Anisotropic Diffusion of Noisy Surfaces and Noisy Functions on Surfaces

Chandrajit L. Bajaj [*]
Department of Computer Science,
University of Texas, Austin, TX 78712
Email: bajaj@cs.utexas.edu

Guoliang Xu [†]
State Key Laboratory of Scientific and Engineering Computing,
ICMSEC, Chinese Academy of Sciences, Beijing
Email: xuguo@lsec.cc.ac.cn

Abstract

We present a unified anisotropic geometric diffusion PDE model for smoothing (fairing) out noise both in triangulated 2-manifold surface meshes in $I\!\!R^3$ 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

Key words: Surface function diffusion; Loop's subdivision; Riemannian manifold, Texture Mapping.

1 Introduction

Problem Considered. Given a discretized triangular surface mesh $G_d \subset I\!\!R^3$ (geometric information) and a discretized function-vector $F_d \subset I\!\!R^{\kappa-3}$. 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 (non-discretized) representations for the smoothed geometry and

[*]Supported in part by NSF grants ACI-9982297, KDI-DMS-9873326, SANDIA/LLNL BD 4485-MOID

[†]Currently visiting the Center for Computational Visualization, UT, Austin, TX

surface function data. Our tertiary goal is to provide approaches for visualizing the smoothness of both the geometric and physical information during the smoothing process. In this paper, we use terms faring and smoothing interchangeably.

Motivation Quite often, discretized surfaces under investigation suffer from noise or errors in geometry (see Fig 1.1). 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 surfaces and function on surfaces (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 from the inaccuracy in spatial distribution of the mesh density (topology) and the quantization of the numerical vertex coordinate data.

The errors of the geometric data and surface function data may often be coherent. For example, if the surface function data comes from the numerical solution of some physical phenomena over a domain, the errors in the geometric data certainly cause errors in the solution. In such a case, it might be rational to combine the geometry and surface function data together, 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 grey-scale 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 hyper-surface in $\mathbb{R}^4$, then the surface is given by the graph $(x, y, z, f(x,y,z))$ for $(x,y,z) \in G$. In most cases, when the surface geometry and function on surface data errors are not coherent, the smoothing is 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 obtains a minimization problem that minimizes certain objective functions [10, 12, 20, 25, 31], such as thin plate energy, membrane energy [16], total curvature [17, 32], or sum of distances [19]. Using local interpolation or fitting, or replacing differential operators with divided difference operators, the minimization problems are discretized to arrive at finite dimensional linear or nonlinear systems. Approximate solutions are then obtained by solving the systems.

The main idea of evolution is borrowed from the solution of the linear heat conduction equation $\partial_t \rho - \Delta \rho = 0$ for equilibrating spatial variation in concentration, where $\Delta := \mathrm{div}\nabla$ is the Laplace operator. This PDE (partial differential equation) based evolution technique was originally transplanted to image processing (see [21, 22, 30]. In [30], 453 relevant references are listed) from the area of numerical solution of PDE. This was extended to smoothing or fairing noisy surfaces (see [3, 4, 6]). For surfaces, the counterpart of the Laplacian Δ is the Laplace-Beltrami operator Δ_M (see [7]). One then obtains the geometric diffusion equation

$$\partial_t x - \Delta_M x = 0 \tag{1.1}$$

for surface point $x(t)$ on the surface $M(t)$.

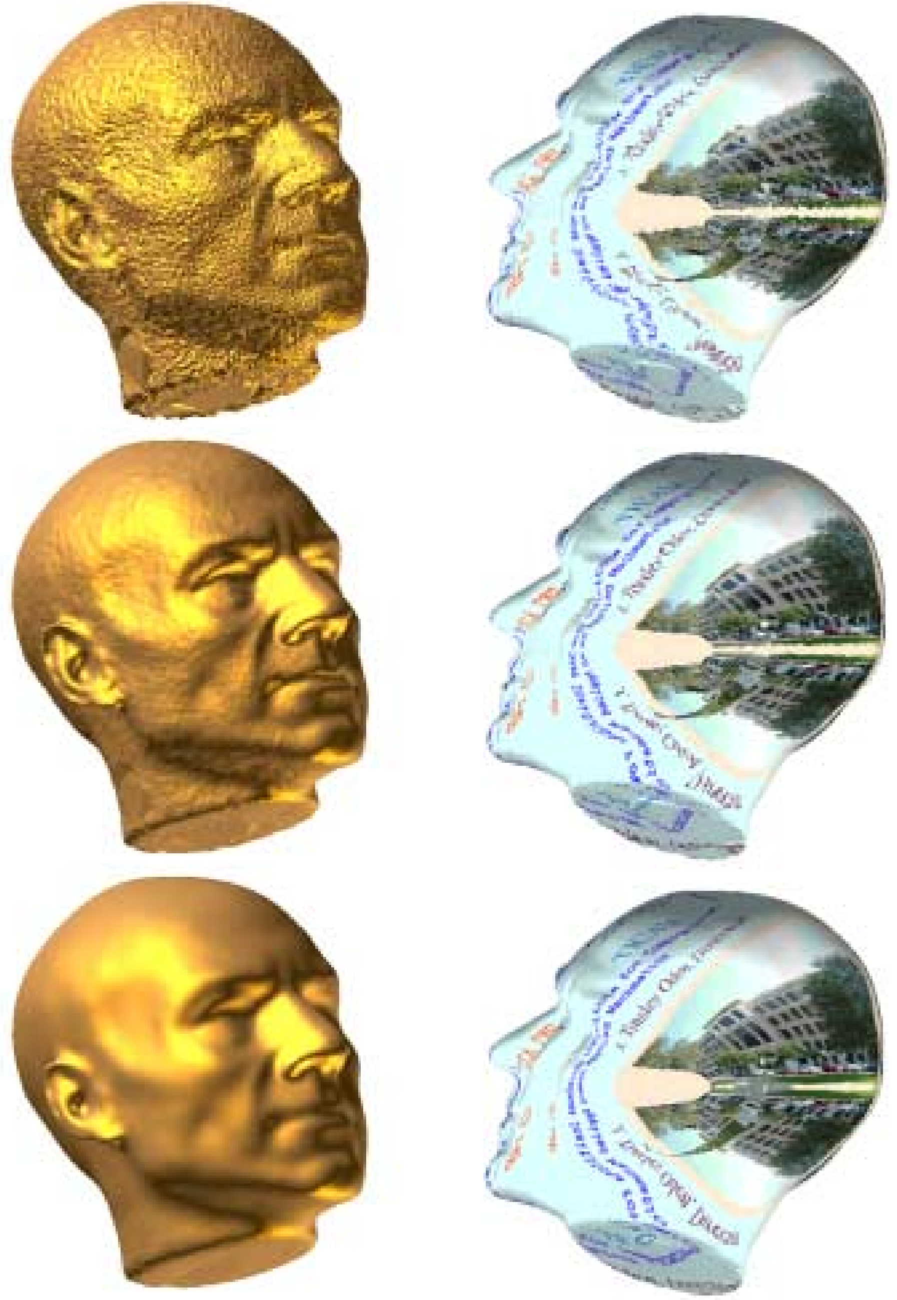

Fig 1.1: First column: Fairing the geometry of the head model of Picard(146,036 triangles). The second and third figures in this column are the meshes after 1 and 4 steps of fairing. Second column: Fairing texture coordinates while the geometry is fixed. The second and third figures of this are the fairing results after 1 and 4 iterations. In all the examples in this paper, the timestep τ is 0.001.

Taubin [29] discussed the discretized operator of the Laplacian and related approaches in the context of generalized frequencies on meshes. Kobbelt [15] considered discrete approximations of the Laplacian in the construction of fair interpolatory subdivision schemes. This work was extended [16] to arbitrary connectivity for purposes of multiresolution interactive editing. Desbrun et al. [4] use an implicit discretization of geometric diffusion to obtain a strongly stable numerical smoothing scheme. Clarenz et al. [3] introduced anisotropic geometric diffusion to enhance features while smoothing. All these are based on a discretized surface model. Hence, the first and second order derivative information, such as normals, tangents and curvatures, are 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 in [6]. 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 which simulate different kinds of flows of surface (see [33] for references) using the equation $\partial_t x - v(x,t) = 0$, where $v(x,t)$ represents the instantaneous stationary velocity field.

In 2D image processing, Sochen [27] and Yezzi [13] treated images as high-dimensional surfaces and processed them based on projected curvature motion flows. A similar treatment was adopted by Desbrun et al [5] for denoising bivariate data embedded in high dimensional spaces while preserving the edges. Curvature flows were also used in [26] (Chapter 16) for image enhancement and noise removal

For fairing functions on surfaces, Kimmel [14] 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 smoothed independently. For example, the signal processing approach for meshes proposed in [11] has been used to smooth the coordinates of texture mapping. In this paper we provide a new approach when vector-function data on a surface is treated simultaneously, both together and independently of the surface data.

