Diffusion Equations over Arbitrary Triangulated Surfaces for Filtering and Texture Applications

Chunlin Wu, Jiansong Deng, and Falai Chen

Abstract—In computer graphics, triangular mesh representations of surfaces have become very popular. Compared with parametric and implicit forms of surfaces, triangular mesh surfaces have many advantages such as being easy to render, being convenient to store, and having the ability to model geometric objects with arbitrary topology. In this paper, we are interested in data processing over triangular mesh surfaces through partial differential equations (PDEs). We study several diffusion equations over triangular mesh surfaces and present corresponding numerical schemes to solve them. Our methods work for triangular mesh surfaces with arbitrary geometry (the angles of each triangle are arbitrary) and topology (open meshes or closed meshes of arbitrary genus). Besides the flexibility, our methods are efficient due to the implicit/semi-implicit time discretization. We finally apply our methods to several filtering and texture applications such as image processing, texture generation, and regularization of harmonic maps over triangular mesh surfaces. The results demonstrate the flexibility and effectiveness of our methods.

Index Terms—Triangular mesh surface, diffusion equation, finite-volume method, image processing, texture generating, harmonic map.

1 Introduction

Triangular mesh surfaces have been widely used in computer graphics in the last three decades. Compared with parametric and implicit surfaces, mesh surfaces have many advantages such as being easy to render, being convenient to store, and having the ability to model geometric objects with arbitrary topology. There have been a huge volume of literature on the modeling and processing of mesh surfaces such as rendering [32], subdivision [15], [59], compression and simplification [47], [26], fairing [23] and editing [56], [2], parameterization, and texture mapping [43], [57]. However, there exists little work on data processing such as image processing and vector field processing over mesh surfaces.

In the past decade, there is much work on data processing over parametric and implicit surfaces. Among various techniques, methods based on partial differential equations (PDEs) have attracted much attention. The reason is that other techniques such as wavelet analysis are difficult to be generalized over surfaces. In 1997, Kimmel proposed scale-space concepts of images on parametric surfaces via PDEs and presented numerical methods to solve the problem [30]. Following the framework, Spira and Kimmel studied problems of image enhancement [44] and image segmentation [45] over parametric surfaces. Therein, PDEs are defined and numerically solved in parametric domains. In [10], the authors considered data processing over implicit surfaces. They studied several typical PDEs on implicit surfaces and applied them to several problems such as image denoising, harmonic map regularization, and pattern formation. In 2005, the authors of the present paper provided an image inpainting algorithm over implicit surfaces [54]. Different from PDEs over parametric surfaces, PDEs over implicit surfaces are numerically solved in the euclidean space in which the surfaces are embedded. For efficiency, PDEs are actually solved in a narrow band near the implicit surface, and before numerically solving the PDEs, the data need to be extended to this narrow band. For details about data extension, the reader is referred to [34], [54], and [53].

Based on previous work, there are obviously two ways to deal with the data processing problem over mesh surfaces: by converting the mesh surfaces into either parametric forms or implicit forms. The first approach is based on mesh parameterization. The mesh surface is parameterized first (globally or locally), and then, PDEs are solved in the parameter domain by numerical schemes proposed for parametric surfaces. The method used to generate textures in [52] and the flow simulation technique reported in [46] fall into this category. However, on one hand, the difficulty of surface global parameterization brings a fatal weakness since there is no global parameterization for most mesh surfaces at all. On the other hand, elaborate boundary handling of patches should be considered in piecewise-parameterization-based methods. Besides, to our knowledge, general surfaces cannot be parameterized without distortion. It should be pointed out that the method in [49] to solve PDEs over mesh surfaces is intrinsically based on parameterization. For the second approach, one can construct an implicit representation for the given mesh surface first [58] and then solve the problem over the implicit surface. Again, new difficulties appear. One needs to convert the data from mesh surfaces to implicit surfaces when constructing the implicit representation, and then data, extension is needed before data processing. This affects the efficiency of the whole process.

In this paper, we solve data processing problems via directly solving diffusion equations over mesh surfaces for filtering and texture applications. We use the finite-volume method (FVM) coupled with implicit/semi-implicit time integrals to discretize these equations. In fact, similar

• *The authors are with the Department of Mathematics, University of Science and Technology of China, Hefei 230026, P.R. China. E-mail: {wucl, dengjs, chenfl}@ustc.edu.cn.*

Manuscript received 29 Apr. 2007; revised 6 Sept. 2007; accepted 13 Dec. 2007; published online 9 Jan. 2008.
Recommended for acceptance by J. Stam.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2007-04-0046.
Digital Object Identifier no. 10.1109/TVCG.2008.10.

spatial discretization techniques have been developed in the scientific computing community where meshes are usually of simple topology and generally euclidean. These techniques include methods with finite-element flavor [13], covolume and mimetic methods for compatible discretizations of differential operators [1], generalized finite-difference methods (FDMs) [12], and FVM [8]. However, little work focuses on solving PDEs directly over mesh surfaces in 3D space—surfaces represented as polyhedrons—and only a few references catch our attention [24], [22], [42]. Du and Ju developed FVM schemes to solve the elliptic type of equations over mesh surfaces [24]. Desbrun et al. built a powerful toolkit, namely, Discrete Exterior Calculus (DEC), over simplicial complexes [22], [28] and applied it to fluid simulations successfully [25]. However, in DEC, simplices are required to be well centered [28]. In [42], the authors simulate fluid flows over mesh surfaces, where the modified FDM is adopted. Our FVM-based method is valid for arbitrary triangulated surfaces, including seriously irregular mesh surfaces (which appear frequently in the adaptive representation of mesh surfaces) and mesh surfaces of arbitrary topology. Besides, our methods benefit from implicit/semi-implicit time discretizations and thus are very effective. Furthermore (and most importantly), our contribution focuses on the applications of the technique in image processing and texture generation.

The paper is organized as follows: In Section 2, we introduce some concepts and notations. Then, several diffusion equations are proposed in Section 3, and their corresponding numerical methods are presented in Section 4. In Section 5, we discuss some applications of the PDEs in image processing, harmonic map regularization and texture generation. Conclusion and future work are included in Section 6.

2 Notations

Assume that M is a triangulated mesh surface with arbitrary geometry and topology in $\mathbb{R}^3$. The set of vertices, the set of edges, and the set of triangles of M are denoted as $\{v_i : i = 0, 1, \ldots, \mathrm{V}-1\}$, $\{e_i : i = 0, 1, \ldots, \mathrm{E}-1\}$, and $\{\tau_i : i = 0, 1, \ldots, \mathrm{T}-1\}$, respectively. Here, V is the number of vertices, E is the number of edges, and T is the number of triangles. We explicitly denote an edge e whose endpoints are p, q as $[p, q]$. Similarly, a triangle τ whose vertices are p, q, and r is denoted as $[p, q, r]$. If v is an endpoint of an edge e, then we denote it as $v \prec e$. Similarly, if e is an edge of a triangle τ, it is denoted as $e \prec \tau$; if v is a vertex of a triangle τ, it is denoted as $v \prec \tau$ [28]. For a given triangle τ, its barycenter and circumcenter are denoted by $BC(\tau)$ and $CC(\tau)$, respectively. Let $BC(e)$ be the barycenter of edge e. Obviously, the circumcenter $CC(e)$ of an edge e is just the barycenter of e. Similarly, we define the barycenter and circumcenter of a vertex v to be itself, that is, $BC(v) = CC(v) = v$. Let $N_1(i)$ be the 1-neighborhood of vertex v_i. It is the set of indices of vertices that are connected to v_i. Let $D_1(i)$ be the 1-disk of the vertex v_i. $D_1(i)$ is the set of triangles, with v_i being one of their vertices. It should be pointed out that the 1-disk of a boundary vertex is topologically just half of a disk.

For each vertex v_i, we define a piecewise linear function ϕ_i such that $\phi_i(v_j) = \delta_{ij}$, $i, j = 0, 1, \ldots, \mathrm{V}-1$, where δ_{ij} is the Kronecker delta. It is obvious that the support of ϕ_i is the 1-disk of the vertex v_i. Furthermore, $\{\phi_i : i = 0, 1, \ldots, \mathrm{V}-1\}$ has the following good properties: 1) nonnegativity, that is, $\phi_i \geq 0$, $i = 0, 1, \ldots, \mathrm{V}-1$, and 2) partition of unity, that is, $\sum_{0 \leq i \leq \mathrm{V}-1} \phi_i \equiv 1$. A function u defined over the triangulated surface M is considered to be a piecewise linear function. Suppose u has function value u_i at vertex v_i, $i = 0, 1, \ldots, \mathrm{V}-1$; then, $u(p) = \sum_{0 \leq i \leq \mathrm{V}-1} u_i \phi_i(p)$ for any $p \in M$. Similarly, we can define vector-valued functions $(u_1(p), u_2(p), \ldots, u_d(p))$ on M. For example, the normal vector of M is such a function. In some applications, the piecewise constant function over M is used, that is, a single value is assigned to each triangle of M [28].

