothing is satisfactory. Therefore, we always compare the evolution effect with respect to the initial state. Let

$$\mathcal{H}(t) = \int_{M(t)} \|H(t,x)\|^2\,dx \Big/ \int_{M(0)} \|H(0,x)\|^2\,dx,$$

where $H(t,x)$ is the mean curvature vector at the point x and time t. We use the derivative (see (28)) of $\mathcal{H}(t)$ to test the stable state of the evolution. If the data are not very noisy and the shape of the mesh is not complicated, such as the sphere data, $\mathcal{H}(t)$ reduces slowly. In this case, the stopping criterion (28) works well. However, the derivative sometimes cannot help us make the right judgment. For instance, if the shape of a mesh is complicated, even though it is not noisy, $\mathcal{H}(t)$ still reduces quickly for quite a long time and then slows down. Using the derivative of $\mathcal{H}(t)$ in such a case will lead to late termination, which makes the mesh oversmoothed (fine detail features are lost). In this case, we prefer to use $\max \|x_i(t) - x_i\|$ to control the termination (see (29)). If the data are very noisy (high-frequency noise), $\mathcal{H}(t)$ reduces quickly at the beginning of the evolution and slows down quickly. In this case, examining how much the derivate is reduced relative to $\mathcal{H}'(0)$ is more reasonable (see (27)). Of course, there are an infinite number of cases between these extremes. These considerations make us choose the following three stopping conditions.

$$\left|\mathcal{H}'(t)/\mathcal{H}'(0)\right| \le \epsilon_1, \quad \text{or} \tag{27}$$

$$\left|\mathcal{H}'(t)\right| \le \epsilon_2, \quad \text{or} \tag{28}$$

$$\max \|x_i(t) - x_i\| \ge \epsilon_3, \tag{29}$$

where ϵ_i are user-specified control constants. Based on experience, we choose $\epsilon_1 = 0.005$, $\epsilon_2 = 8.0$, $\epsilon_3 = 0.2$. The evolution stops if one of the three conditions is satisfied. If the data are very noisy, condition (27) is most likely satisfied first. If the data are smooth, and the shape is simple, condition (28) is most likely satisfied first. The remaining case may first be satisfied by condition (29).

Choice of Timestep τ
Suppose the final result we want is at time T. This time moment can be approached through several timesteps. Although the semi-implicit discretization is stable, the timestep has a significant effect on the linear system derived. If the timestep is large, the iteration method for solving the system converges slowly. On the contrary, if the timestep is very small, the surface will have no significant change, therefore more steps are required. We determine τ according to the change rate of the surface size. Denote the x, y, and z components of the surface point $x(t)$ and the functional components on the surface as $x_1(t), \ldots, x_k(t)$. Then from (23), we have

$$(\partial_t x_i(t), x_i(t))_{M(t)} = -\big(D\nabla_{M(t)} x_i(t), \nabla_{M(t)} x_i(t)\big)_{TM(t)}, \quad i = 1, \ldots, k.$$

Since D is positive definite, we have

$$\frac{\partial (x(t), x(t))_{M(t)}}{\partial t} = 2(\partial_t x(t), x(t))_{M(t)} = -2\sum_{i=1}^{k} E_i(t) \leq 0, \tag{30}$$

where $E_i(t) = (D\nabla_{M(t)} x_i(t), \nabla_{M(t)} x_i(t))_{TM(t)} \geq 0$, and $E(t) := \sum_{i=1}^{k} E_i(t)$ is called the *energy* of the surface $M(t)$ at time t.

If $D = 1$ and $k = 3$, it is not difficult to show, by (3), that $E(t) = 2\text{Area}(M(t))$.

From (30), we know that if $k = 3$, the surface size $S(M(t)) = (x(t), x(t))_{M(t)}$ decreases with a speed $4\text{Area}(M(t))$. For one step evolution, the surface size decreases approximately by the amount of $4\tau\text{Area}(M(t))$. If we want the size of the surface to decrease around 1% of the $\text{Area}(M(t))$, then we should choose $\tau = 0.0025$ approximately. Such an amount of change in size is visually significant. In our experiments, even at $\tau = 0.0001$ the change is still visually significant at the early stages of the evolution; however, at later stages, the change becomes increasingly less significant. Usually, we choose τ around 0.001 and determine the number n of iterations from $n\tau \approx T$.

4.3 Handling of Surface Boundaries

If the given triangulation has boundaries, the limit surface patches for the triangles that are incident to the boundaries are incompletely defined by only a 1-ring of existing neighboring vertices. We solve the problem by constructing an artificial triangle layer on the outside of the boundary, so that all boundary triangles become interior triangles.

The artificial triangle layer is constructed as follows. For each triangle on the boundary, another triangle is constructed, so that when Loop's subdivision rule is applied to the boundary edge, it will reproduce the midpoint of the edge. Let $[x_i x_j]$ be a boundary edge, and $[x_i x_j x_k]$ be the boundary triangle; then the new triangle is $[x_i x_j x_k']$, with $x_k' = x_i + x_j - x_k$.

In solving the linear system (26), since the boundary vertices and the artificial boundary vertices are fixed, the corresponding unknowns for the boundary vertices now become known, and move to the right-hand side of the equation.