Our Approach and Contributions.

a. *Establishing a unified diffusion model.* In this paper, we simply call a triangular surface mesh with function values on each of the vertices of the mesh an *attributed triangular mesh*. We treat 3-dimensional discrete surface data and $(\kappa-3)$-dimensional function data on the surface as a discretized version of a 2-dimensional Riemannian manifold embedded in $\mathbb{R}^\kappa$. We establish a PDE diffusion model for such a manifold. Though the derivation of the model involves Riemannian geometry, the outcome we obtained is simple and easy to understand.

b. *Discretizing in a smooth function space.* We combine the limit function representation of Loop's subdivision for triangular meshes with an established 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 are obtained. What makes our discretization distinct from previous work is we are smoothing *globally smooth functions* instead of *discrete functions*. Working with a smooth function model of finite dimension (instead of linear elements), related quantities, such as gradients, tangents, normals and curvatures, can be computed exactly and naturally from the smooth function representation. Hence our current framework is more accurate.

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 is the same the one given in [3]. The second column in Fig 1.2 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 base 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. 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. The second row of Fig 1.2 provides an example to illustrate this difference. Both of the figures start from the same noisy input (the top-left figure) and a fairing of three steps (timestep 0.001) is applied with the identity diffusion tensors. The left figure, which is the result of linear finite element implementation, smoothes out more detailed features (see the ears, eyes, lips and nose) than the right, which is the result our approach, and at the same time the large scale features (see the head) of the left are less smooth than that of the right. It should be pointed out that the larger support of basis functions leads to more nonzero (five times more in 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, the test results show that the condition of the discretized linear system of our approach is often better than that of the linear element approach. For the example we mentioned above, our approach needs 23, 18, 16 and 17 iterations for solving the linear systems by the Gauss Seidel methods for the time steps $1, \cdots, 4$, within the L_∞ error $9 * 10^{-6}$. The linear element approach needs $57, 67, 73$ and 77 iterations, respectively. This is understandable. Since the support of the basis functions of the linear element is small, the tiny triangles will cause very small elements in the matrix of the discretized linear system, which worsens the condition of the system. Such a problem is relatively moderate in our approach.

The evolution process produces not only a sequence of attributed triangular meshes at different time steps, 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 computed easily.

2 The Diffusion Model

The diffusion model that we are going to use is a generalization of the heat equation $\partial_t \rho - \Delta \rho = 0$ in Euclidean space to a 2-dimensional manifold embedded in $I\!R^\kappa$. Such a generalization to 3D surface has been given by Clarenz et al [3]. The generalization to a 2-dimensional manifold embedded in $I\!R^\kappa$ is similar. First, we establish the

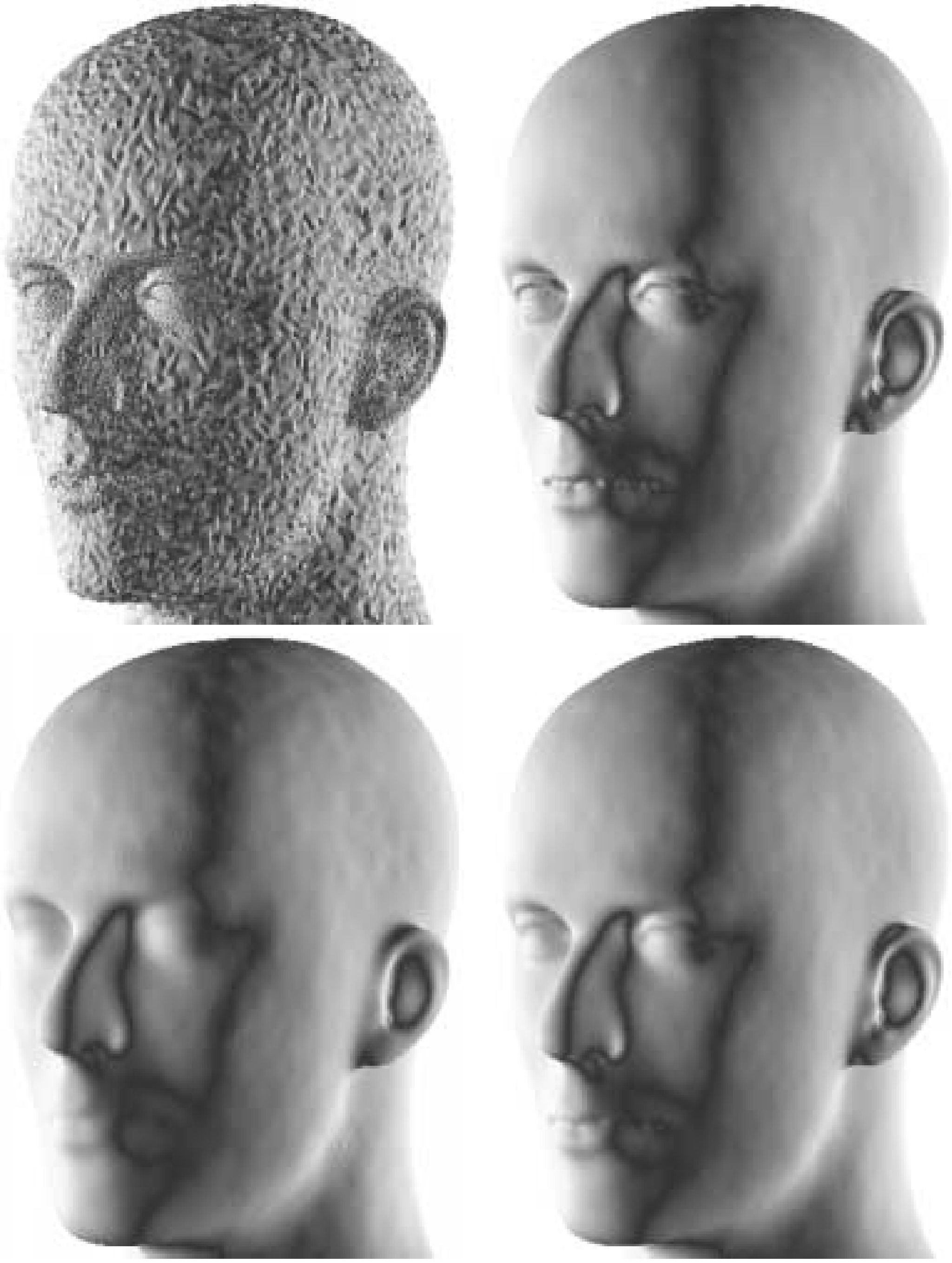

Fig 1.2: The first figure in the first row is the initial geometry mesh. The second figure is the fairing result after 3 iteration steps of our implementation with time length $t = 0.001$ and with an anisotropic diffusion tensor to preserve the sharp features around the eyes, nose, mouth and ear. The left and right figures in the second row are the fairing result by the linear finite element implementation and our approach, respectively, after 3 iteration steps with time length $t = 0.001$, and with an identity diffusion tensors.

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 then discussed in §4. Suppose we are given $\kappa-3$ ($\kappa \geq 3$) functions $f(x) = (f_1(x), f_2(x), \cdots, f_{\kappa-3}(x)) \in F$, $x \in G$. We assume that surface G is a two dimensional manifold embedded in $\mathbb{R}^3$. We will combine the geometric position x and function $f(x)$ together 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, some concepts, such as tangents, gradients, Laplacian, curvatures and integrations, that are well understood for surface, must be defined properly. Fortunately, these ideas are already very well developed in the field of *Riemannian Geometry* (see [7, 23, 34]). In the following, we shall borrow the required terminologies and concepts from that field and reformulate them to fit our diffusion problem.

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), then the tangent space T_xM at $x \in M$ is spanned by $\{\frac{\partial}{\partial \xi_1}, \frac{\partial}{\partial \xi_2}\}$. For a given point $x \in x_\alpha(U_\alpha) \subset M$, the tangent vector components $\frac{\partial}{\partial \xi_1}$ and $\frac{\partial}{\partial \xi_2}$ 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*:

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

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

$$\int_M f dx := \sum_\alpha \int_{U_\alpha} \phi_\alpha\ f(x_\alpha)\sqrt{det(g_{ij})} d\xi_1\ d\xi_2, \tag{2.1}$$

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

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

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

$$t_i^T \nabla_M f = \frac{\partial (f \circ x)}{\partial \xi_i}, \quad i = 1, 2, \tag{2.2}$$

where $t_i = \frac{\partial x}{\partial \xi_i}$ are the tangent vectors. Note that $\nabla_M f$ is invariant under the surface local reparameterization. From (2.2), we have

$$\nabla_M f = [\ t_1,\ t_2\ ] G^{-1} \left[\ \frac{\partial(f \circ x)}{\partial \xi_1},\ \frac{\partial(f \circ x)}{\partial \xi_2}\ \right]^T, \tag{2.3}$$

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 [23]):

$$(\mathrm{div}_M v, \phi)_M = -(v, \nabla_M \phi)_{TM}, \quad \forall \phi \in C_0^\infty(M), \tag{2.4}$$