3 Diffusion Equations on Triangulated Surfaces

In this section, we present several diffusion equations over triangulated mesh surfaces, including linear/nonlinear isotropic and anisotropic models. Assume that M is a triangulated mesh surface, and a piecewise linear function $u(p)$ is defined on M. We denote the gradient operator on M by ∇_M and the Laplace-Beltrami operator on M by $\triangle_M$. We assume that the initial function defined on M is $f(p)$. For open surfaces, boundary conditions are needed. If the mesh surface is closed, then the boundary condition is ignored automatically.

3.1 Linear Model

The first model is a linear PDE defined as follows:

$$\begin{cases} u_t = \triangle_M u, \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p, 0) = f(p), \end{cases} \tag{1}$$

where $\vec{n}$ is the intrinsic outer normal of the boundary of M lying on the tangent plane of M, and ∂M is the boundary of M. Here, the Neumann boundary condition is set. When the mesh surface M degenerates to a planar domain, the linear model is just the classical heat equation, which is widely applied in planar image processing, visualization of 2D vectors, etc.

3.2 Nonlinear Model

A general nonlinear PDE model is given as follows:

$$\begin{cases} u_t = \nabla_M \cdot (g(|\nabla_M u|)\nabla_M u), \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p, 0) = f(p), \end{cases} \tag{2}$$

where $g(\cdot)$ is a nonnegative function (usually monotonically descending). Specifically, by taking $g(s) = \frac{1}{\sqrt{s^2+\beta}}$, we have the following typical nonlinear model:

$$\begin{cases} u_t = \nabla_M \cdot \left(\frac{1}{\sqrt{|\nabla_M u|^2+\beta}} \nabla_M u \right), \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p, 0) = f(p), \end{cases} \tag{3}$$

where β is a small positive number introduced to avoid zero division. When the mesh surface M reduces to a planar domain, the above model degenerates to the TV model [40], [17], [20], which is very classical in planar image processing. We call (3) the *intrinsic TV model*.

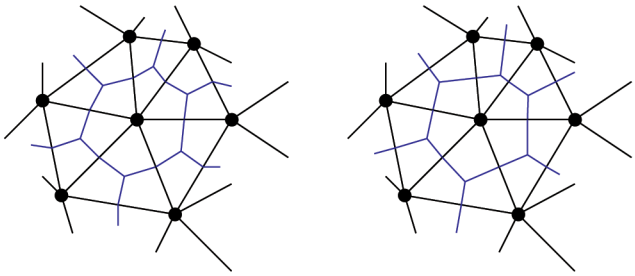

Fig. 1. A mesh and its dual mesh. (a) Barycentric dual. (b) Circumcentric dual.

3.3 Anisotropic Model

The linear and nonlinear models (1) and (2) are essentially isotropic, that is, diffusion rates along different directions are the same at a fixed point on M. In this section, we present a general anisotropic PDE model. Assume that $e_1(p)$ and $e_2(p)$ are two orthogonal piecewise constant vector fields in the tangent space of M. That is, $e_1(p)$ and $e_2(p)$ are constant vectors in each triangle of M, and they constitute an orthogonal basis of the underlying space of the triangle. Furthermore, we assume that $g_1(|\nabla_M u|)$ and $g_2(|\nabla_M u|)$ are two positive functions. The anisotropic model reads

$$\begin{cases} u_t = \nabla_M \cdot (g_1(\nabla_M u \cdot e_1)e_1 + g_2(\nabla_M u \cdot e_2)e_2), \\ (g_1(\nabla_M u \cdot e_1)e_1 + g_2(\nabla_M u \cdot e_2)e_2) \cdot \vec{n}|_{\partial M} = 0, \\ u(p,0) = f(p). \end{cases} \quad (4)$$

For different problems, appropriate functions g_1 and g_2 have to be chosen. Especially, if $g_1 = g_2$, the anisotropic model degenerates to an isotropic one. Furthermore, if $g_1 = g_2 = 1$, the model (4) degenerates to the linear model (1). From this point of view, the anisotropic model is the generalized form of isotropic models.

4 Numerical Methods

In this section, numerical schemes are proposed to solve the isotropic and anisotropic equations introduced in Section 3. We adopt implicit/semi-implicit FVM schemes, that is, implicit/semi-implicit time discretization coupled with FVM spatial discretization.

4.1 Dual Meshes

Dual meshes are widely used in computational electromagnetism [1] and Discrete Exterior Calculus [28], [22]. Fig. 1 shows two typical dual meshes. The original mesh consists of black lines, whereas the dual mesh is colored in blue. The dual mesh in Fig. 1a is *barycentric dual*, formed by connecting the barycenter and the middle point of each edge in each triangle. The dual mesh in Fig. 1b is *circumcentric dual*, formed by connecting the circumcenter and the middle point of each edge in each triangle. The circumcentric dual is in fact the Voronoi graph of the original mesh. Any mesh always has a barycentric dual, whereas its circumcentric dual may not exist. When a triangle is an obtuse triangle, its circumcenter lies outside of it. In this case, some numerical computation difficulties arise. A method to deal with this case can be found in [33] and [55]. For simplicity, we adopt the barycentric dual in this paper.

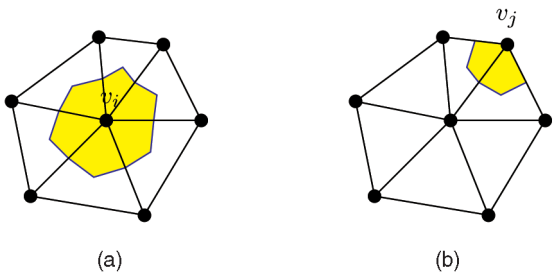

Fig. 2. Vertices and their control cells. (a) Barycentric dual: control cell of an inner vertex. (b) Barycentric dual: control cell of a boundary vertex.

4.2 Control Cell

Based on the concept of dual meshes, we assign a *control cell* C_i to each vertex v_i of mesh surface M. The concept of control cells was used in [33] for computing discrete geometry quantities. Fig. 2a shows the control cell C_i for an inner vertex v_i of the original mesh, whereas Fig. 2b shows the control cell for a boundary vertex. For the inner vertex v_i, the boundary of C_i is $\partial C_i = \bigcup_{\tau \in D_1(i)} \bigcup_{v_i \prec e \prec \tau} [BC(e), BC(\tau)]$. For the boundary vertex, the boundary of the control cell is

$$\partial C_j = \left(\bigcup_{\tau \in D_1(j)} \bigcup_{v_j \prec e \prec \tau} [BC(e), BC(\tau)] \right) \bigcup \left(\bigcup_{v_j \prec e \subseteq \partial M} [BC(v_j), BC(e)] \right).$$

Here, the orientations of intervals such as $[BC(e), BC(\tau)]$ should be taken in a sense with agreement to the clockwise or counterclockwise orientations of the boundaries of control cells.

4.3 Numerical Discretization

We discretize the PDEs via implicit/semi-implicit FVM schemes: implicit/semi-implicit time discretization and spatial discretization by integrating the PDEs over some control cells. Then, a highly sparse linear system of equations is obtained for each model. In the following, we describe the details. The reader may refer to [8] and the references therein for more knowledge about FVM.

4.3.1 Discretization of the Linear Model

For each vertex v_i of the mesh model, we integrate the two sides of (1) on the control cell C_i:

$$\int_{C_i} u_t dC_i = \int_{C_i} \triangle_M u dC_i. \quad (5)$$

According to the divergence theorem, the right-hand side of (5) is

$$\int_{\partial C_i} \nabla_M u \cdot \vec{n} dl,$$

where $\vec{n}$ is the intrinsic outer normal of ∂C_i on M. To get a more explicit expression of the above integral, we discuss it in two cases: v_i is an inner vertex or a boundary vertex.