5. ANISOTROPIC DIFFUSION TENSOR

The aim of anisotropic diffusion is to enhance sharp features in one direction and smoothing in another direction. To this end, we need to introduce the concept of *principal curvatures* and *principal directions* of a two-manifold $M \subset \mathbb{R}^\kappa$. Let n be a normal vector field on M. Let $\mathcal{A}_n$ be the *second fundamental*

tensor with respect to n (see Willmore [1993, pp. 119–121]). Then $\mathcal{A}_n$ is a self-adjoint map from TM to TM. The *principal curvatures* $k_1(x)$, $k_2(x)$ and *principal directions* $e_1(x)$, $e_2(x)$ with respect to n are defined as the eigenvalues and the orthonormal eigenvectors of $\mathcal{A}_n$. However, the principal curvatures and principal directions are not uniquely defined since the normal vector field is not so for $\kappa > 3$. We choose a vector field $h(x) = H(x)/\|H(x)\|$, which is the normalized mean curvature vector field of the manifold M and is uniquely defined. Here $H(x)$ is the mean curvature vector given by

$$H(x) = \frac{Q(g_{22}t_{11} + g_{11}t_{22} - 2g_{12}t_{12})}{2\det(G)}, \tag{31}$$

where $t_i = \partial/\partial\xi_i$, $t_{ij} = \partial^2/\partial\xi_i\partial\xi_j$, and $Q = I - [t_1, t_2]G^{-1}[t_1, t_2]^T \in \mathbb{R}^{\kappa\times\kappa}$. Considering the diffusion equation described is the mean curvature motion, choosing this vector field is natural.

We now illustrate how to compute the principal curvatures and principal directions with respect to h. Due to the space limitation, the detailed derivations are given in Xu and Bajaj [2002]. Let

$$A = \Lambda^{-1/2}KF_hK^T\Lambda^{-1/2} \in \mathbb{R}^{2\times2}, \quad [u_1, u_2] = [t_1, t_2]K^T\Lambda^{-1/2},$$

where $F_h = -(t_{ij}^T\ h(x))_{ij=1}^2$, $K \in \mathbb{R}^{2\times2}$, and $\Lambda \in \mathbb{R}^{2\times2}$ are defined by

$$G = K^T\Lambda K, \quad K^TK = I, \quad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2).$$

Let A be expressed as

$$A = P\ \mathrm{diag}(k_1, k_2)\ P^T, \quad \text{with}\ \ P^TP = I.$$

Then k_1 and k_2 are the principal curvatures and v_1 and v_2, defined by

$$[v_1, v_2] := [u_1, u_2]P = [t_1, t_2]K^T\Lambda^{-1/2}P,$$

are the corresponding principal directions with respect to the direction vector h.

Now we define our anisotropic diffusion tensor. Let $k_{\epsilon,1}$, $k_{\epsilon,2}$ be the principal curvatures, and $e_{\epsilon,1}(x)$, $e_{\epsilon,2}(x)$ be the principal directions of M_ϵ at point $x(t)$. Then any vector z could be expressed as

$$z = \alpha e_{\epsilon,1}(x) + \beta e_{\epsilon,2}(x) + N_\epsilon(x),$$

where $N_\epsilon(x)$ is the normal component of z. Then define $D := D_\epsilon$ by

$$D_\epsilon z = \alpha g(k_{\epsilon,1})e_{\epsilon,1}(x) + \beta g(k_{\epsilon,2})e_{\epsilon,2}(x) + N_\epsilon(x),$$

where $g(s)$ is defined by

$$g(s) = \begin{cases} 1; & |s| \le \lambda, \\ \left(1 + \frac{(s-\lambda)^2}{\lambda^2}\right)^{-1}; & |s| > \lambda. \end{cases} \tag{32}$$

$M_\epsilon(t)$ is the solution of (10) at time ϵ with $D = 1$ and initial value $M(t)$. λ is a given parameter that detects sharp features. The reason we use $M_\epsilon(t)$ instead of $M(t)$ to compute the diffusion tensor is that the evaluation of the shape parameters on a noisy function might be misleading with respect to the original but unknown function. Hence we prefilter the current function $M(t)$ by mean curvature motion before we evaluate the shape parameters. Figures 8 and 9 show the results of our anisotropic smoothing solution.

6. DISTINCT FEATURES OF THE SMOOTHING SCHEME

We have illustrated in Section 1 that using Loop's subdivision does have some advantages over using linear finite elements, because the support of the Loop's basis is bigger than that of the hat basis function and the Loop's basis is smooth. The difference in the size of support of basis functions makes