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

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

$$\partial_t x(t) - \Delta_{M(t)} x(t) = 0, \tag{2.5}$$

where $\Delta_{M(t)} = \mathrm{div} \circ \nabla_{M(t)}$ is known as the Laplace-Beltrami operator on $M(t)$. However, to be able to enhance sharp features, a *diffusion tensor* D, acting on the gradient, is introduced. Hence the final model we use is

$$\partial_t x(t) - \mathrm{div}_{M(t)}(D \nabla_{M(t)} x(t)) = 0, \tag{2.6}$$

$$M(0) = M, \tag{2.7}$$

where $M(t)$ is the solution manifold at time t, $x(t)$ is a point on the manifold, and 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. If $D(x) = I$, an identity operator, then (2.6) becomes $\partial_t x(t) = 2H(x)$, since $\Delta_M x = 2H(x)$ (see [35], page 151), where $H(x)$ is the mean curvature vector at x. Hence the equation described is the mean curvature motion (MCM). The mean curvature motion has a displacement in the mean curvature vector direction, but not in the tangent direction. If $D(x)$ is not an identity operator, tangential displacement occurs. The details of the discussion for choosing the diffusion tensor are in §5. Using (2.4), the diffusion problem (2.6)-(2.7) can be reformulated as the following variational form

$$\begin{cases} \text{Find a smooth } x(t) \text{ such that} \\ (\partial_t x(t), \theta)_{M(t)} + (D \nabla_{M(t)} x(t), \nabla_{M(t)} \theta)_{TM(t)} = 0, \ \forall \theta \in C_0^\infty(M(t)) \\ M(0) = M. \end{cases} \tag{2.8}$$

Other Alternatives of the Diffusion Model. In establishing the diffusion model, we have combined the geometry and physics together. 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 five possibilities: **(a)**. *Both the data have errors and the errors are coherent.* **(b)**. *Both the data have errors and the errors are not coherent.* **(c)**. *Only the physical data has errors.* **(d)**. *Only the geometric data has errors.* **(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 $G(t) \subset I\!\!R^3$ and $F(t) \subset I\!\!R^{\kappa-3}$ denote the geometry and the physics information at time t, respectively. Then (2.8) becomes the following two problems:

$$\begin{cases} \text{Find a smooth } g(t) \in I\!\!R^3 \text{ such that} \\ (\partial_t g(t), \theta)_{G(t)} + (D\nabla_{G(t)} g(t), \nabla_{G(t)}\theta)_{TG(t)} = 0, \quad \forall \theta \in C_0^\infty(G(t)), \\ G(0) = G, \end{cases} \tag{2.9}$$

and

$$\begin{cases} \text{Find a smooth } f(t) \in I\!\!R^{\kappa-3} \text{ such that} \\ (\partial_t f(t), \theta)_{G(t)} + (D\nabla_{G(t)} f(t), \nabla_{G(t)}\theta)_{TG(t)} = 0, \quad \forall \theta \in C_0^\infty(G(t)), \\ F(0) = F, \end{cases} \tag{2.10}$$

where $G(t)$ is the solution of (2.9) at time t. Case **(e)** does not need to be considered. In case **(c)**, we separate the geometry and physics. We use the notation $G = G(t)$ to denote the geometry, and again use $F(t) \subset I\!\!R^{\kappa-3}$ to denote the physics information. Then (2.8) becomes

$$\begin{cases} \text{Find a smooth } f(t) \in I\!\!R^{\kappa-3} \text{ such that} \\ (\partial_t f(t), \theta)_G + (D\nabla_G f(t), \nabla_G\theta)_{TG} = 0, \quad \forall \theta \in C_0^\infty(G), \\ F(0) = F, \end{cases} \tag{2.11}$$

where $f(t) \in F(t)$ is the function of $F(t)$. Since G is fixed, the system (2.11) is linear. In case **(d)**, we need only to solve problem (2.9).

3 Subdivision Surfaces

We shall discretize the proposed diffusion problem in a function space which is defined by the limit of Loop's subdivision. This section describes only the relevant results on surface subdivision. It will be clear soon that these results are valid on the subdivision of functions defined on surfaces.

Subdivision schemes generate smooth surfaces via a limit procedure of an iterative refinement starting from an initial mesh which serves as the control mesh of the limit surface. Several subdivision schemes for generating smooth surfaces have been proposed. Some of them are interpolatory, i.e., the vertex positions of the coarse mesh are fixed, while only the newly added vertex positions need to be computed (see e.g., [17] for quadrilateral meshes, [9, 36] for triangular meshes), while others are approximating (see e.g., [2, 8] for quadrilateral meshes, [18] for triangular meshes). Approximating schemes compute both the old and new vertex positions. Generally speaking, approximating schemes produce better quality surfaces than those produced by interpolatory schemes. Hence, in this work, we shall use an approximating scheme for triangular meshes proposed by Loop [18]. This scheme produces C^2 limit surfaces except at a finite number of isolated points where the surface is C^1.

The limit surfaces of a subdivision scheme are defined by an infinite iteration procedure. There is no close form for the limit surface in general. This makes the exact evaluation of the surface at any point difficult. Fortunately, for Loop's scheme, a fast method exists for evaluating the limit surface (see [28]). For the purpose of numerically computing the area-integration, evaluation at any surface point is required. This is another reason for choosing Loop's scheme.

3.1 Loop's Subdivision Scheme

In Loop's subdivision scheme, the initial control mesh and the subsequent refined meshes consist of triangles only. In the refinement, each triangle is subdivided linearly into 4 sub-triangles. Then the vertex position of the refined mesh is computed as the weighted average of the vertex position of the unrefined mesh. Consider a vertex x_0^k at level k with neighbor vertices x_i^k for $i = 1, \cdots, n$ (see Fig 3.1), 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

$$x_i^{k+1} = \frac{3x_0^k + 3x_i^k + x_{i-1}^k + x_{i+1}^k}{8}, \quad i = 1, \cdots, n, \tag{3.1}$$

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

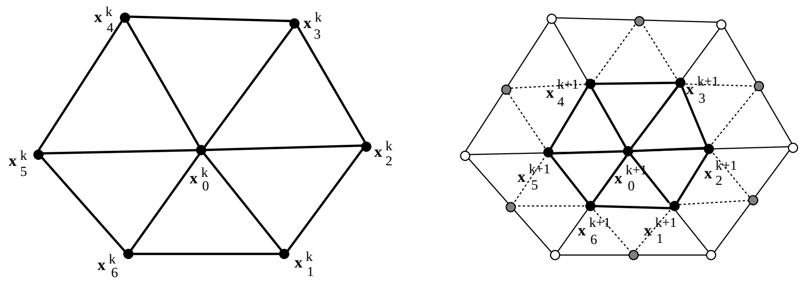

Fig 3.1: Refinement of triangular mesh around a vertex.

according to

$$x_0^{k+1} = (1 - na)x_0^k + a\left(x_1^k + x_2^k + \cdots + x_n^k\right), \tag{3.2}$$

where $a = \frac{1}{n}\left[\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}cos\frac{2\pi}{n}\right)^2\right]$. Note that all newly generated vertices have a valence of 6, while the vertices inherited from the original mesh at level zero may have a valence other than 6. We will refer to the former case as *ordinary* and to the later case as *extraordinary*.

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

$$T = \{(\xi_1, \xi_2) \in \mathbb{R}^2 : \xi_1 \geq 0, \xi_2 \geq 0, \xi_1 + \xi_2 \leq 1\}. \tag{3.3}$$

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 Fig 3.2). 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. We refer to such a patch as *regular*. A regular patch is controlled by 12 basis functions:

$$x(\xi_1, \xi_2) = \sum_{i=1}^{12} N_i(\xi_1, \xi_2) x_i, \tag{3.4}$$

where the label i refers to the local numbering of the vertices that is shown in Fig 3.2. The surface within the shaded triangle in this figure is defined by the 12 local control vertices. The basis N_i are given as follows (see [28]):

$$\begin{aligned}
N_1 &= \tfrac{1}{12}(\xi_0^4 + 2\xi_0^3\xi_1), \\
N_2 &= \tfrac{1}{12}(\xi_0^4 + 2\xi_0^3\xi_2), \\
N_3 &= \tfrac{1}{12}\left[\xi_0^4 + \xi_1^4 + 6\xi_0^3\xi_1 + 6\xi_0\xi_1^3 + 12\xi_0^2\xi_1^2 + (2\xi_0^3 + 2\xi_1^3 + 6\xi_0^2\xi_1 + 6\xi_0\xi_1^2)\xi_2\right], \\
N_4 &= \tfrac{1}{12}[6\xi_0^4 + 24\xi_0^3(\xi_1 + \xi_2) + \xi_0^2(24\xi_1^2 + 60\xi_1\xi_2 + 24\xi_2^2) \\
&\quad + \xi_0(8\xi_1^3 + 36\xi_1^2\xi_2 + 36\xi_1\xi_2^2 + 8\xi_2^3) + (\xi_1^4 + 6\xi_1^3\xi_2 + 12\xi_1^2\xi_2^2 + 6\xi_1\xi_2^3 + \xi_2^4)],
\end{aligned} \tag{3.5}$$

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 bases 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, i.e., at least one of its vertices has a valence other than 6, the resulting patch is not a quartic box spline. We assume extraordinary vertices are isolated, i.e., 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 evaluation of irregular patches, we use the scheme proposed by Stam [28]. 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 each subdivision of an irregular patch produces three regular patches and one irregular patch (see Fig 3.3). Repeated subdivision of the irregular patch will produce a sequence of regular patches. The surface patch is

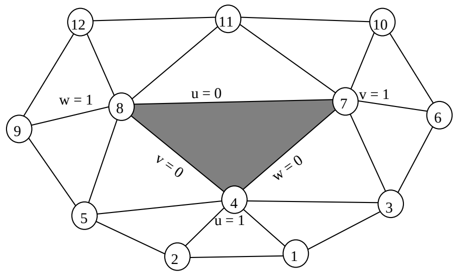

Fig 3.2: The vertex numbering of a regular patch with 12 control points. Over the shaded triangle, the regular patch is defined.