If v_i is an inner vertex, the above integral becomes

$$\int_{\bigcup_{\tau\in D_1(i)}\bigcup_{v_i\prec e\prec\tau}[BC(e),BC(\tau)]} \nabla_M u\cdot\vec{n}dl = \sum_{\tau\in D_1(i)}\int_{\bigcup_{v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\nabla_M u\cdot\vec{n}dl,$$

where the orientation of interval $[BC(e), BC(\tau)]$ should be taken in a sense with agreement to the orientation of ∂C_i, as mentioned in Section 4.2. When restricted in a triangle $\tau = [v_i, v_j, v_k]$

$$\nabla_M u|_\tau = u_i\nabla_M\phi_i + u_j\nabla_M\phi_j + u_k\nabla_M\phi_k.$$

According to the piecewise linearity of ϕ_i, ϕ_j, and ϕ_k, we know that $\nabla_M\phi_i$, $\nabla_M\phi_j$, and $\nabla_M\phi_k$ are three constant vectors when restricted in τ [28]; hence, so is $\nabla_M u|_\tau$. Therefore, the right-hand side of (5) becomes

$$\sum_{\tau\in D_1(i)}\nabla_M u\cdot\int_{\bigcup_{e,v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\vec{n}dl.$$

By exchanging the order of sums, we obtain

$$u_i\omega_{ii} + \sum_{j\in N_1(i)} u_j\omega_{ij}, \tag{6}$$

where

$$\begin{cases}\omega_{ij} = \sum\limits_{\tau,[v_i,v_j]\prec\tau}\nabla_M\phi_j\cdot\int_{\bigcup_{e,v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\vec{n}dl,\\ \omega_{ii} = -\sum\limits_{j\in N_1(i)}\omega_{ij}.\end{cases} \tag{7}$$

If v_i is a boundary vertex, by a similar derivation, we get the same expression as (6) by noticing the boundary condition of the PDE model. Thus, we get a uniform spatial discretization for all the mesh surfaces. Since the barycentric dual mesh is chosen, the coefficients in (6) are determined, and therefore, they are time independent.

Now, we discretize the linear model. We adopt implicit time discretization as follows:

$$A_i\frac{u_i^{n+1}-u_i^n}{\triangle t} = u_i^{n+1}\omega_{ii} + \sum_{j\in N_1(i)} u_j^{n+1}\omega_{ij},$$

where A_i is the area of the control cell of v_i. Denoting $U = (u_0, u_1, \ldots, u_{\mathrm{V}-1})'$, the above equation is formulated in the matrix form

$$WU^{n+1} = \mathrm{diag}(A_0, A_1, \ldots, A_{\mathrm{V}-1})U^n, \tag{8}$$

where W is determined by $\{\omega_{ij}\}$ and the areas of the control cells. As one will see, W is a highly sparse and symmetric matrix. Hence, the preconditioned biconjugate gradient (PBCG) method is a good choice to solve the linear system; see the Appendix. Since an implicit scheme is applied, one can choose large time steps. This discretization method is much more efficient than explicit schemes. It should be pointed out that one need not design a special storage structure for the sparse coefficient matrix by using the 1-disks and 1-neighbors of vertices of mesh surfaces.

4.3.2 Discretization of the Nonlinear Model

By a similar derivation with the linear model, we integrate on the two sides of the first equation of (2) over the control cell C_i of vertex v_i. According to the divergence theorem, the right-hand side of the equation becomes

$$\sum_{\tau\in D_1(i)} g(|\nabla_M u|)\nabla_M u\cdot\int_{\bigcup_{e,v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\vec{n}dl.$$

Since $\nabla_M u$ is a piecewise constant vector field on M, it is necessary to calculate the values on two triangles sharing the common edge, respectively. The contribution of triangle $\tau = [v_i, v_j, v_k]$ in the above summation is

$$g(|\nabla_M u|_\tau|)(u_i c_{ii,\tau} + u_j c_{ij,\tau} + u_k c_{ik,\tau}), \tag{9}$$

where

$$\begin{cases} c_{ij,\tau} = \nabla_M\phi_j\cdot\int_{\bigcup_{e,v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\vec{n}dl,\\ c_{ik,\tau} = \nabla_M\phi_k\cdot\int_{\bigcup_{e,v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\vec{n}dl,\\ c_{ii,\tau} = -c_{ij,\tau} - c_{ik,\tau}.\end{cases} \tag{10}$$

Once the dual-mesh structure is chosen, coefficients $c_{ii,\tau}$, $c_{ij,\tau}$, and $c_{ik,\tau}$ are determined, and therefore, they are independent of time.

We discretize the model (2) via a semi-implicit scheme:

$$A_i\frac{u_i^{n+1}-u_i^n}{\triangle t} = \sum_{\tau\in D_1(i)} g(|\nabla_M u|_\tau^n|)\left(u_i^{n+1}c_{ii,\tau} + u_j^{n+1}c_{ij,\tau} + u_k^{n+1}c_{ik,\tau}\right).$$

It can be written as in the matrix form:

$$K^n U^{n+1} = \mathrm{diag}(A_0, A_1, \ldots, A_{\mathrm{V}-1})U^n, \tag{11}$$

where A_i, $i = 0, 1, \ldots, \mathrm{V}-1$, and U are defined in the linear model, and K^n is also highly sparse and symmetric. Again, the PBCG method is a good choice to solve the system. Note that the coefficient matrix K^n is dependent on not only the coefficients $\{c_{ii,\tau}, c_{ij,\tau}, c_{ik,\tau}, c_{ji,\tau}, c_{jj,\tau}, c_{jk,\tau}, c_{ki,\tau}, c_{kj,\tau}, c_{kk,\tau}\}$, but also the data U^n, so it should be updated dynamically. However, the updating procedure is very simple and does not expend much CPU cost; see the Appendix for details.

4.3.3 Discretization of the Anisotropic Model

Based on the numerical scheme of the nonlinear isotropic PDE, one can design a discretization scheme for the anisotropic model (4) analogously. We integrate the two sides of the first equation in (4) over the control cell C_i of v_i and write the right-hand side of the equation as

$$\sum_{\tau\in D_1(i)}\left(g_1(|\nabla_M u|)(\nabla_M u\cdot e_1)e_1 + g_2(|\nabla_M u|)(\nabla_M u\cdot e_2)e_2\right)\cdot\int_{\bigcup_{e,v_i\prec e\prec\tau}[BC(e),BC(\tau)]}\vec{n}dl.$$

Here, we use the fact that $\nabla_M u$, $g_1(|\nabla_M u|)$, $g_2(|\nabla_M u|)$, e_1, and e_2 are all constants when restricted on a triangle.

Concretely, the contribution of triangle $\tau = [v_i, v_j, v_k]$ in the above summation is

$$\begin{aligned} & g_1\big(|\nabla_M u|_\tau|\big)\Big(u_i c^1_{ii,\tau} + u_j c^1_{ij,\tau} + u_k c^1_{ik,\tau}\Big) \\ & \quad + g_2\big(|\nabla_M u|_\tau|\big)\Big(u_i c^2_{ii,\tau} + u_j c^2_{ij,\tau} + u_k c^2_{ik,\tau}\Big), \end{aligned} \tag{12}$$

where

$$\begin{cases} c^1_{ij,\tau} = (\nabla_M \phi_j \cdot e_1)\left(e_1 \cdot \int_{\bigcup_{e, v_i \prec e \prec \tau} [BC(e), BC(\tau)]} \vec{n} dl\right), \\ c^2_{ij,\tau} = (\nabla_M \phi_j \cdot e_2)\left(e_2 \cdot \int_{\bigcup_{e, v_i \prec e \prec \tau} [BC(e), BC(\tau)]} \vec{n} dl\right), \\ c^1_{ik,\tau} = (\nabla_M \phi_k \cdot e_1)\left(e_1 \cdot \int_{\bigcup_{e, v_i \prec e \prec \tau} [BC(e), BC(\tau)]} \vec{n} dl\right), \\ c^2_{ik,\tau} = (\nabla_M \phi_k \cdot e_2)\left(e_2 \cdot \int_{\bigcup_{e, v_i \prec e \prec \tau} [BC(e), BC(\tau)]} \vec{n} dl\right), \\ c^1_{ii,\tau} = -c^1_{ij,\tau} - c^1_{ik,\tau}, c^2_{ii,\tau} = -c^2_{ij,\tau} - c^2_{ik,\tau}. \end{cases} \tag{13}$$

These coefficients are also determined once the dual-mesh structure and the orthogonal vector fields are chosen.