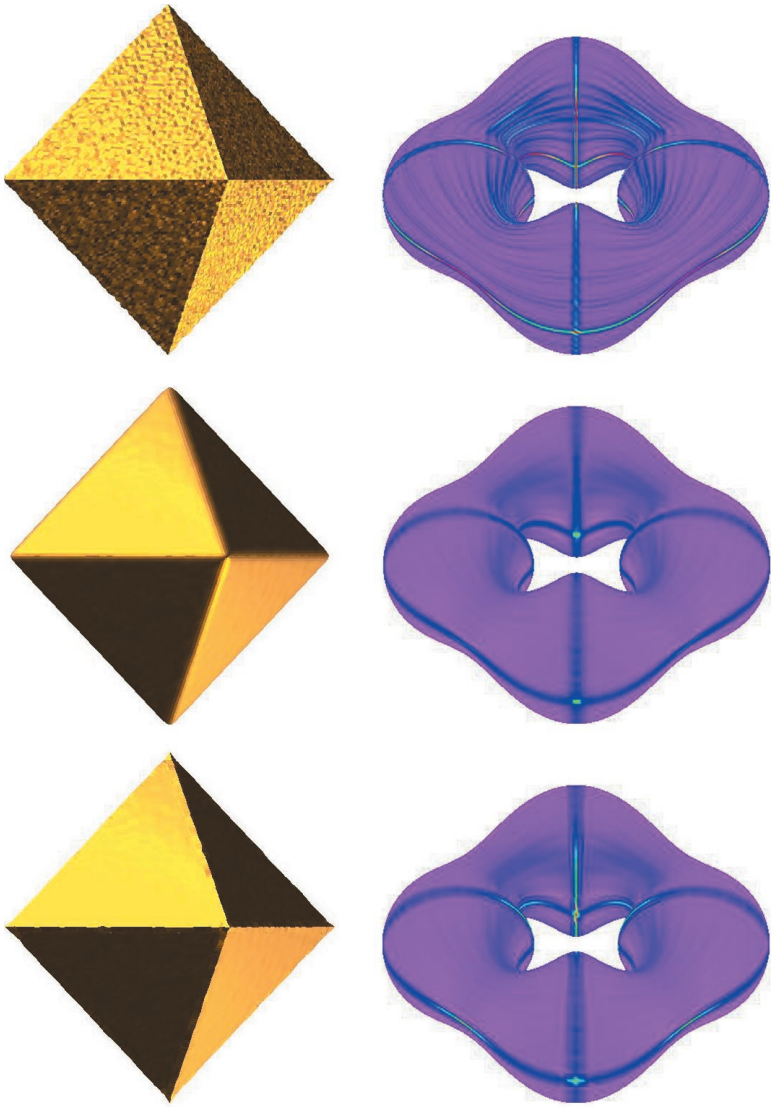

Fig. 8. First column: the first figure is a bumpy input mesh (32,786 triangles). The second and third figures are the faired meshes after four fairing iterations, with the identity and an anisotropic diffusion tensor, respectively. For the anisotropic diffusion tensor, we choose $\lambda = 2$, $\epsilon = 0.001$. Second column (mean curvature plots): the first figure is the input mesh (25,600 triangles) with two noisy functions on the surface. The functions are not smooth (but continuous) at the planes $x = 0$, $y = 0$, and $z = 0$. The second and third figures are the results after four fairing iterations, with identity and an anisotropic diffusion tensor ($\lambda = 2.5$, $\epsilon = 0.001$), respectively.

our evolution more efficient than those previously reported, due to the increased bandwidth of affected frequencies. The reduction speed of high-frequency noises of our approach is not that drastic, but still fast, and the reduction speed of lower-frequency noises is not that slow. Hence the bandwidth of affected frequencies is wider. This could also be observed from the linear system of the discretization. For the linear element approach, each vertex relates only to its 1-ring neighbors, whereas for our approach, each vertex relates to its 3-ring neighbors.

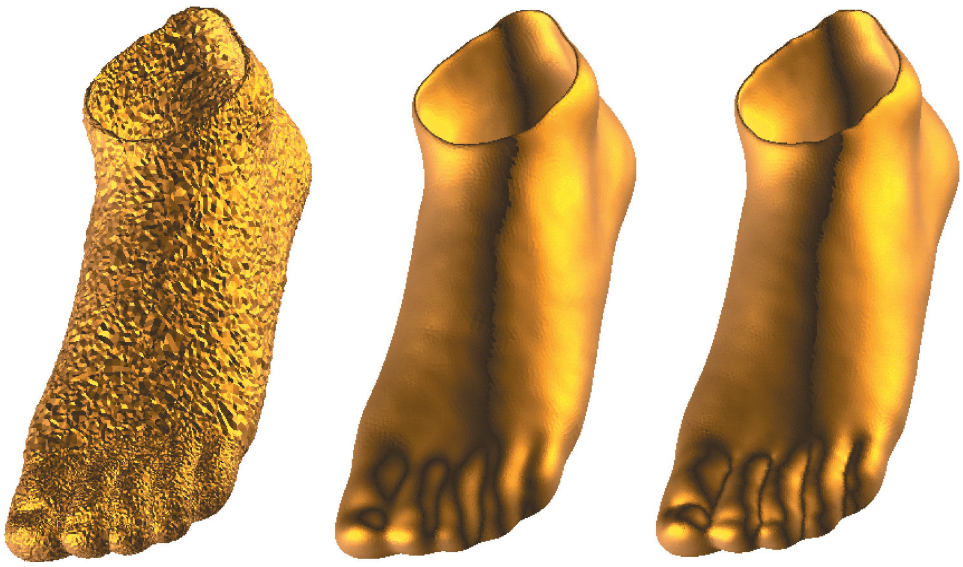

Fig. 9. The left figure shows the initial geometry mesh. The second and third figures show the faired meshes after three fairing iterations with identity and nonidentity diffusion tensors, respectively, where timestep $\tau = 0.001$, and $\lambda = 2.0$, $\epsilon = 0.001$ for the nonidentity diffusion tensor.

It is well known that Loop's subdivision generates quite pleasant-looking surfaces. We believe that using Loop's subdivision surface as the smoothing target is a good choice. Using this smooth representation of the surface, gradient, normal, and curvature at any point can be computed easily. Furthermore, the evolution process produces not only a sequence of attributed triangular meshes at different timesteps, but also a sequence of smooth functions. By sampling these smooth functions, new attributed triangular meshes at a resolution higher than that of the original mesh can be produced.

Apart from the linear finite element approach of Clarenz et al. [2000], we have also implemented Taubin's [1995] signal processing approach and Desbrun et al.'s [2000a], implicit fairing approach for comparison with our approach. Figures 11 and 12 show the smoothing results for these schemes. The input mesh is shown in Figure 10. The noise of the mesh comes from a lossy mesh compression-decompression scheme, in which the binary representation of the vertices is truncated using a specified number of bits. The first row of Figure 11 shows the smoothing results (after 10, 70 smoothing steps) of the signal processing approach. Parameters in the scheme are chosen to be the best ones as Taubin [1995] suggested, where we take $k_{pb} = 0.1$, $\lambda = 0.631398$, and $\mu = -0.673952$. The results show that after a dozen smoothing steps, further iterations do not have much smoothing effect. A good feature of this approach is that detailed features are well kept, but the smoothed surfaces are not smooth enough (underfairing). The second row contains the implicit fairing results (two smoothing steps with $\lambda = 0.005$), where the approximated Laplacian is given by (11) of Desbrun et al. [2000a]. The first row of Figure 12 illustrates the smoothing results (two smoothing steps with timestep length 0.001) of Clarenz et al. [2000]. The results show that the behavior of the implicit fairing and Clarenz et al's fairing are similar. Both of them give quite good smoothing results, but several small features are smoothed out too (overfairing). The second row of Figure 12 contains the results of our approach. It can be seen that much more detailed features are kept and the global flatter features are much smoother. These results show that our results are the best among the four approaches. Another example to compare the behavior of our approach with the linear element approach is given in Figure 2.