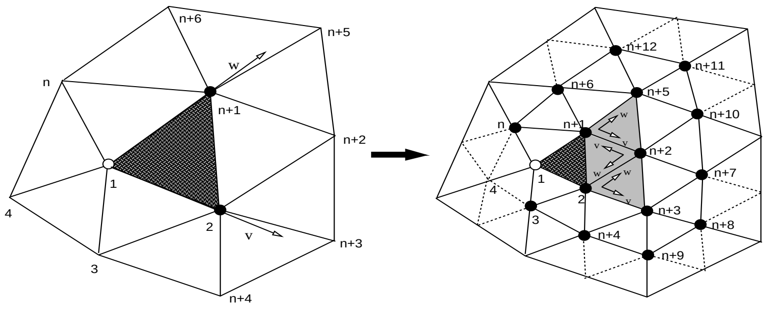

Fig 3.3: The vertex with empty circle is extraordinary. After one subdivision, the irregular patch (dark shaded part) is split into one irregular patch (dark shaded part) and three regular patches (light shaded parts).

piecewise parameterized. The subdomains T_j^k are given as follows:

$$\begin{aligned}
T_1^k &= \{(\xi_1,\xi_2) : \xi_1 \in [2^{-k}, 2^{-k+1}], &\quad& \xi_2 \in [0, 2^{-k+1} - \xi_1]\},\\
T_2^k &= \{(\xi_1,\xi_2) : \xi_1 \in [0, 2^{-k}], && \xi_2 \in [2^{-k} - \xi_1, 2^{-k}]\},\\
T_3^k &= \{(\xi_1,\xi_2) : \xi_1 \in [0, 2^{-k}], && \xi_2 \in [2^{-k}, 2^{-k+1} - \xi_1]\}.
\end{aligned} \tag{3.6}$$

These subdomains are mapped onto T by the transform

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

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

$$x(\xi_1,\xi_2)|_{T_j^k} = \sum_{i=1}^{12} x_i^{k,j} N_i(t_{k,j}(\xi_1,\xi_2)), \quad j = 1,2,3; \quad k = 1,2,\cdots, \tag{3.7}$$

where $x_i^{k,j}$ are the properly chosen 12 control vertices around the irregular patch at the level k that define a regular surface patch. Using the vertex numbering and local coordinate system shown in Fig 3.3, it is easy to see that the three set control vertices are

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

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, i.e.,

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

where $X^k = [x_1^k, \cdots, x_{n+6}^k]^T$, $\tilde{X}^k = [x_1^k, \cdots, x_{n+6}^k, x_{n+7}^k, \cdots, x_{n+12}^k]^T$, and A and $\tilde{A}$ are defined by the subdivision rule. 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 amount 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 [28] for details.

4 Discretization

In Riemannian geometry, differentiable functions are smooth and C^∞. However, our discretized version of the diffusion problem will be 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 $M = \bigcup_{\alpha=1}^k \mathcal{T}_\alpha$, $\overset{\circ}{\mathcal{T}}_\alpha \cap \overset{\circ}{\mathcal{T}}_\beta = \emptyset$ for $\alpha \neq \beta$, where $\overset{\circ}{\mathcal{T}}_\alpha$ is the interior of the *triangular function patch* $\mathcal{T}_\alpha$. Each triangular function patch is assumed to be parameterized locally as

$$x_\alpha : \quad T \to \mathcal{T}_\alpha; \quad (\xi_1, \xi_2) \;\mapsto\; x_\alpha(\xi_1, \xi_2). \tag{4.1}$$

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 (2.1) is replaced by

$$\int_M f dx := \sum_\alpha \int_T f(x_\alpha(\xi_1, \xi_2)) \sqrt{det(g_{ij})} d\xi_1 \; d\xi_2. \tag{4.2}$$

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

4.1 Spatial Discretization

Let M_d^l be the limit function of the initial control mesh M_d. Then, instead of solving the problem (2.8), we solve the following alternative problem

$$\begin{cases} \text{Find } \; x(t) \in V_{M(t)}^k, \quad \text{such that} \\ (\partial_t x(t), \theta)_{M(t)} + (D\nabla_{M(t)} x(t), \nabla_{M(t)} \theta)_{TM(t)} = 0, \quad \forall \theta \in V_{M(t)} \\ M(0) = M_d^l, \end{cases} \tag{4.3}$$

where $V_{M(t)} \subset C^1(M(t))$ is a finite dimensional space spanned by the basis functions $\{\phi_i(x)\}_{i=1}^m$. $\phi_i(x)$ is defined by the limit of the Loop's subdivision for the zero control values everywhere except at x_i where it is one. Hence the support of $\phi_i(x)$ is local and it covers the 2-ring neighborhood of vertex x_i. Let e_j, $j = 1, \cdots, m_i$ be the 2-ring neighborhood elements. Then if e_j is regular, the explicit box-spline expression as in (3.4) exists for $\phi_i(x)$ on e_j. Using (3.5), we could derive the BB-form coefficients for base ϕ_i (see Fig. 4.1.b). All these coefficients have a factor $\frac{1}{24}$. Hence, the function value at x_i is $\frac{1}{2}$. Note that the base ϕ_i derived is the same as the triangular C^2 quartic base given by Sabin (see [24]). These expressions could be used to evaluate $\phi_i(x)$ in forming the linear system (4.6). If e_i is irregular, local subdivision, as described in §3.3, is needed around e_i until the parameter values of interest are interior to a regular patch. Let $x(t) = \sum_{i=1}^m x_i(t)\phi_i(x)$, $x_i(t) \in I\!R^\kappa$,

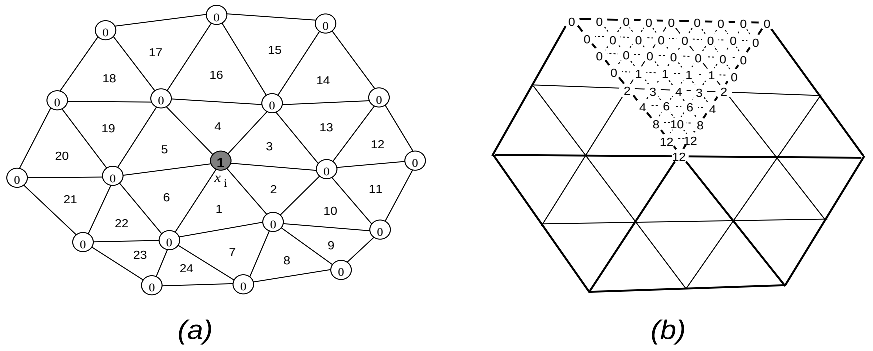

Fig 4.1: The quartic Bézier coefficients (each has a factor 1/24) of basis function. The coefficients on the other five macro-triangles are obtained by rotating the top macro-triangle around the center to the other five positions.

and $\theta = \phi_j(x)$. Then (4.3) may be written as

$$\begin{cases} \sum_{i=1}^m x_i'(t)\,(\phi_i(x), \phi_j(x))_{M(t)} + \\ \sum_{i=1}^m x_i(t)(D\nabla_{M(t)}\phi_i(x), \nabla_{M(t)}\phi_j(x))_{TM(t)} = 0, \\ x_j(0) = x_j, \end{cases} \tag{4.4}$$

for $j = 1, \cdots, m$, where x_j is the j-th vertex of the initial mesh M_d. (4.4) is a set of nonlinear ordinary equations for the unknown functions $x_i(t)$, $i = 1, \cdots, m$.

4.2 Time Discretization

Given a time step $\tau > 0$, suppose we have an approximate solution at $t = n\tau$. Now we want to get approximate solution at the next time step $t = (n+1)\tau$ by the semi-implicit Euler scheme. Let X^n be approximation of $x(n\tau)$. Then the semi-implicit discretization of (4.4) is

$$\left(X^{n+1} - X^n, \phi_i\right)_{M(n\tau)} + \tau \left(D^n \nabla_{M(n\tau)} X^{n+1}, \nabla_{M(n\tau)} \phi_i\right)_{TM(n\tau)} = 0, \qquad (4.5)$$

for $i = 1, \cdots, m$. Let $x(t) = \sum_{i=1}^m x_i(t)\phi_i(x)$. Then (4.5) can be written as a linear system:

$$(M^n + \tau L^n(D^n))X((n+1)\tau) = M^n X(n\tau), \qquad (4.6)$$

where $X(t) = [x_1(t), \cdots, x_m(t)]^T$, $X(0) = [x_1, \cdots, x_m]^T$,

$$M^n = \left((\phi_i, \phi_j)_{M(n\tau)}\right)_{i,j=1}^m,$$
$$L^n(D^n)) = \left((D^n \nabla_{M(n\tau)} \phi_i, \nabla_{M(n\tau)} \phi_j)_{TM(n\tau)}\right)_{i,j=1}^m.$$

Note that both M^n and $L^n(D^n)$ are symmetric. Since $\phi_1, \phi_2, \cdots, \phi_m$ 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, $M^n + \tau L^n(D^n)$ is symmetric and positive definite.