Now, the numerical scheme of the anisotropic (4) is described as

$$\begin{aligned} A_i \frac{u_i^{n+1} - u_i^n}{\triangle t} = & \sum_{\tau \in D_1(i)} g_1^n\Big(u_i^{n+1} c^1_{ii,\tau} + u_j^{n+1} c^1_{ij,\tau} + u_k^{n+1} c^1_{ik,\tau}\Big) \\ & + \sum_{\tau \in D_1(i)} g_2^n\Big(u_i^{n+1} c^2_{ii,\tau} + u_j^{n+1} c^2_{ij,\tau} + u_k^{n+1} c^2_{ik,\tau}\Big) \end{aligned}$$

or in matrix form

$$L^n U^{n+1} = \mathrm{diag}(A_0, A_1, \ldots, A_{\mathrm{V}-1}) U^n. \tag{14}$$

When $g_1(|\nabla_M u|)$ and $g_2(|\nabla_M u|)$ are not constant functions, the coefficient matrix L^n is dependent on not only the coefficients $\{c^1_{ii,\tau}, c^1_{ij,\tau}, c^1_{ik,\tau}, c^2_{ii,\tau}, c^2_{ij,\tau}, c^2_{ik,\tau}, c^1_{ji,\tau}, c^1_{jj,\tau}, c^1_{jk,\tau}, c^2_{ji,\tau}, c^2_{jj,\tau}, c^2_{jk,\tau}, c^1_{ki,\tau}, c^1_{kj,\tau}, c^1_{kk,\tau}, c^2_{ki,\tau}, c^2_{kj,\tau}, c^2_{kk,\tau}\}$, but also the data u at t^n. Hence, L^n need to be updated dynamically in each time step. Similarly, L^n is also highly sparse and symmetric.

In our applications, we choose two constants for $g_1(|\nabla_M u|)$ and $g_2(|\nabla_M u|)$. In this case, the coefficient matrix is independent of time and, hence, the updating procedure is skipped.

4.3.4 Computation of Coefficients

In this section, we derive more concise expressions for those coefficients obtained in the above sections. As one can see, the integrals of the normal vectors over the boundaries of control cells play an important role.

Theorem 1. *Let $\tau = [A, B, C]$ be a triangle and $\Gamma = \Gamma(t)$, $t \in [0, 1]$, be an arbitrary curve within the triangle with endpoints $\Gamma(0) = P$ and $\Gamma(1) = Q$, as shown in Fig. 3a. Then*

$$\int_{\Gamma(t)} \vec{n} dl = |PQ| \vec{n}_{PQ}, \tag{15}$$

where $\vec{n}$ is the normal vector of $\Gamma(t)$, and $\vec{n}_{PQ}$ is the unit vector perpendicular to the line segment $[P, Q] = \Gamma(1) - \Gamma(0)$.

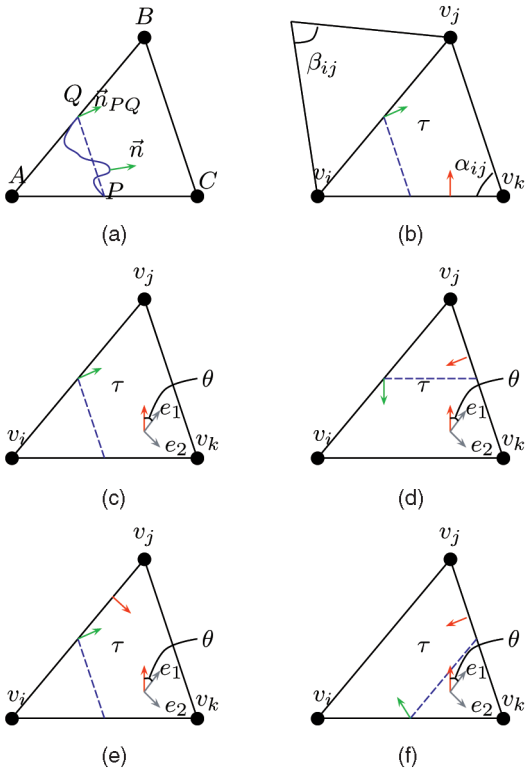

Fig. 3. Computation of coefficients. (a) Normal vector integral. (b) ω_{ij} and $c_{ij,\tau}$. (c) $c^1_{ij,\tau}$ and $c^2_{ij,\tau}$. (d) $c^1_{ji,\tau}$ and $c^2_{ji,\tau}$. (e) $c^1_{ik,\tau}$ and $c^2_{ik,\tau}$. (f) $c^1_{ki,\tau}$ and $c^2_{ki,\tau}$.

Proof. Consider the underlying plane of the triangle $[A, B, C]$ with coordinate system (x, y). Then, we have

$$\begin{aligned} \int_{\Gamma(t)} \vec{n} dl &= \int_0^1 \frac{(y', -x')}{|\Gamma'(t)|} |\Gamma'(t)| dt \\ &= \int_0^1 (y', -x') dt \\ &= \left(\int_0^1 y' dt, \int_0^1 -x' dt\right) \\ &= |PQ| \vec{n}_{PQ}. \end{aligned}$$

This proves the theorem. □

Based on the above theorem, the coefficients in the previous sections can be computed as follows: In Fig. 3b, the dashed blue line segment whose endpoints are the intersections of the boundary curve of control cell C_i within $\tau = [v_i, v_j, v_k]$ with the edges of τ is parallel to edge $[v_j, v_k]$. The green vector is the normal of the dashed blue line. The red vector stands for the gradient vector $\nabla_M \phi_j$ restricted in τ that is perpendicular to the opposite edge of vertex v_j and whose length is the reciprocal of the distance between v_j and the underlying line of $[v_i, v_k]$ [28]. Thus, the coefficients defined in (7), (10), and (13) can be calculated through simple vector operations; see the Appendix for details.

Further analysis will help to show the symmetry properties of the coefficient matrices in the diffusion models. Referring to Fig. 3b, by some analysis, one can easily obtain

$$\omega_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}),$$

which shows the symmetry of the coefficient matrix of the linear model (1). This expression was also obtained in [28] and [22]. We also have

$$c_{ij,\tau} = \frac{1}{2}\cot\alpha_{ij},$$

which demonstrates the symmetry of the coefficient matrix of the nonlinear model (2). Expressions for the coefficients of the anisotropic model such as $c^1_{ij,\tau}$ and $c^2_{ij,\tau}$ are a bit more complex. Assume that two piecewise constant orthogonal vector fields e_1 and e_2 are given. Suppose the angle from $\nabla_M\phi_j$ to e_1 (in clockwise) is θ, as shown in Figs. 3c, 3d, 3e, and 3f, and e_2 is rotated from e_1 clockwise by $\frac{\pi}{2}$. Then, we can express those coefficients via θ and angles $\angle i$, $\angle j$, and $\angle k$ with respect to vertices v_i, v_j, and v_k of τ. From (13) and Figs. 3c and 3d, we have

$$c^1_{ij,\tau} = \frac{1}{2\sin\angle k}\cos\theta\cos(\theta-\angle k) = c^1_{ji,\tau},$$

which shows the symmetry of $c^1_{ij,\tau}$ and $c^1_{ji,\tau}$. From Figs. 3e and 3f, we obtain that

$$c^1_{ik,\tau} = \frac{-1}{2\sin\angle j}\cos(\theta+\angle i)\cos(\theta-\angle k) = c^1_{ki,\tau}.$$

For $c^2_{ij,\tau}$, $c^2_{ji,\tau}$, $c^2_{ik,\tau}$, and $c^2_{ki,\tau}$, one simply replaces θ by $\theta+\frac{\pi}{2}$ and gets

$$c^2_{ij,\tau} = \frac{1}{2\sin\angle k}\sin\theta\sin(\theta-\angle k) = c^2_{ji,\tau}$$

and

$$c^2_{ik,\tau} = \frac{-1}{2\sin\angle j}\sin(\theta+\angle i)\sin(\theta-\angle k) = c^2_{ki,\tau}.$$

Therefore, the coefficient matrix of the anisotropic model is also symmetric. Furthermore, one can verify that $c^1_{ij,\tau} + c^2_{ij,\tau} = c_{ij,\tau}$. This fact is expected since the nonlinear model is just a special case of the anisotropic model in which $g_1(\cdot) = g_2(\cdot)$.

5 Applications

In this section, several applications of the three PDE models are to be presented. These applications include image denoising, image inpainting, harmonic map regularization, and texture generating over triangulated mesh surfaces.

Planar image processing is a classical and popular research field with many interesting problems and applications. A wealthy of literature has discussed the problems, and many techniques have been put forward in recent decades. Among them, the PDE-based methods have attracted much attention. These methods include scale-space constructing algorithms [31], [29], [36], [38], [51] for multiresolution representations, edge-preserving image denoising techniques [40], [17], [16], [18], [27], image decomposition methods [3], [4], [35], image inpainting algorithms [11], [9], [19], [6], [7], edge detection and segmentation techniques based on level set methods [14], [21], [39], [50], etc. Two good review papers in this aspect are [20] and [41].

5.1 Denoising Images over Triangulated Surfaces

So far, there are many efficient planar image denoising models including linear and various nonlinear PDEs. It can be shown that the linear PDE model is equivalent to Gaussian convolution. To preserve edges while denoising, various nonlinear PDE models are proposed by modifying the coefficient of heat exchange in the linear model [36] or by the variational principle [40]. Nonlinear image denoising models are difficult to be realized via convolution-based methods. Later on, PDE-based methods were generalized to denoise images on implicit surfaces [10]. In [5], the authors described a texture denoising model on mesh surfaces coupled with a surface fairing technique. Their method falls into finite-element methods (FEMs) based on the local parameterization of triangular mesh surfaces. Furthermore, handling of surface boundaries should be considered in their method since a smooth function space based on Loops subdivision is used.