Fig. 10. The input noisy triangular mesh of a head for Figures 11 and 12.

7. CONCLUSIONS AND EXAMPLES

We have presented a PDE-based anisotropic diffusion approach for fairing noisy geometric surface data and function vector data on the surface. The finite-element discretization of the diffusion problem is realized by a combination of the limit function representation of Loop's subdivision together with the diffusion model.

Figure 13 shows isocontour plots of the functions on surfaces. For a function $f(x)$ defined on a smooth surface M, an *isocontour* or *isocurve* is defined as $\{x \in M : f(x) = c\}$ for a given constant c, which is called the *isovalue*. The smoothness of the isocontours reflects the smoothness of the function. Hence a simple approach for visualizing the smoothness of the function on a surface is to plot a family of isocontours for a given sequence of isovalues $\{c_i\}_{i=1}^{n}$. In our problem, we have a function vector $x(t) \in \mathbb{R}^{\kappa}$ instead of one scalar function. One way to visualize them together is to plot isocontours for $\|x(t)\|^2$. Figure 13 shows both the geometry (the first column) and the function (the second column) isotropic diffusion effects.

Figure 14 shows the application of the diffusion process to surface texture maps. We first generate 2D texture vector coordinates (u, v) at each of the vertices of the surface triangulation, and then using a pseudorandom number generator, a noise is introduced to each of the components. The vector coordinates are smoothed by treating them as functions on a surface. To show the regularizing effect of the diffusion process, the textures chosen are 512×512 images with regular patterns. The texture for the bunny is a netlike pattern woven from strips. The texture for the torus model consists of alternating blue and green squares with a red disc in each of the squares. The first row is the initial texture map. The second and third are after one and five fairing iterations of the texture vector coordinates.

In all the examples given above, we have chosen the parameter ω in Equation (9) to be zero. This is because the ideal smoothing results are obtained by a short period of time evolution. However, from the equalities in (6), we can see that a longstanding evolution will cause the surface to shrink towards the origin. To avoid this occurring in long time evolution, we choose $\omega > 0$, so that the evolved surface approximates the initial surface $M(0)$. It is easy to understand that the larger ω we choose, the better the approximation to $M(0)$ we achieve. However, if $M(0)$ is very noisy, too large ω will make the evolved

Fig. 11. First row: the smoothing results of Taubin's signal processing approach. Second row: the smoothing results of Desbrun et al.'s implicit fairing approach.