The coefficient matrix of the system (4.6) is highly sparse. An iterative method for solving such a system is desirable. We solve it by Gauss Seidel iteration if the time step $\tau < 350/N$, and otherwise by the 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 (4.6) is valid for solving problems (2.8)–(2.11), though it is derived for (2.8) 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 (2.9), and the last $\kappa - 3$ columns are the solution of (2.10) and (2.11). For all the cases we mentioned in §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 (2.11), 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 a mean curvature motion, we can determine T by examining the reduction rate of the mean curvature. For a given mesh, which part is noise that should be smoothed out is subjective. The information at time t is not enough to judge whether the smoothing is satisfactory. Therefore, we always compare the evolution effect with 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 shall use the derivative (see (4.8)) of $\mathcal{H}(t)$ to test the stable state of the evolution. If the data is 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 (4.8) works well. However, the derivative sometimes can not 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 fast for quite a long time and then slows down. Using the derivative of $\mathcal{H}(t)$ in such a case will lead to a late termination, which makes the mesh over-smoothed (the features are lost). In this case, we prefer to use $\mathcal{H}(t)$ itself to control the termination (see (4.9)). If the data is very noisy (high frequency noise), $\mathcal{H}(t)$ reduces fast 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 (4.7)). Of course, there are an infinite number of cases between these extremes. These considerations make us choose the following three stopping criteria.

$$|\mathcal{H}'(t)/\mathcal{H}'(0)| \le \epsilon_1, \quad \text{or} \tag{4.7}$$

$$|\mathcal{H}'(t)| \le \epsilon_2, \quad \text{or} \tag{4.8}$$

$$\mathcal{H}(t) \le \epsilon_3, \tag{4.9}$$

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 is very noisy, condition (4.7) is most likely satisfied first. If the data is smooth, and the shape is simple, condition (4.8) is most likely satisfied first. The remaining case may first be satisfied by condition (4.9).

Choice of Timestep τ. Suppose the final result we want is at time T. This time moment can be approached through several time steps. Though 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 surface as $x_1(t), \cdots, x_k(t)$. Then from (4.3), we have

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

Since D is positive definite, we have

$$\frac{\partial (x(t), x(t))_{M(t)}}{\partial t} = 2(\partial_t x(t), x(t))_{M(t)} = -2\sum_{i=1}^{k} E_i(t) \le 0, \tag{4.10}$$

where $E_i(t) = (D\nabla_{M(t)}x_i(t), \nabla_{M(t)}x_i(t))_{TM(t)} \ge 0$, 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 (2.3), that $E(t) = 2\text{Area}(M(t))$.

From (4.10), we know that if $k = 3$, the surface size $S(M(t))$ decreases in the 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 one percent of the $\text{Area}(M(t))$, then we should choose $\tau = 0.0025$ approximately. Such an amount of change in size is significant visually. In our experiment, even $\tau = 0.0001$ the change is still visually significant at the early stage of the evolution. But at later stages, the change becomes increasingly less significant. Usually, we choose τ around 0.001 and determine the number n of iterations by $n\tau \approx T$.

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 concepts of *principal curvatures* and *principal directions* of a 2-manifold $M \subset I\!R^{\kappa}$. Let n be a normal vector field on M. Let $\mathcal{A}_n$ be the *second fundamental tensor* with respect to n (see [35] pages 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 is 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 $k > 3$. We will choose a vector field $h = H(x)/\|H(x)\|$, 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 (6.2). Considering the diffusion equation described is the mean curvature motion, choosing this vector field is natural.

We will now illustrate how to compute the principal curvatures and principal directions with respect to h. Due the space limitation, the detailed derivations are given in [1]. Let $t_i = \frac{\partial}{\partial \xi_i}$, $t_{ij} = \frac{\partial^2}{\partial \xi_i \partial \xi_j}$, $[\tilde{e}_1, \tilde{e}_2] = [t_1, t_2]W$ and

$$W = \begin{bmatrix} g_{11}^{-\frac{1}{2}} & -g_{12}[g_{11}\det(G)]^{-\frac{1}{2}} \\ 0 & g_{11}[g_{11}\det(G)]^{-\frac{1}{2}} \end{bmatrix}, \quad A_h = -W^T[t_1, t_2]^T \left[\frac{\partial h}{\partial \xi_1} \frac{\partial h}{\partial \xi_2}\right] W. \quad (5.1)$$

Then $\tilde{e}_1$ and $\tilde{e}_2$ are orthonormal and A_h, a symmetric 2×2 matrix, is the matrix representation of $\mathcal{A}_n$. Let $A_h = S \ \mathrm{diag}[k_1, k_2] \ S^T$, with $S^T S = I$, then k_1 and k_2 are the principal curvatures and e_1 and e_2, defined by $[e_1, e_2] = [\tilde{e}_1, \tilde{e}_2]S$, are the corresponding principal directions of $\mathcal{A}_h$. To give an explicit expression for A_h, let $g_{ijk} = t_i^T t_{jk}$, $g_{ijkl} = t_{ij}^T t_{kl}$, then we can derive that

$$A_h = W^T \tilde{A}_h W / (2\det(G)\|H(x)\|),$$

where $\tilde{A}_h$ is 2×2 symmetric matrix defined by

$$\begin{aligned} \tilde{A}_h &= [A_1 u, A_2 u] - g_{22}A_{11} - g_{11}A_{22} + 2g_{12}A_{12} \\ A_1 &= \begin{bmatrix} g_{111} & g_{211} \\ g_{112} & g_{212} \end{bmatrix}, \quad A_2 = \begin{bmatrix} g_{112} & g_{212} \\ g_{122} & g_{222} \end{bmatrix}, \\ u &= G^{-1}\left\{A_1^T[g_{22}, \ -g_{12}]^T + A_2^T[-g_{12}, \ g_{11}]^T\right\}, \end{aligned}$$

$A_{kl} = (g_{ijkl})_{ij=1}^2$, $(k, l) = (1, 1), (2, 2), (1, 2)$.

Now we can define our anisotropic diffusion tensor. Let $\kappa_{\epsilon,1}$, $\kappa_{\epsilon,2}$ 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

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

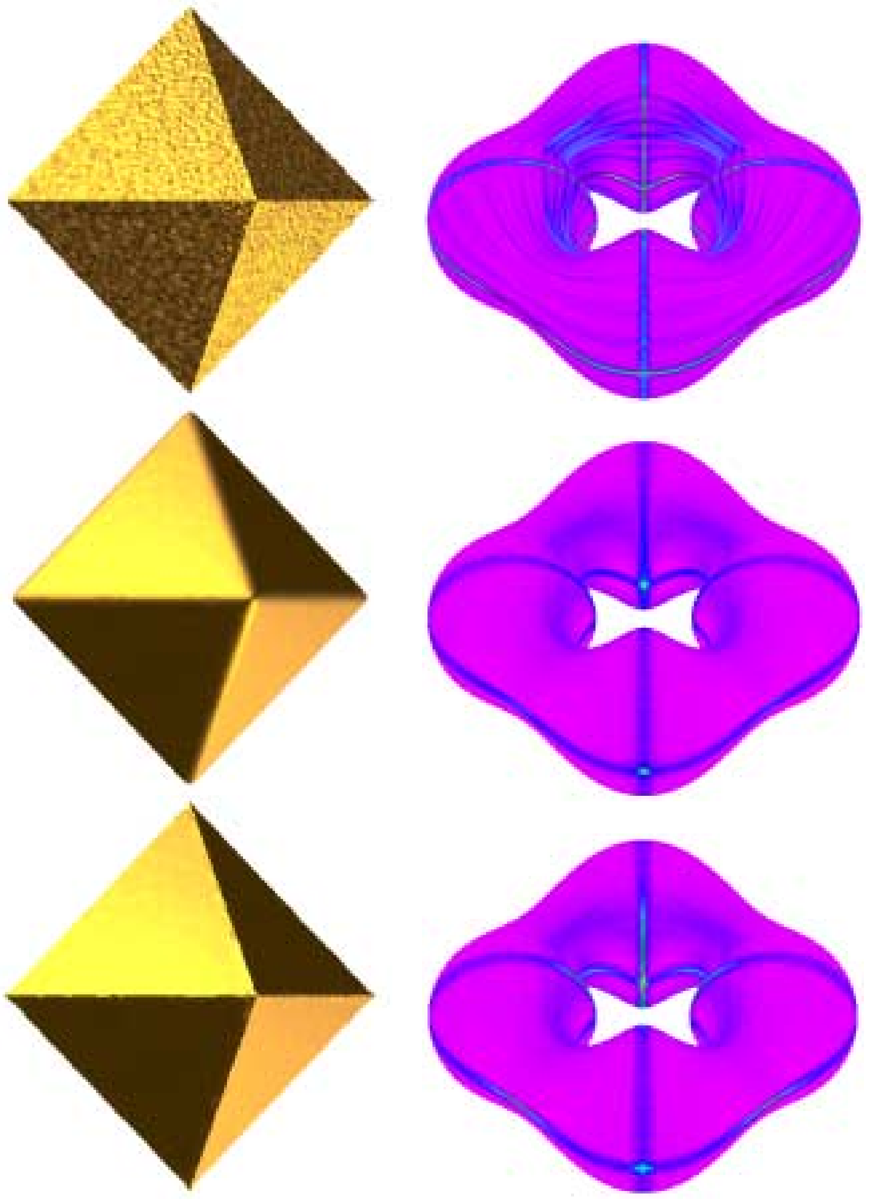

Fig 5.1: The first column: The first figure is a bumpy input mesh (32,786 triangles). The second and third figures are the faired meshes after 4 fairing iterations, with the identity and an anisotropic diffusion tensor, respectively. For the anisotropic diffusion tensor, we choose $\lambda = 2$, $\epsilon = 0.001$. The 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 4 fairing iterations, with identity and an anisotropic diffusion tensor ($\lambda = 2.5$, $\epsilon = 0.001$), respectively.