In this section, we propose an algorithm for denoising images over triangulated surfaces by directly solving PDEs over the surfaces. The advantage is that difficulties are avoided because of nonconversions between mesh surfaces and parametric/implicit surfaces. In the following, the function u stands for the color information over the mesh surface. Furthermore, the noise model is assumed to be Gaussian. This yields a constraint term (also called a fidelity term)

$$\lambda(f-u)$$

in the final PDE model, where λ is the Lagrangian multiplier (see [40] for the planar case and [10]). The linear denoising model is thus

$$\begin{cases} u_t = \triangle_M u + \lambda(f-u), \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p,0) = f(p). \end{cases} \tag{16}$$

Based on the intrinsic TV (3), a typical nonlinear denoising model is

$$\begin{cases} u_t = \nabla_M \cdot \left(\frac{1}{\sqrt{|\nabla_M u|^2+\beta}}\nabla_M u\right) + \lambda(f-u), \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p,0) = f(p). \end{cases} \tag{17}$$

Different from the method in [5], the numerical discretization of (16) and (17) based on FVM is straightforward according to the numerical schemes described in the above section, and no special treatment of surface boundaries is needed in the final computation. The choice of the Lagrangian multiplier λ is a consideration. A different λ will result in a different effect. When λ is large, the denoising procedure will be affected mainly by the constraint term. However, if λ is small, then the diffusion term becomes the leading factor. λ can be considered to be in inverse proportion to the variance of the noise. In practical applications, the variance of the noise for each individual image is estimated since the accurate computation is expensive and unnecessary, as demonstrated in our experiments. Our experiments show that this simple and direct strategy works very well.

Our models work for any triangulated surfaces with arbitrary geometry and topology. We provide several examples to demonstrate the method, as shown in Figs. 4

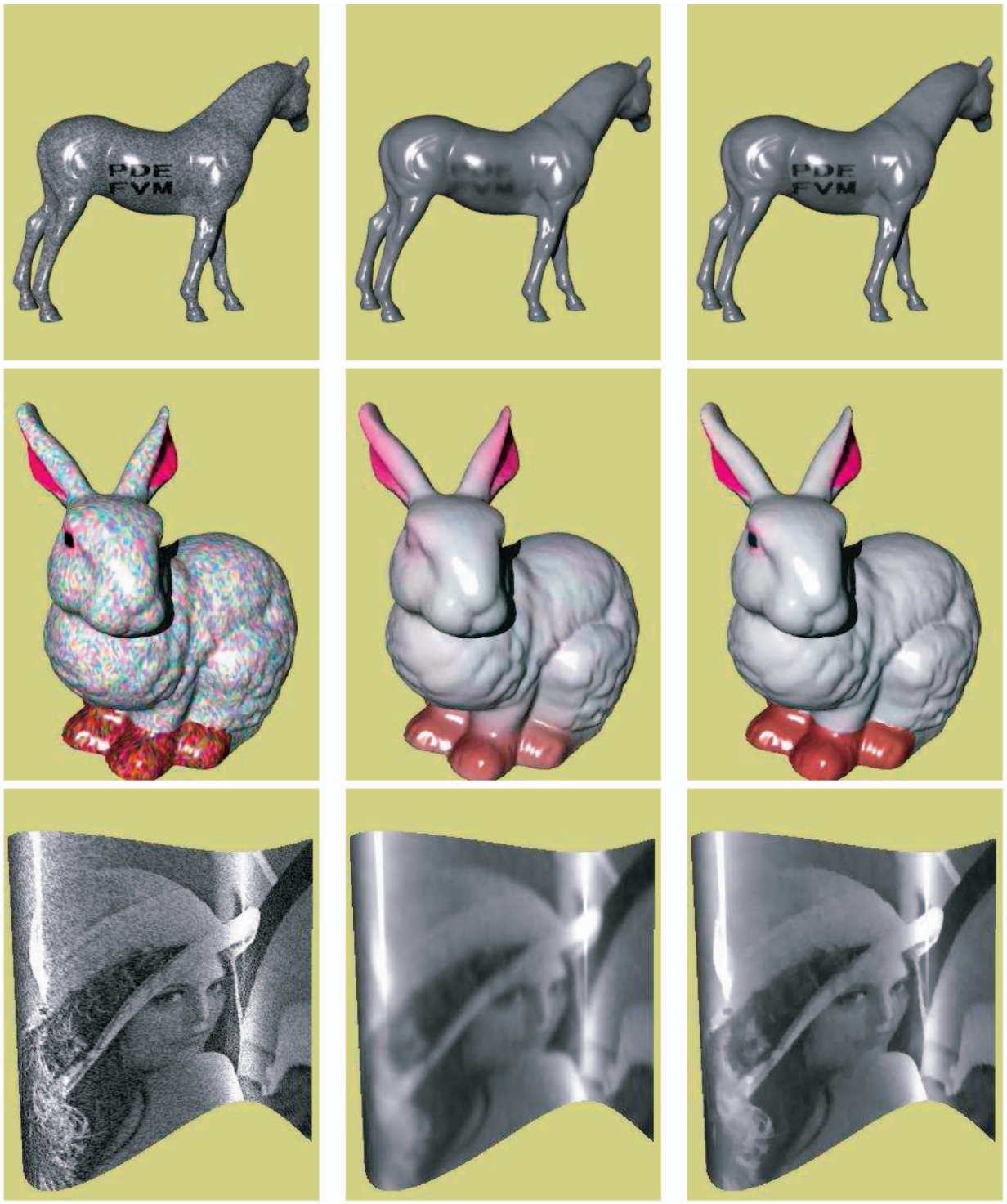

Fig. 4. Denoising images on closed (the first and the second rows) and open (the third row) surfaces. For the third row, boundary conditions should be considered.

and 5. Each row in the figures illustrates an example. In each example, the left image is the original image with noise, the middle image is the result of the linear denoising model (16), and the right image is the result of the intrinsic TV model (17). In Fig. 4, the first row shows an example of denoising a gray image with 25 percent noise on a closed surface, the second is an example of denoising a color image with 50 percent noise on a closed surface, and the third row illustrates an example of denoising the Lena image with 33 percent noise painted on an open mesh surface on which Neumann boundary conditions are used. From the examples, one can see that the nonlinear model produces better results than the linear model. Specifically, the nonlinear model preserves edges perfectly, whereas the linear model smoothes out them when removing noises from images. The CPU cost for both image denoising models is less than 1 minute.

In Fig. 5, we illustrate two examples of denoising images with different noises on a closed surface. The noise in the first example is 20 percent and much less than that in the second example in which 80 percent noise is added. As one can see