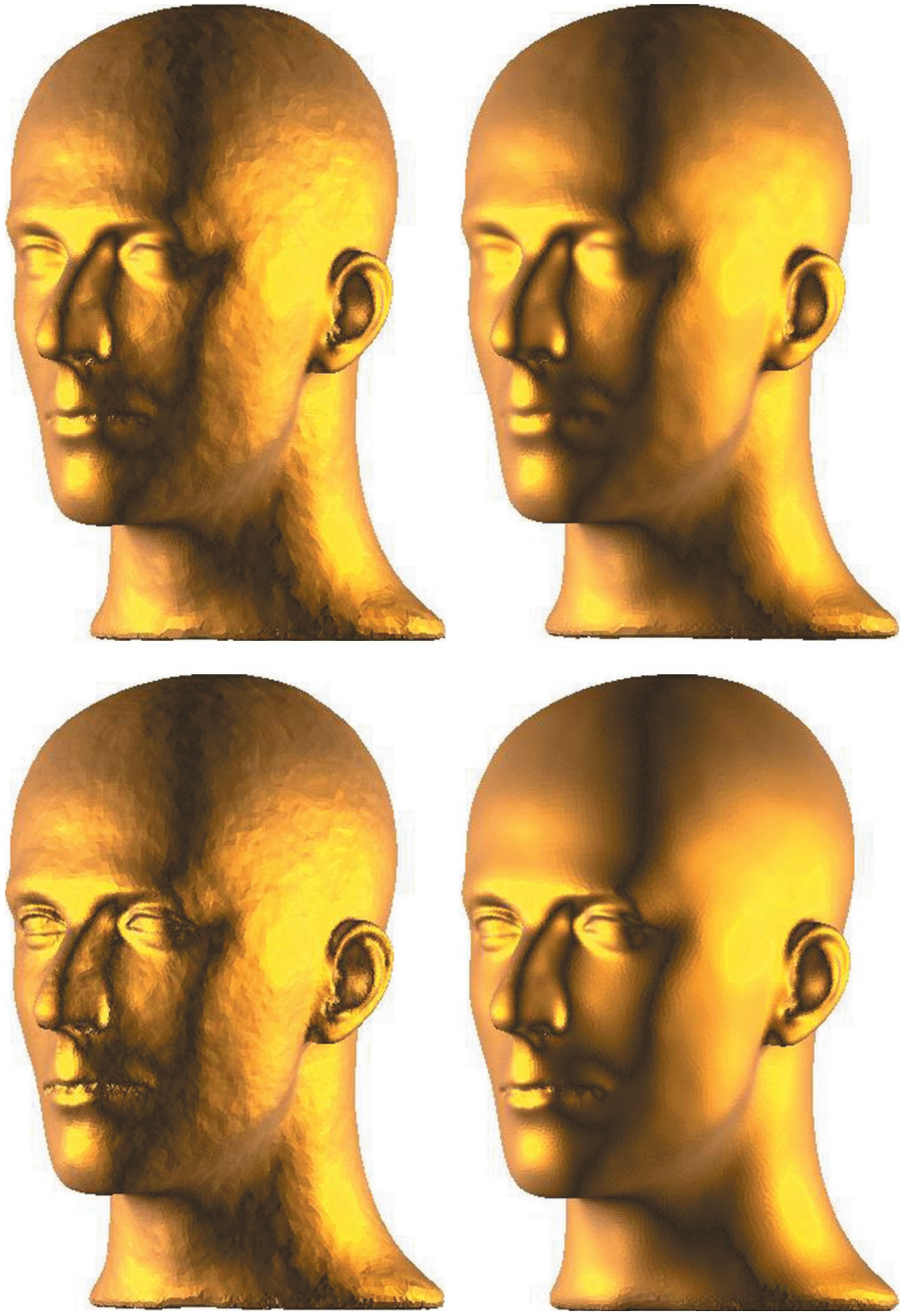

Fig. 12. First row: the smoothing results of Clarenz et al.'s approach. Second row: The smoothing results of our approach with isotropic diffusion tensor.

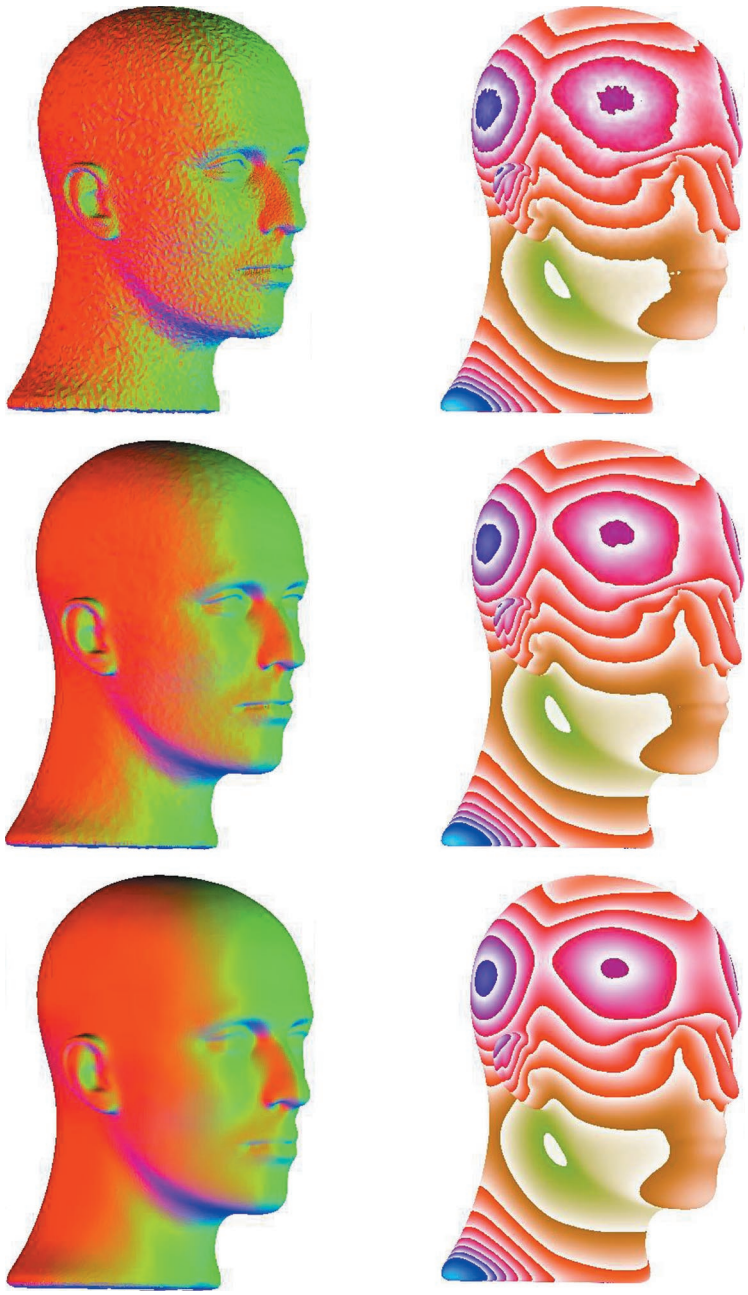

Fig. 13. First column: the first figure is the input noisy geometry (102, 208 triangles); the second and third figure are the geometry diffusion results after 1 and 4 fairing iterations, respectively. Second column: the isocontour plots of the function $\|x\|^2$ on the smoothed head. The three figures show the results after 0, 1, and 4 fairing iterations, respectively.