where $g(s)$ is defined by

$$g(s) = \begin{cases} 1; & s \leq \lambda, \\ \left(1 + \frac{(s-\lambda)^2}{\lambda^2}\right)^{-1}; & s > \lambda, \end{cases} \tag{5.2}$$

$M_\epsilon(t)$ is the solution of (2.8) at time ϵ with $D = 1$ and initial value $M(t)$. λ is a given parameter which detects the sharp feature. The reason we use $M_\epsilon(t)$, instead of $M(t)$, to compute the diffusion tensor is 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 the mean curvature motion before we evaluate the shape parameters. Fig 5.1 shows the effect of the anisotropic smoothing.

6 Smoothness Visualization

6.1 Iso-Contour Plot

For a function $f(x)$ defined on a smooth surface M, a *iso-contour* or *iso-curve* is defined as $\{x \in M : f(x) = c\}$ for a given constant c, which is called the *iso-value.* The smoothness of the iso-contours could reflect 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 iso-contours for a given sequence of iso-values $\{c_i\}_{i=1}^n$. In our problem, we have a function vector $x(t) \in I\!R^\kappa$ instead of one scalar function. One way to visualize them together is to plot iso-contours for $\|x(t)\|^2$. Fig 6.1 shows both the geometry (the first column) and the function (the second column) diffusion effects. In this example and the other examples in this section, a scalar function value at vertex $p_i = (x_i, y_i, z_i)$ of the given mesh is specified as $x_i^2 + |y_i| + sin(z_i) + \epsilon_i$, where $\epsilon_i = (-1)^i 0.3 d_i/(i(mod\ 5) + 1.0)$ are regarded as the noise with the d_i as the average distance of the one-ring neighbor vertex positions to p_i.

6.2 Riemannian Curvature Plot

It is well known the curvature at any point is a good measurement of the speed of motion of a surface, in its normal direction and away from its tangent plane. The counterpart of Gaussian curvature for a surface is *Riemannian curvature* for a Riemannian manifold (see [7, 34]). To visualize the smoothness of both the geometric and surface function data, the Riemannian curvature can be computed and coded into color. Let $t_i = \frac{\partial}{\partial \xi_i}$, $t_{ij} = \frac{\partial^2}{\partial \xi_i \partial \xi_j}$. Then we derived the following formula for Riemannian curvature (see [1] for detail):

$$K(x) = \frac{t_{12}^T T_{12} t_{12} - t_{11}^T T_{12} t_{22} + t_{11}^T t_{22} - t_{12}^T t_{12}}{\|t_1\|^2 \|t_2\|^2 - (t_1^T t_2)^2}, \tag{6.1}$$

where $T_{12} = [t_1, t_2] G^{-1} [t_1, t_2]^T$. It is easy to check that $K(x)$ coincide with the Gaussian curvature if $k = 3$. Fig 6.2 shows both the geometry diffusion effect (the first column) and the geometry & function diffusion effect (the last column, Riemannian curvature plot).

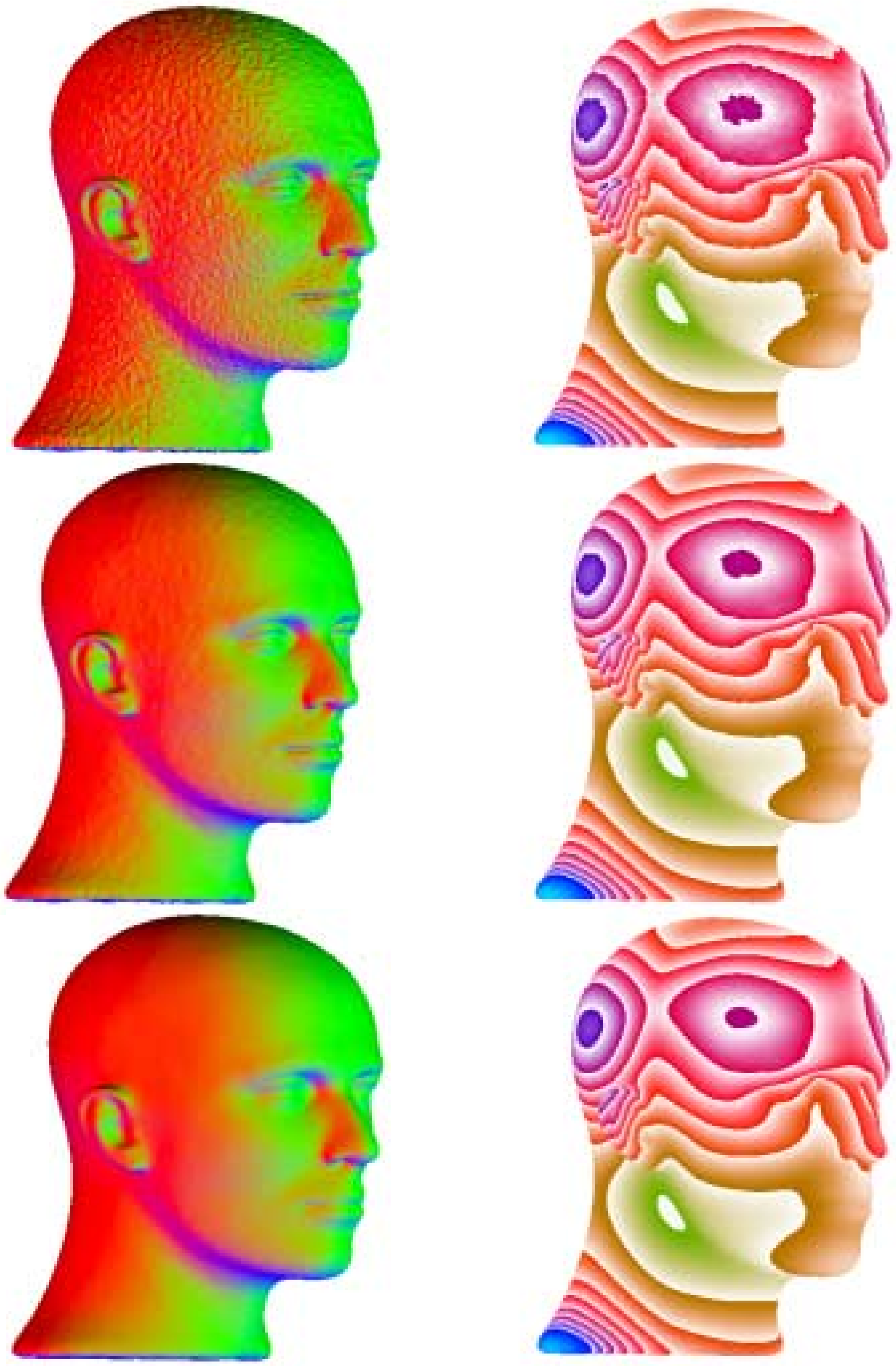

Fig 6.1: First column: The first figure is the input noisy geometry (102,208 triangles), the second and third figure are the geometry diffusion results after 1 and 4 fairing iterations, respectively. Second column: The iso-contour plots of the function $\|x\|^2$ on the smoothed head. The three figures show the results after 0, 1 and 4 fairing iterations, respectively.