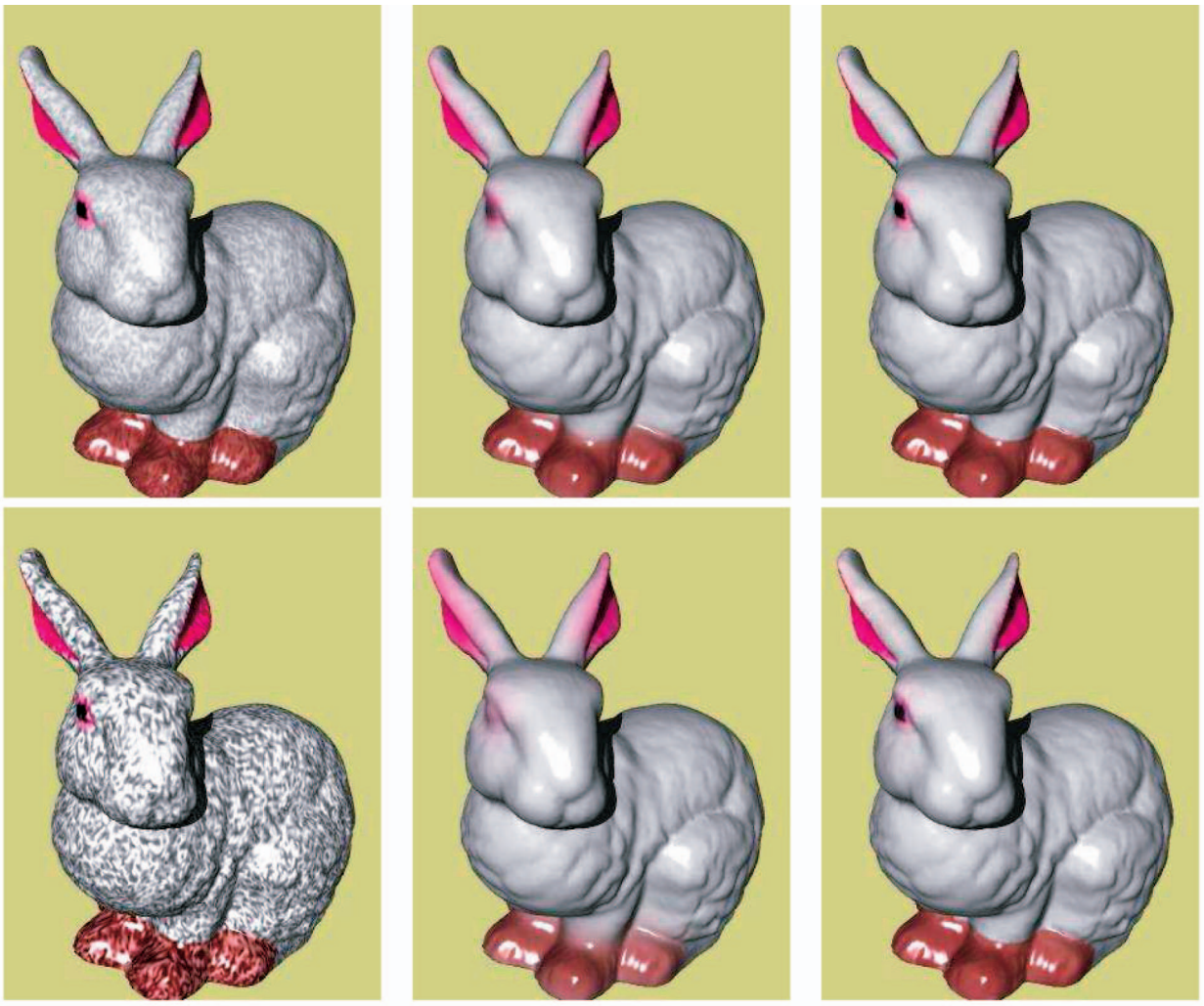

Fig. 5. Denoising images with different noises on a closed surface.

from the denoising results, noise removal is much more difficult in the second example than in the first one. However, the nonlinear model still produces acceptable results—preserving edges—although the noise is much stronger. For the linear denoising model, when the noise is strong enough, small structures of images will be smeared by the blur effect of the model (look at the eyes of the bunny).

5.2 Inpainting Images on Triangulated Surfaces

The image inpainting problem was first proposed by Bertalmio et al. in [11]. Usually, distortions such as scratches exist in an old photo. Image inpainting algorithms aim to repair these distortions automatically by computers. Inpainting algorithms can also be applied to features of movie magic, text removal [20], etc. There are many techniques for inpainting planar images [11], [9], [19], [6], [7]. However, little work is done on inpainting images on surfaces. The authors of the present paper proposed an image inpainting algorithm on implicit surfaces [54]. In this section, we present models and algorithms for inpainting images over triangulated mesh surfaces.

We assume that the noise model is Gaussian and the image information over $D \subset M$ is distorted. Since $f|_D$ is missing and, hence, not trustful, we introduce a function, which is called inpainting mask, as

$$1_D(p) = \begin{cases} 1, p \in M \setminus D \\ 0, p \in D \end{cases}$$

to help to build the fidelity term. The linear inpainting model is

$$\begin{cases} u_t = \triangle_M u + \lambda 1_D(f-u), \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p,0) = f(p)(f_{\mathrm{rd}}(p)), \end{cases} \tag{18}$$

where $f_{\mathrm{rd}}(p)$ is a random initial guess introduced as a choice of the initial condition, which is defined as

$$f_{\mathrm{rd}}(p) = \begin{cases} f(p), p \in M \setminus D, \\ \text{random number}, p \in D. \end{cases}$$

Sometimes, this simple trick can provide a faster inpainting procedure. A typical nonlinear model is the following intrinsic TV model:

$$\begin{cases} u_t = \nabla_M \cdot \left(\frac{1}{\sqrt{|\nabla_M u|^2 + \beta}} \nabla_M u \right) + \lambda 1_D(f-u), \\ \frac{\partial u}{\partial \vec{n}}|_{\partial M} = 0, \\ u(p,0) = f(p)(f_{\mathrm{rd}}(p)), \end{cases} \tag{19}$$

where $f_{\mathrm{rd}}(p)$ is the same as that in (18).

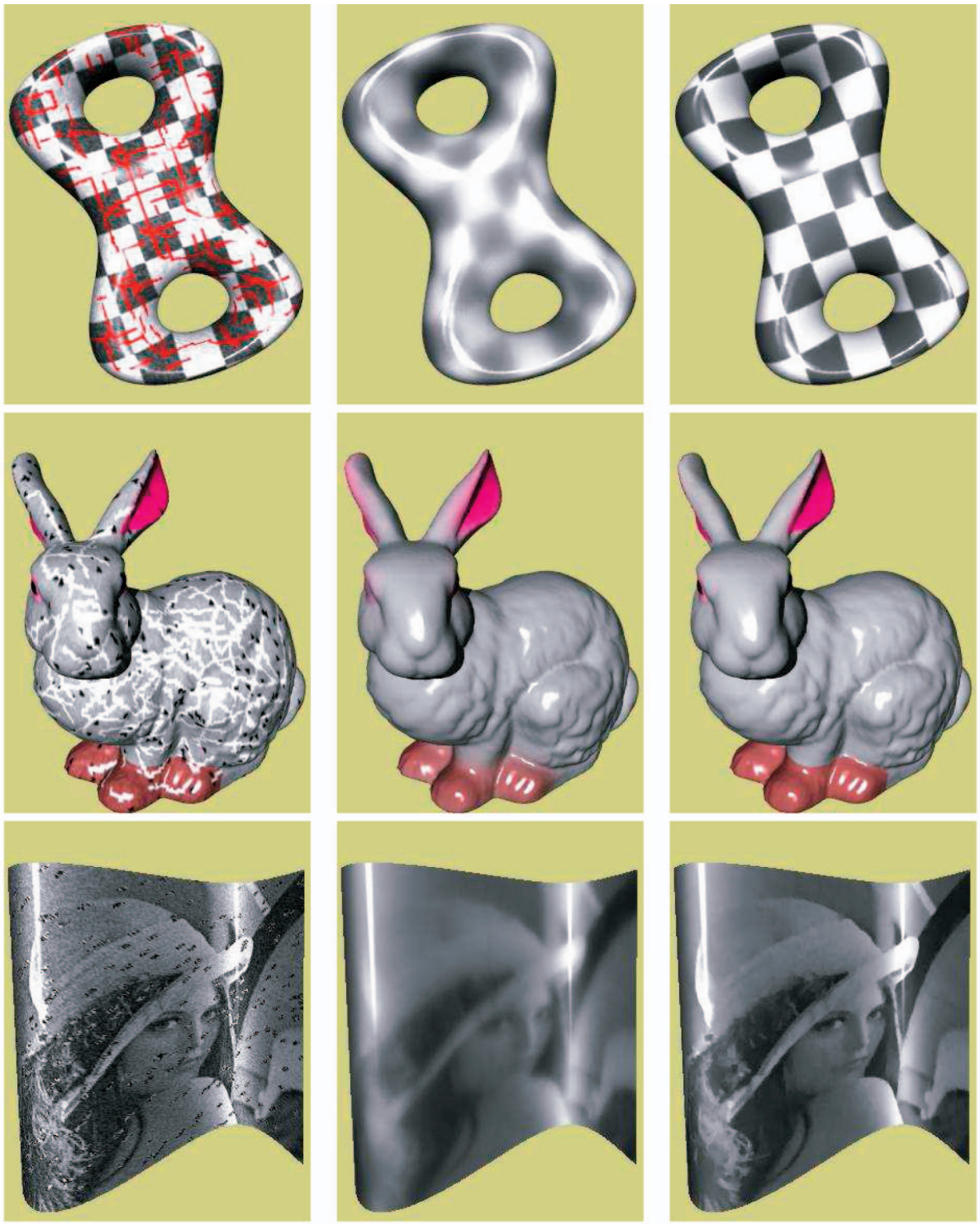

Fig. 6. Inpainting images on closed (the first and the second rows) and open (the third row) surfaces. Therein, the surface in the first row has a genus of 2. For the third row, boundary conditions are considered.