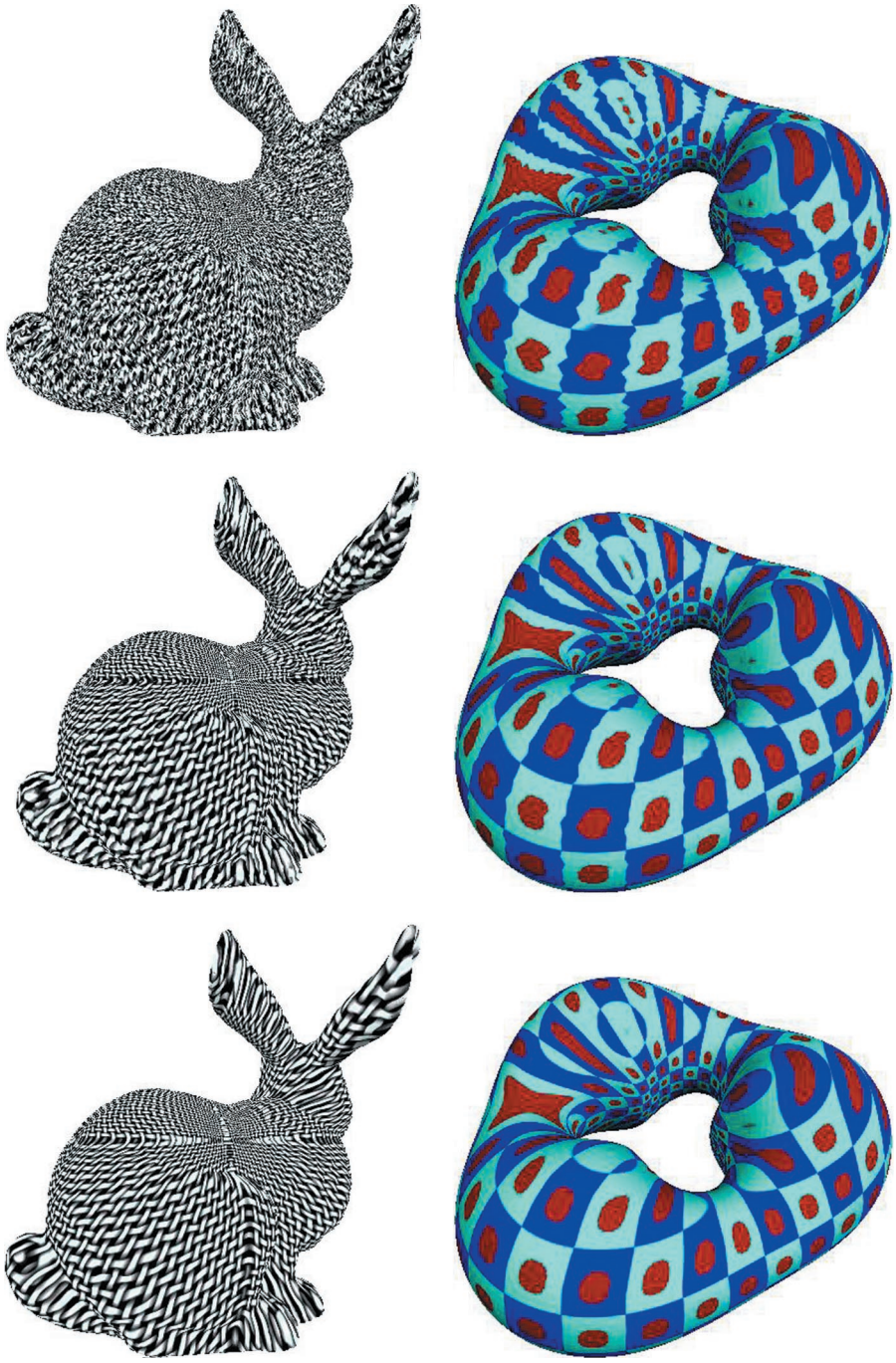

Fig. 14. Each of the two columns shows the isotropic diffusion of initial noisy texture maps data after one and five fairing iterations, respectively.

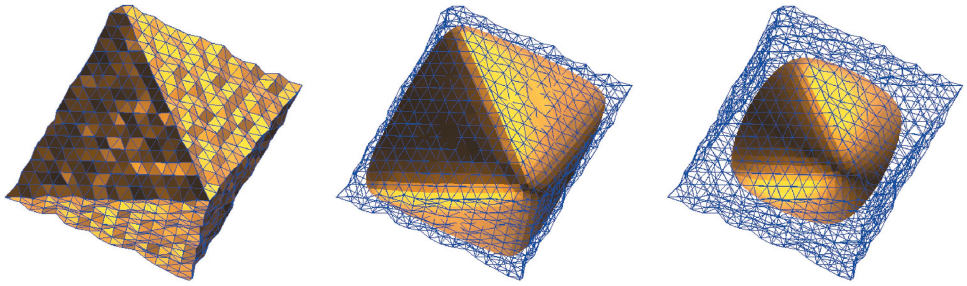

Fig. 15. The left is the input. The remaining two figures are the stable states for $\omega = 10.0, 2.5$, respectively. The net outside each of the surface meshes consists of all the edges of the initial mesh.

Table I. First Row: Values of ω. Second Row: Areas of Evolved Surfaces at Stable States. Third Row: Iteration Numbers for Arriving at Stable States. Fourth Row: Maximal Errors Between Initial and Final Meshes. Fifth Row: Mean Errors Between Initial and Final Meshes

Parameter ω	1.25	2.5	5.0	10.0	20.0	40.0	80.0
Surface Area	$8.75^{*}10^{-4}$	29.21	43.22	50.24	54.47	57.31	59.47
Iteration No.	217	431	175	91	55	34	19
Maximal Error	3.071	1.370	0.894	0.633	0.457	0.331	0.234
Mean Error	2.102	0.577	0.250	0.124	0.070	0.045	0.032

Table II. Second Column: Number of Triangles. Third Column: Times for Computing Stiff Matrix. Fourth Column: Number of Iterations for Solving Linear Systems for Four Timesteps. Last Column: Average Times (per Iteration) for Solving Linear Systems