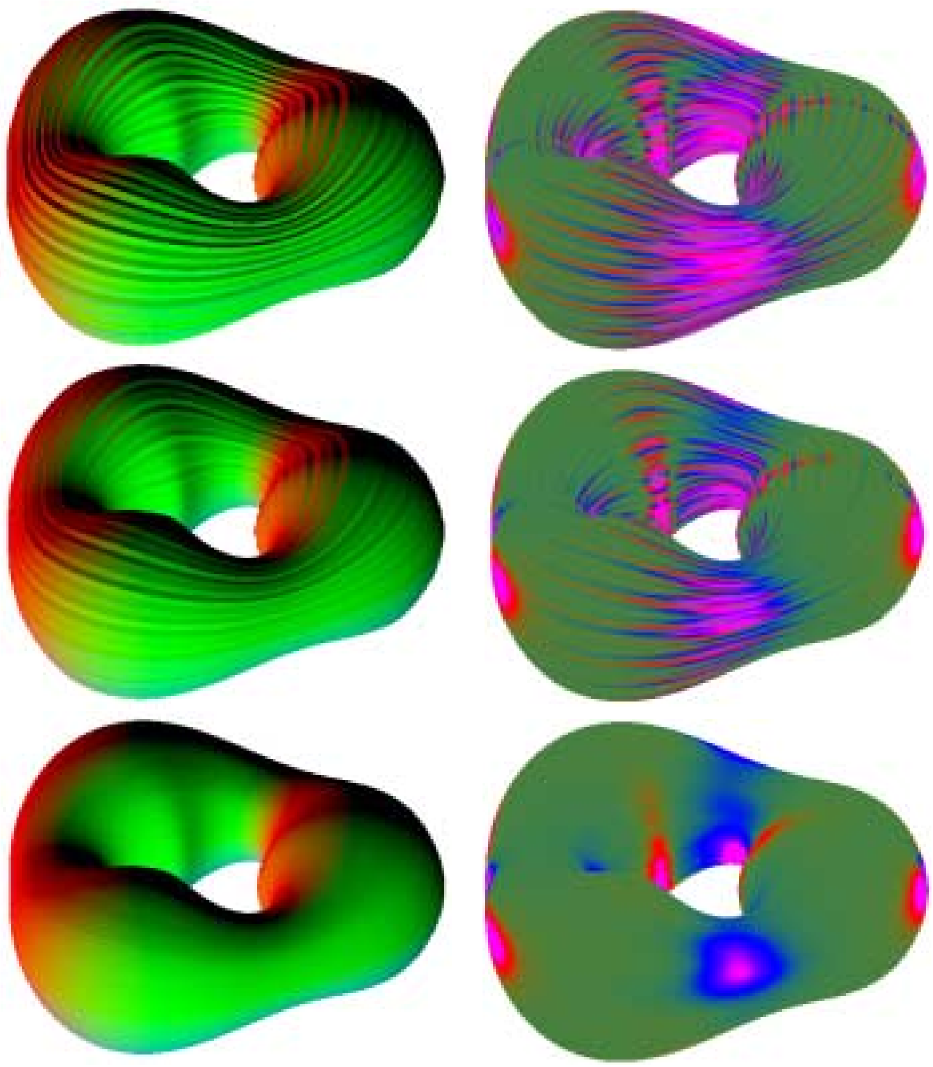

Fig 6.2: First column: The first figure is a noisy input geometry (25,600 triangles), the second and third are the geometry diffusion results after 1 and 6 fairing iterations. Second column: The first is the Riemannian curvature plots on the noisy surface with a given noisy function. The second and the third are the results of 1 and 9 fairing iterations for both the surface and the function on surface.

6.3 Mean Curvature Plot

It is known that $\Delta_M x = 2H(x)$ (see [35], page 151), where $H(x)$ is the mean curvature vector. The diffusion equation (2.5) describes the mean curvature flow. Hence, the mean curvature plot is the right choice for visualizing the smoothness of the data. For a 2-dimensional Riemannian submanifold M of $I\!R^{\kappa}$, the mean curvature vector is well defined (see [35] page 119). We arrive at the following simple form (see [1])

$$H(x) = \frac{[g_{22}t_{11} + g_{11}t_{22} - 2g_{12}t_{12}]^{\perp}}{2(g_{11}g_{22} - g_{12}^2)}, \tag{6.2}$$

where $[\,\cdot\,]^{\perp}$ denotes the vector component that is orthogonal to the tangent space (*the normal component of the vector*). Differing from the classical mean curvature for a surface, the mean curvature vector is a vector in the normal space. For $k = 3$, $H(x)^T n(x)$, which is the length (with a sign) of $H(x)$, is the classical mean curvature. Here $n(x)$ represents the unit normal of the surface at x. The figures in the second column of Fig 6.3 show the plot of $\|H\|$. In this example both the geometry and surface function data are smoothed.

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 the combination of the limit function representation of Loop's subdivision together with the diffusion model.

Additional examples given in Fig 7.1 show the application of the diffusion process to the surface texture maps (2D texture vector coordinates at each of the vertices of the surface triangulation). 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 net-like 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.

Finally, we summarize, in Table 7.1, the time consumed by some of the examples. The third column is the time (in seconds) for forming the stiffness matrix (one time step). 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 forming the matrix is fixed, while the time for solving the linear system depends greatly on the used solver. For the sake of comparison, how the cost relates to the number of triangles, the times are only for the evolution of the geometry. These computations were conducted on a SGI Onyx2, using a single processor.

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

References

[1] C. Bajaj and G. Xu. Curvatures Computations of 2-manifolf in $\mathbb{R}^k$, *Manuscript*, 2001.

[2] E. Catmull and J. Clark. Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes. *Computer Aidded Design*, 10(6):350–355, 1978.

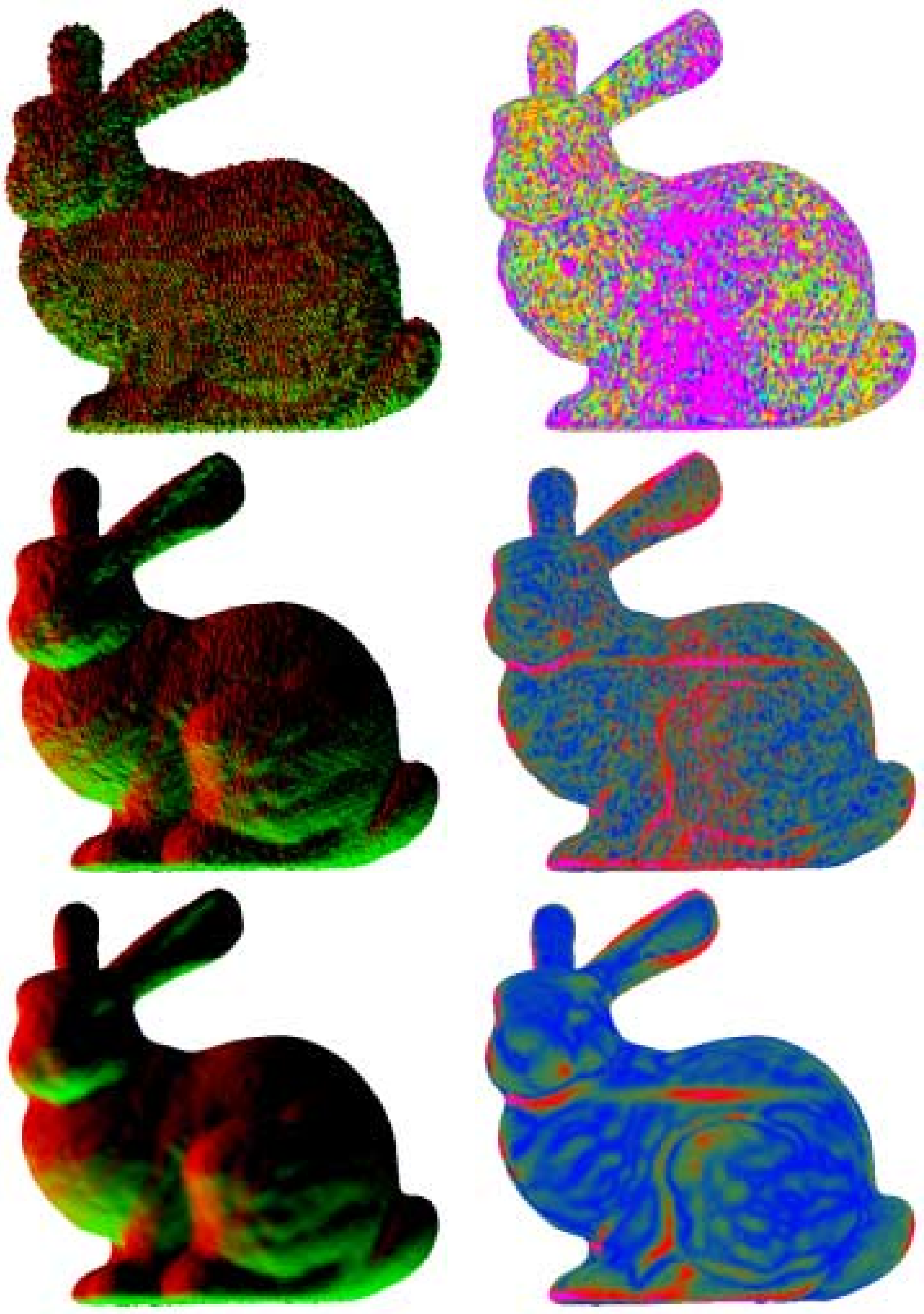

Fig 6.3: First column: The first figure is the input noisy geometry (69,473 triangles), the second and third figures show the surface diffusion results after 2 and 4 fairing iterations, respectively. Second column: The first is the mean curvature plots $\|H(x)\|$ for noisy surface and noisy function on surface. The second and third figures are the results of 2 and 4 fairing iterations for both the surface and the function on the surface.