A similar technique with the discretization of (16) and (17) is applied to numerically discretizing (18) and (19). Here, three examples are illustrated in Fig. 6 to demonstrate the inpainting algorithm. For each row of an example, the left is the distorted image in which scratches (even crossing edges of images) or random spots or a combination of these two distortions can be observed; the middle and right images are inpainted results via the linear model (18) and the nonlinear inpainting model (19), respectively. As in the image denoising problem, the nonlinear inpainting model produces a better effect than the linear model. The linear model (18) blurs the undistorted part of the original image while inpainting the distorted part, whereas the nonlinear inpainting model (19) preserves the undistorted part perfectly. To speed up the computation, we choose larger time steps than those in the image denoising problem. Also, random initial guesses are used. Again,

only a few iterations are needed to obtain the results shown in Fig. 6. CPU costs do not exceed 1 minute.

5.3 Regularizing Harmonic Maps on Triangulated Surfaces

Consider a vector-valued function $u = (u_1, \ldots, u_m) : M \to S^{m-1}$ on a manifold M, where S^{m-1} is the unit sphere of dimension $m-1$. Especially, if $m = 3$, u is restricted on a 2D sphere, that is, a vector-valued function that has three components with unit length. Many quantities such as principal directions and normal directions of the surface M, and normalized **RGB** vectors of color images fall into this category. In the following, we introduce models to regularize vector-valued map u.

As pointed out in [10], one can obtain a coupled system of PDEs:

$$\frac{\partial u_k}{\partial t} = \nabla_M \cdot \left(|\nabla_M u|^{p-2} \nabla_M u_k \right) + u_k |\nabla_M u|^p, \qquad 1 \le k \le m,$$

for harmonic map regularization via constrained energy minimization. Here,

$$|\nabla_M u| = \sqrt{\sum_{1 \le k \le m} |\nabla_M u_k|^2},$$

and $p \ge 1$. In this paper, we study two typical regularization models: the linear model ($p = 2$) and a nonlinear model ($p = 1$). For a triangulated surface M, with the boundary and initial conditions concerned, the two models are given as follows:

$$\begin{cases} \frac{\partial u_k}{\partial t} = \nabla_M \cdot (\nabla_M u_k) + u_k |\nabla_M u|^2, \quad 1 \le k \le m, \\ \frac{\partial u_k}{\partial \vec{n}}|_{\partial M} = 0, \quad 1 \le k \le m, \\ u_k(p, 0) = f_k(p), \quad 1 \le k \le m, \end{cases} \tag{20}$$

and

$$\begin{cases} \frac{\partial u_k}{\partial t} = \nabla_M \cdot \left(\frac{\nabla_M u_k}{\sqrt{|\nabla_M u|^2 + \beta}} \right) + u_k |\nabla_M u|, \quad 1 \le k \le m, \\ \frac{\partial u_k}{\partial \vec{n}}|_{\partial M} = 0, \quad 1 \le k \le m, \\ u_k(p, 0) = f_k(p), \quad 1 \le k \le m, \end{cases} \tag{21}$$

where β is a small positive number.

It is straightforward to discretize (20) and (21) as previous models. After discretization, linear systems are then obtained whose coefficient matrices have little difference from those for models (1) and (3). It should be pointed out that all components must be calculated in each time step since they are coupled. We illustrate two examples in Fig. 7. Mesh surfaces with noisy normals are shown in the first column. The second column is the result of the linear model (20), and the third column is that of the nonlinear regularization model (21). The second row is the zoom-ins of the first row. One can see that the nonlinear regularization model preserves the sharp features of normals, whereas the linear model smoothes them out (look at the nose of the horse in the zoom-ins, also the lip and jaw of the Venus head). The computational time for both models ranges from several seconds to several minutes.

5.4 Generating Textures on Triangulated Surfaces

Texture mapping on mesh surfaces is very popular and has been studied for many years in computer graphics community. So far, a lot of algorithms have been put forward to map textures onto mesh surfaces, and most of these algorithms are based on surface parameterization. However, as pointed out in previous sections, parameterization causes many difficulties such as global parameterization is very hard for complex geometry and topology, local parameterization introduces distortion, and one has to handle cracks of atlases.

In this section, we present a method to generate textures directly on mesh surfaces by PDEs. In fact, such techniques appeared in more than 10 years ago [49], [52], and they are based on a kind of reaction-diffusion equations that initially appeared in chemistry science and were discovered by Turing in 1952 to generate "patterns" (textures in images) [48]. The basic idea in these techniques is to have a number of "chemicals" that diffuse at different rates and react with each other. The pattern is then obtained by assigning a brightness value to the concentration of one of the chemicals. It is obvious that different reaction-diffusion equations or even the same equation with different parameters generate different patterns. This method attracts little attention for its low efficiency. The first reason affecting the efficiency is that the PDEs are not solved directly over mesh surfaces. The authors of [52] solved the PDEs in the piecewise parametric spaces of the given surface. In [49], the authors projected the neighbors of a vertex of the mesh surface to a plane at first and then constructed a Voronoi graph and locally solved their reaction-diffusion model numerically. This projecting procedure is intrinsically a local parameterization, and hence, the algorithm is costly. Furthermore, errors are introduced into the numerical computation for metric distortion of parameterization. The second reason is that their numerical methods are all based on explicit schemes, which brings a strict time step constraint when the triangles of the mesh surfaces are seriously irregular and nonuniform. In this section, we present a new method to numerically solve the reaction-diffusion equations directly on mesh surfaces based on implicit/semi-implicit FVM schemes.

The first model is given as follows:

$$\begin{cases} \frac{\partial u_1}{\partial t} = D_1 \triangle_M u_1 + F(u_1, u_2), \\ \frac{\partial u_2}{\partial t} = D_2 \triangle_M u_2 + G(u_1, u_2), \\ \frac{\partial u_1}{\partial \vec{n}}|_{\partial M} = 0, \\ \frac{\partial u_2}{\partial \vec{n}}|_{\partial M} = 0, \end{cases} \tag{22}$$

where u_1 and u_2 are concentrations of two "chemicals" whose diffusion rates are D_1 and D_2, respectively, and F and G are two functions indicating the reaction between the two chemicals. The initial values of u_1 and u_2 are usually random numbers and hence are omitted in the reaction-diffusion model. Equation (22) is an isotropic model. To control the final textures more flexibly, one can adopt the following anisotropic model:

$$\begin{cases} \frac{\partial u_1}{\partial t} = \nabla_M \cdot \big(g_{11}(|\nabla_M u_1|)(\nabla_M u_1 \cdot e_1) e_1 + g_{12}(|\nabla_M u_1|) \\ (\nabla_M u_1 \cdot e_2) e_2 \big) + F(u_1, u_2), \\ \frac{\partial u_2}{\partial t} = \nabla_M \cdot \big(g_{21}(|\nabla_M u_2|)(\nabla_M u_2 \cdot e_1) e_1 + g_{22}(|\nabla_M u_2|) \\ (\nabla_M u_2 \cdot e_2) e_2 \big) + G(u_1, u_2), \\ (g_{11}(|\nabla_M u_1|)(\nabla_M u_1 \cdot e_1) e_1 + g_{12}(|\nabla_M u_1|)(\nabla_M u_1 \cdot e_2) e_2) \\ \cdot \vec{n}|_{\partial M} = 0, \\ (g_{21}(|\nabla_M u_2|)(\nabla_M u_2 \cdot e_1) e_1 + g_{22}(|\nabla_M u_2|)(\nabla_M u_2 \cdot e_2) e_2) \\ \cdot \vec{n}|_{\partial M} = 0, \end{cases} \tag{23}$$

in which g_{ij}, $i = 1, 2$, and $j = 1, 2$ are piecewise constant functions indicating diffusion rates of chemicals along different directions, and e_1 and e_2 are two piecewise constant orthogonal vector fields on M. Thus, one can

Fig. 7. Regularizing normal vectors on surfaces. The second row is the zoom-ins of the first row.

control the shapes of the final textures by modifying these parameters.

There are many choices for F and G. Different choices result in different reaction-diffusion models. Here, we choose the classical Turing model:

$$\begin{aligned} F(u_1, u_2) &= s(\gamma - u_1 u_2), \\ G(u_1, u_2) &= s(u_1 u_2 - u_2 - \mu), \end{aligned} \tag{24}$$

where s is the reaction rate, γ is the growth rate, and μ is the decay.

The discretizations of the isotropic model (22) and the anisotropic model (23) are straightforward based on implicit/semi-implicit FVM schemes. As in the harmonic map regularization problem, here, u_1 and u_2 are updated simultaneously since they are coupling with each other. Our numerical methods are direct and do not require any localprojection operation. Besides, the time-step constraint disappears. These advantages improve the algorithms' efficiency dramatically. We illustrate four examples, as shown in Fig. 8. Figs. 8a, 8b, and 8c show textures generated by the isotropic model (22), whereas the texture in Fig. 8d is generated by the anisotropic model (23). As one can see, the texture in Fig. 8d is with a certain direction. Comparatively, the anisotropic reaction-diffusion model generates more flexible textures by control of directional diffusion rates of chemicals than the isotropic one. CPU costs for generating textures are more expensive than those in the above three applications. Several minutes are needed to generate each surface texture in Fig. 8. However, it is much faster than the previous methods.