Examples	Triangles #	Form matrix	Iterations #	Averaging
Fig 1.1(left)	146,036	24.8s	23, 24, 57, 86	0.964s
Fig 6.3(right)	102,208	12.2s	23, 18, 16, 17	0.202s
Fig 5.1(left)	32,786	3.5s	13, 12, 12, 12	0.021s

surface less smooth. On the other hand, too small ω could not prevent the evolved surface from shrinking to zero. Hence choosing a proper ω is a crucial subproblem. Here a theoretical question that is left open is: *for a given error bound ϵ, determine ω so that the evolved surface approximates (for any time) the initial surface within error ϵ*. Figure 15 shows the effect of ω. The left figure shows the input mesh. The figures in the middle and right show the stable states of the evolution for $\omega = 10.0, 2.5$, respectively. Here we choose $\tau = 0.01$. In Table I, we list some numerical results for this example. The first row in this table gives the values of ω. The second row gives the corresponding surface areas for the stable states (the input mesh has area 66.11). The third row contains the required iteration numbers for arriving at the stable states. The fourth and fifth rows contain the maximal and mean errors between the initial mesh and the final meshes. The maximal and mean error are defined as $\max_i \|x_i(0) - x_i(t)\|$ and $(1/m)\sum_i \|x_i(0) - x_i(t)\|$, respectively, where m is the number of vertices. The stable states are recognized by checking the residual in solving the linear system (26). If the residual of the initial solution (which is the numerical solution of the PDE at the previous timestep) is within our error control bound (we take it to be $9 * 10^{-6}$), then we regard the evolution as at the stable stage. The numerical results in the table show that if $\omega \to \infty$, the surface areas are increasing and tend to the initial surface area, the required iteration numbers, the maximal errors, and mean errors are decreasing and approach zero. The results also show that if ω is less than a certain number, say $\omega \leq 1.5$, the surface will shrink numerically to zero.

Finally, we summarize, in Table II, the computation time needed by our examples. The third column is the time (in seconds) for forming the stiffness matrix (one timestep). The fourth column is the required number of iterations for solving the linear systems by the Gauss–Seidel method. The last column is the average time per Gauss–Seidel iteration. We separate the total time into two parts, because the cost for generating the matrix is fixed, and the time for solving the linear system depends greatly on the solver used. These computations were conducted on a SGI Onyx2, using a single processor.

ACKNOWLEDGMENT

The authors greatly thank Dr. Jos Stam for providing us with the eigenstructures of the Loop subdivision matrix, Bob Holt for careful reading of this article, and Lei Xu for creating the texture images.

REFERENCES

ACTION, S. T. 1998. Multigrid anisotropic diffusion. *IEEE Trans. Image Process. 7*, 3, 280–291.

ARDEN, G. 2001. Approximation properties of subdivision surfaces. PhD Thesis, University of Washington.

BANSCH, E. AND MIKULA, K. 2001. Adaptivity in 3D image processing. Manuscript.

BERTALMIO, M., CHENG, L. T., AND OSHER, S. 2000a. Variational problems and partial differential equations on implicit surfaces. CAM Rep. 00-23, UCLA, Mathematics Department.

BERTALMIO, M., SAPIRO, G., CHENG, L. T., AND OSHER S. 2000b. A framework for solving surface partial differential equations for computer graphics applications. CAM Rep. 00-43, UCLA, Mathematics Department.

CATMULL, E. AND CLARK, J. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput. Aided Des. 10*, 6, 350–355.

CHAN, T. AND SHEN, J. 1999. Variational restoration of non-flat images features: Models and algorithms. CAM Rep. 99-20, UCLA, Mathematics Department.

CHOPP, D. L. AND SETHIAN, J. A. 1999. Motion by intrinsic Laplacian of curvature. *Interfaces Free Bound. 1*, 1–18.

CIRAK, F., ORTIZ, M., AND SCHRODER, P. 2000. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Int. J. Numer. Meth. Eng. 47*, 2039–2072.

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *Proceedings of Viz2000, IEEE Visualization*, (Salt Lake City, Utah), 397–505.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000a. Implicit fairing of irregular meshes using diffusion and curvature flow. *SIGGRAPH99*, 317–324.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000b. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Proceedings of Graphics Interface'2000*, 145–152.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000c. Discrete differential-geometry operators in nD. Available at http://www.multires.caltech.edu/pubs/.

DO CARMO, M. 1992. *Riemannian Geometry*. Boston.

DOO, D. AND SABIN, M. 1978. Behaviours of recursive division surfaces near extraordinary points. *Comput. Aided Des. 10*, 6, 356–360.

DYN, N., LEVIN, D., AND GREGORY, J. A. 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph. 9*, 2 (April), 160–169.

GREINER. G. 1994. Variational design and fairing of spline surface. *Comput. Graph. Forum 13*, 143–154.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Mutiresolution signal processing for meshes. In *SIGGRAPH '99 Proceedings*, 325–334.

HARR ROMENY, E. B. 1994. *Geometry Driven Diffusion in Computer Vision*. Kluwer Academic, Boston.

HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEND, M., JIN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. 1994. *Piecewise smooth surfaces reconstruction*. In *Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH94*, 295–302.

HOPPE, H., DEROSE, T., DUCHAMPI, T., MCDONALD, J., AND STUETZLE, W. 1993. *Mesh Optimization*. In *Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH9xi34*, 19–26.

HUBELI, A. AND GROSS, M. 2000. Fairing of non-manifolds for visualization. In *Proceedings of Viz2000, IEEE Visualization*, (Salt Lake City, Utah), 407–414.

KAWOHL, B. AND KUTEV, N. 1998. Maximum amd comparison principle for one-dimensional anisotropic diffusion. *Math. Ann. 311*, 1, 107–123.

KIMMEL, R. 1997. Intrinsic scale space for images on surfaces: The geodesic curvature flow. *Graph. Models Image Process. 59*, 5, 365–372.