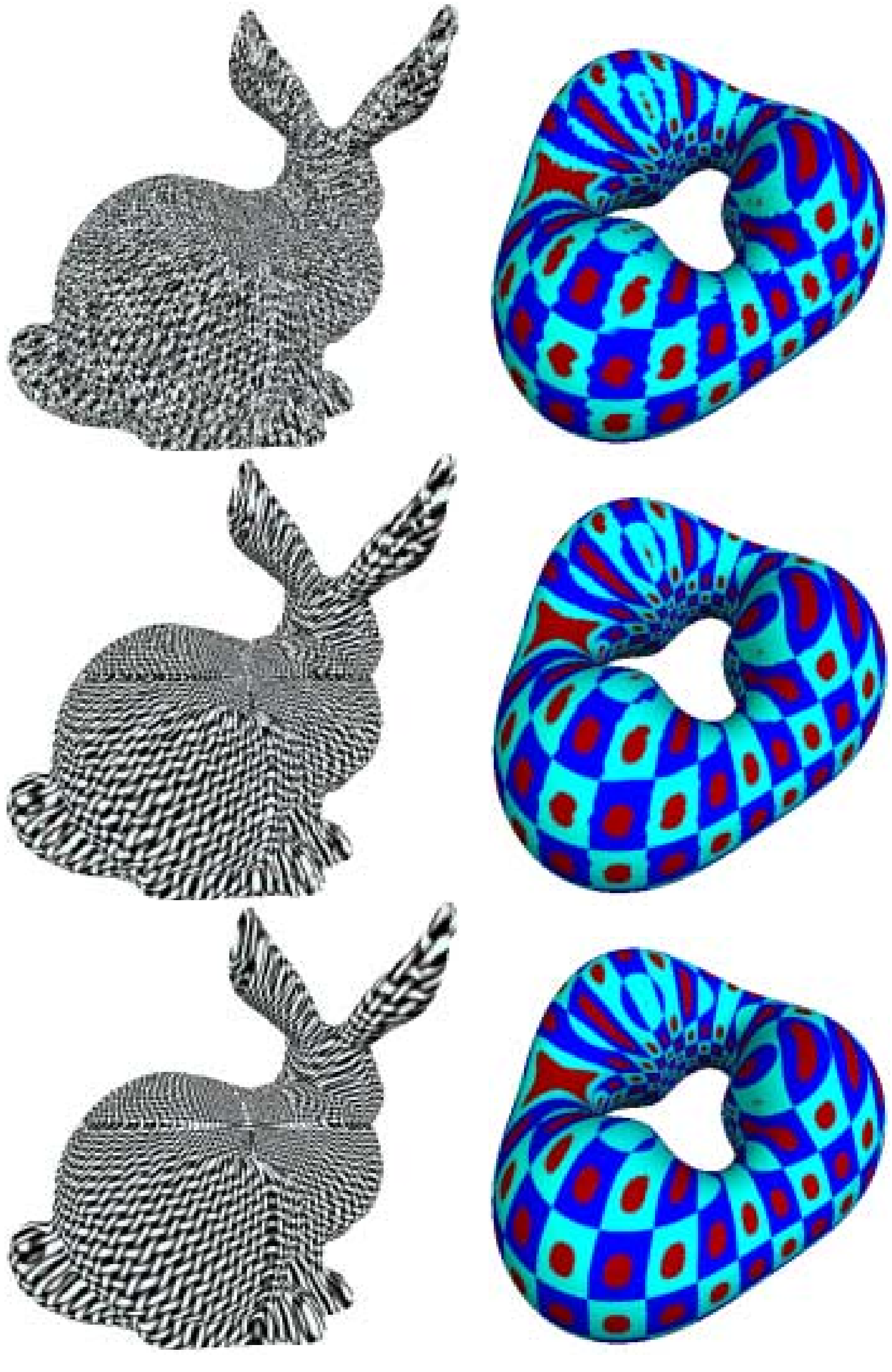

Fig 7.1: Each of the two columns show the diffusion of initial noisy texture maps data after 1 and 5 fairing iterations respectively.

Examples	Triangles #	Form matrix	Iterations #	Averaging
Fig 1.1(left)	146,036	24.8s	23, 24, 57, 86	0.964s
Fig 1.2(right)	102,208	12.2s	23, 18, 16, 17	0.202s
Fig 6.3(left)	69,473	8.2s	18, 14, 22, 26	0.107s
Fig 5.1(left)	32,786	3.5s	13, 12, 12, 12	0.021s
Fig 6.2(left)	25,600	2.7s	18, 14, 13, 12	0.019s

Table 7.1: Second column: number of triangles. Third column: times for computing the stiff matrix. Fourth column: number of iterations for solving the linear systems for the four time steps. Last column: average times (per iteration) for solving the linear systems.

[3] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic Geometric Diffusion in Surface Processing. In *Proceedings of Viz2000, IEEE Visualization*, pages 397–505, Salt Lake City, Utah, 2000.

[4] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. *SIGGRAPH99*, pages 317–324, 1999.

[5] M. Desbrun, M. Meyer, P. Schröder, and A. H Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Proc. Graphics Interface'2000*, pages 145–152, 2000.

[6] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Discrete Differential-Geometry Operators in nD, 2000.

[7] M. do Carmo. *Riemannian Geometry.* Boston, 1992.

[8] D. Doo and M. Sabin. Behavious of Recursive Division Surfaces near Extraordinary Points. *Computer Aidded Design*, 10(6):356–360, 1978.

[9] Nira Dyn, David Levin, and John A. Gregory. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.

[10] G. Greiner. Variational design and fairing of spline surface. *Computer Graphics Forum*, 13:143–154, 1994.

[11] I. Guskov, W. Sweldens, and P. Schröder. Mutiresolution signal processing for meshes. In *SIGGRAPH '99 Proceedings*, pages 325–334, 1999.

[12] A. Hubeli and M. Gross. Fairing Of Non-Manifolds For Visualization. In *Proceedings of Viz2000, IEEE Visualization*, pages 407–414, Salt Lake City, Utah, 2000.

[13] A. Yezzi Jr. Modified Curvature Motion for Image Smoothing and Enhancement. *IEEE Transections on Image Processing*, 7(3):345–352, 1998.

[14] R. Kimmel. Intrinsic scale space for images on surfaces: The geodesic curvature flow. *Graphical Models and Image Processing*, 59(5):365–372, 1997.

[15] L. Kobbelt. Discrete Fairing. In Tim Goodman and Ralph Martin, editors, The Mathematics of Surfaces VII, pages 101–129. Information Geometers, 1996.

[16] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive Muti-Resolution Modeling on Arbitrary Meshes. *SIGGRAPH98*, pages 105–114, 1998.

[17] L. Kobbelt, T. Hesse, H. Prautzsch, and K. Schweizerhof. Iterative Mesh Generation for FE-computation on Free Form Surfaces. *Engng. Comput.*, 14:806–820, 1997.

[18] Charles Loop. *Smooth subdivision surfaces based on triangles. Master's thesis.* Technical report, Department of Mathematices, University of Utah, 1978.

[19] J. L. Mallet. Discrete Smooth Interpolation in Geometric Modelling. *Computer Aided Design*, 24(4):178–191, 1992.

[20] H. Moreton and C. Sequin. Functional Optimization for Fair Surface Design. *ACM Computer Graphics*, pages 409 – 420, 1992.

[21] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In IEEE Computer Society Workshop on Computer Vision, 1987.

[22] T. Preußer and M. Rumpf. An adaptive finite element method for large scale image processing. In *Scale-Space Theories in Computer Vision*, pages 232–234, 1999.

[23] S. Rosenberg. *The Laplacian on a Riemannian Manifold.* Cambridge, Uviversity Press, 1997.

[24] M. Sabin. *The use of piecewise form of numerical representation of shape.* PhD thesis, Hungarian Academy of Science, Budapest, 1976.

[25] N. Sapidis. *Designing Fair Curves and Surfaces.* SIAM, Philadelphia, 1994.

[26] J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1999.

[27] N. Sochen, R. Kimmel, and R. Malladi. A General Framework for Low Level Vision. *IEEE Transections on Image Processing*, 7(3):310–318, 1998.

[28] J. Stam. Fast Evaluation of Loop Triangular Subdivision Surfaces at Arbitrary Parameter Values. In *SIGGRAPH '98 Proceedings, CD-ROM supplement*, 1998.

[29] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95 Proceedings*, pages 351–358, 1995.

[30] J. Weickert. *Anisotropic Diffusion in Image Processing.* B. G. Teubner Stuttgart, 1998.

[31] W. Welch and A. Witkin. Variational Surface Modeling. *Computer Graphics*, 26:157–166, 1992.

[32] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94 Proceedings*, volume 28, pages 247–256, July 1994.

[33] R. Westermann, C. Johnson, and T. Ertl. A Level-Set Method for Flow Visualization. In *Proceedings of Viz2000, IEEE Visualization*, pages 147–154, Salt Lake City, Utah, 2000.

[34] T. J. Willmore. *Total Curvature in Riemannian Geometry.* Ellis Horwood Limited, 1982.

[35] T. J. Willmore. *Riemannian Geometry.* Clarendon Press, 1993.

[36] D. Zorin, P. Schröder, and W. Sweldens. Subdivision for meshes with arbitrary topology. In *SIGGRAPH '96 Proceedings*, pages 71–78, 1996.