KIMMEL, R. AND SOCHEN, N. 1999. Geometric-variational approach for color image enhancement and segmentation. In *Scale-Space Theories in Computer Vision*, Lecture Notes in Computer Science, vol. 1682, M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, Eds. Springer, Berlin.

KIMMEL, R., MALLADI, R., AND SOCHEN, N. 1998. Image processing via the Beltrami operator. In *Proceedings of the third Asian Conference on Computer Vision* (Hong Kong, Jan. 8–11).

KOBBELT, L. 1996. Discrete fairing. In *The Mathematics of Surfaces VII*, T. Goodman and R. Martin, Eds., Information Geometers, 101–129.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. *SIGGRAPH98*, 105–114.

KOBBELT, L., HESSE, T., PRAUTZSCH, H., AND SCHWEIZERHOF, K. 1997. Iterative mesh generation for FE-computation on free form surfaces. *Eng. Comput. 14*, 806–820.

LOOP, C. T. 1978. Smooth subdivision surfaces based on triangles. Master's Thesis. Tech. Rep. Department of Mathematics, University of Utah.

MALLET, J. L. 1992. Discrete smooth interpolation in geometric modelling. *Comput. Aided Des. 24*, 4, 178–191.

MORETON, H. AND SEQUIN, C. 1992. Functional optimization for fair surface design. *ACM Comput. Graph.*, 409–420.

OSHER, S. J. AND FEDKIW, R. P. 2000. Level set methods. CAM Rep. 00-07, UCLA, Mathematics Department.

PERONA, P. AND MALIK, J. 1987. Scale space and edge detection using anisotropic diffusion. In *IEEE Computer Society Workshop on Computer Vision*.

PETERS, J. AND REIF, U. 1997. The simplest subdivision scheme for smoothing polyhedra. *ACM Trans. Graph. 16*, 4, 420–431.

PREUßER, T. AND RUMPF, M. 1999. An adaptive finite element method for large scale image processing. *Scale-Space Theor. Comput. Vis.* 232–234.

ROSENBERG, S. 1997. *The Laplacian on a Riemannian Manifold*. Cambridge University Press, New York.

SABIN, M. 1976. The use of piecewise form of numerical representation of shape. PhD Thesis, Hungarian Academy of Science, Budapest.

SAPIDIS, N. 1994. *Designing Fair Curves and Surfaces*. SIAM, Philadelphia.

SAPIRO, G. 2001. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, New York.

SAPIRO, G. AND RINGACH, D. L. 1998. Anisotropic diffusion on multivalued images with applications to color filtering. *IEEE Trans. Image Process. 5*, 11, 1582–1586.

SCHWEITZER, J. E. 1996. Analysis and application of subdivision surfaces. PhD Thesis, Department of Computer Science and Engineering, University of Washington, Seattle.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, New York.

SOCHEN, N., KIMMEL, R., AND MALLADI, R. 1998. A general framework for low level vision. *IEEE Trans. Image Process. 7*, 3, 310–318.

STAM, J. 1998. Fast evaluation of Loop triangular subdivision surfaces at arbitrary parameter values. In *SIGGRAPH '98 Proceedings, CD-ROM supplement*.

TANG, B., SAPIRO, G., AND CASELLES, V. 2000a. Color image enhancement via chromaticity diffusion. *Int. J. of Comput. Vis. 10*, 5, 701–707.

TANG, B., SAPIRO, G., AND CASELLES, V. 2000b. Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case. *Int. J. Comput. Vis. 36*, 2, 149–161.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *SIGGRAPH '95 Proceedings*, 351–358.

WEICKERT, J. 1996. Foundations and applications of nonlinear anisotropic diffusion filtering. *Z. Angew. Math. Mech. 76*, 283–286.

WEICKERT, J. 1997. Coherence-enhancing diffusion of colour images. In *Proceedings of the VII National Symposium on Pattern and Image Analysis*, vol. 1 (April), 239–244.

WEICKERT, J. 1998. *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart.

WELCH, W. AND WITKIN, A. 1992. Variational surface modeling. *Comput. Graph. 26*, 157–166.

WELCH, W. AND WITKIN, A. 1994. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94 Proceedings*, *28* (July), 247–256.

WESTERMANN, R., JOHNSON, C., AND ERTL, T. 2000. A level-set method for flow visualization. In *Proceedings of Viz2000, IEEE Visualization* (Salt Lake City, Utah), 147–154.

WHITAKER, R. AND BREEN, D. 1998. Level set models for the deformation of solid objects. In *Proceedings of the Third International Workshop on Implicit Surfaces, Eurographics Association* (June), 19–35.

WILLMORE, T. J. 1982. *Total Curvature in Riemannian Geometry*. Ellis Horwood Limited, UK.

WILLMORE, T. J. 1993. *Riemannian Geometry*, Clarendon, New York.

XU, G. AND BAJAJ, C. 2002. Curvature computations of 2-manifold in $I\!R^k$. *J. Comput. Math.* (to appear).

YEZZI, A., JR. 1998. Modified curvature motion for image smoothing and enhancement. *IEEE Trans. Image Process. 7*, 3, 345–352.

ZHAO, H. K., OSHER, S., AND FEDKIW, R. 2001. Fast surface reconstruction using the level set method. CAM Rep. 01-01, UCLA, Mathematics Department.

ZHAO, H. K., OSHER, S., MERRIMAN, B., AND KANG, M. 2000. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Comput. Vis. Image Understand 80*, 3, 295–319.

ZORIN, D. SCHRÖDER, P. AND SWELDENS, W. 1996. Subdivision for meshes with arbitrary topology. In *SIGGRAPH '96 Proceedings*, 71–78.

Received March 2001; revised January 2002; accepted June 2002