6 Conclusions and Future Work

In this paper, we present several diffusion equations (including linear, nonlinear, and anisotropic models)

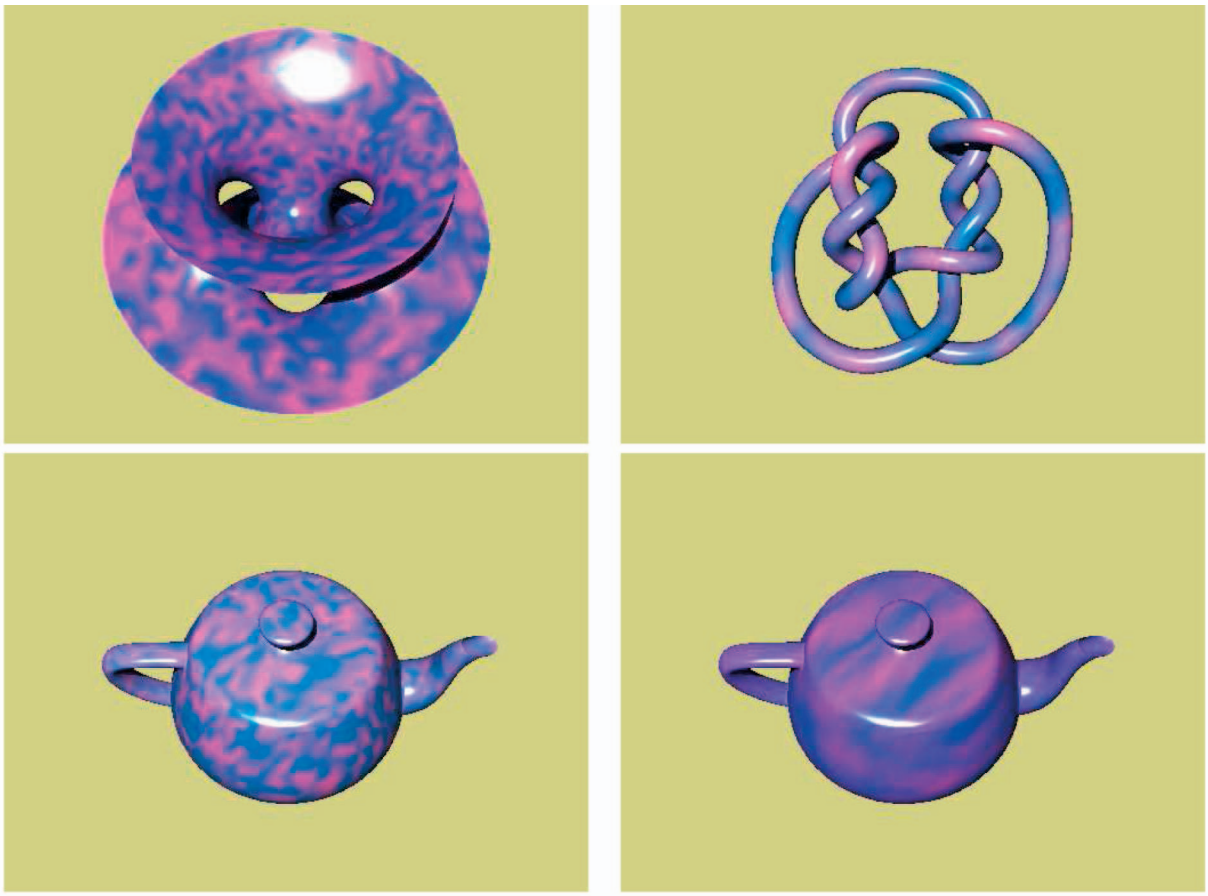

Fig. 8. Generating textures on surfaces.

over triangulated mesh surfaces and design numerical methods to solve these PDEs based on implicit/semi-implicit FVM schemes. Several applications, including image denoising, image inpainting, harmonic map regularization, and texture generating over triangulated surfaces are then discussed with examples. Our methods are direct and require no conversion between mesh surfaces and parametric/implicit surfaces. Thus, problems such as global parameterization, metric distortion, and data extension are avoided. In addition, our methods are not restricted by a strictttime-step constraint due to implicit/semi-implicit discretization, so that they are very efficient comparatively with the previous methods. Furthermore, our numerical schemes are suitable for triangulated surfaces with arbitrary geometry and topology. This is very important in data processing over mesh surfaces since the triangles of the mesh surfaces are usually highly irregular and nonuniform (this is especially obvious in the multiresolution and adaptive representation of mesh surfaces). Experimental results illustrate the flexibility and efficiency of our methods. Examples also suggest that nonlinear models generally produce better effects than linear models.

Several issues need to be considered in our future work. First, we are going to investigate other data processing problems such as deblurring, edge detection of images, applications of the Beltrami framework of manifold representation of images, and removal of other types of noises such as salt-pepper noise, as well as diffusion models based on color space considerations on arbitrary triangulated surfaces. Second, we plan to extend the methods to other types of mesh surfaces such as quadrilateral meshes and even meshes whose valences are arbitrary and variational. The computational efficiency and the theoretical analysis of the numerical methods and applications are also important issues worthy of further study.

Appendix

Implementation Based on PBCG

In this Appendix, we describe some implementation details of our numerical methods. The procedure includes three steps that are detailed as follows:

The first step is extracting topological information of the mesh surface. After loading data (including the vertex list and the triangle list of the mesh, as well as the color information defined on it), we find the 1-disks of all vertices and then 1-neighbors by a loop of the triangle list. Also, we compute the areas of all the triangles and, then, the areas of all the control cells. These data are stored in arrays.

The second step is the computation of coefficient matrices. For each triangle $\tau = [v_i, v_j, v_k]$, we compute its three vectors $\nabla_M \phi_i$, $\nabla_M \phi_j$, and $\nabla_M \phi_k$ and store them in an array indexed by the triangle. Then, coefficients defined in (7), (10), and (13) for the linear, nonlinear, and anisotropic models are calculated. These coefficients are stored in arrays cooperated with the arrays of 1-disks and 1-neighbors. All these quantities can be obtained via simple vector operations. In the following, we narrate this

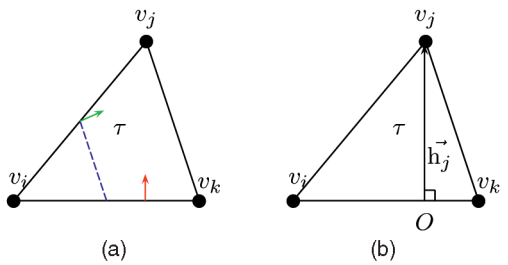

Fig. 9. Computation for normals and gradients.

in detail. From Theorem 1, we need outer normal vectors $\vec{n}$ colored in green and $\nabla_M\phi_i$, $\nabla_M\phi_j$, and $\nabla_M\phi_k$ colored in red, as shown in Fig. 3b, 3c, 3d, 3e, and 3f. Taking this as an example, we refer to Fig. 9. Since the barycentric dual structure is used, we know that the green unit normal in Fig. 9a is parallel to the height vector on edge v_jv_k. Similarly, according to [28], the red gradient in Fig. 9a is parallel to the height vector on edge v_iv_k. Therefore, the key is to compute the height vector of a triangle on some certain edge; see Fig. 9b for an example. This height vector $\vec{h}_j$ can be calculated as follows:

1. Let $\vec{v_1} = v_k - v_i$ and $\vec{v_2} = v_j - v_i$.
2. $\alpha = \frac{\vec{v_2}\cdot\vec{v_1}}{\vec{v_1}\cdot\vec{v_1}}$.
3. $O = \alpha v_k + (1-\alpha)v_i$.
4. $\vec{h}_j = v_j - O$.

Hence, the unit outer normal can be constructed by a reversion and a normalization operation. The gradient is just the height vector multiplied with a scaling factor [28].

The last step is numerically solving the PDEs. Based on the coefficients obtained in the second step, we solve linear systems by the PBCG method since the systems are almost highly sparse. Referring to some classical implementation of the PBCG method such as that described in [37], we need only design two operations: preconditioning and multiplication of sparse matrices and vectors. These operations use only nonzero elements of coefficient matrices and thus are very effective. Keep in mind that the nonzero elements of the coefficient matrices of the systems are all constructed by $\{\omega_{ii}, \omega_{ij}\}$, $\{c_{ii,\tau}, c_{ij,\tau}, c_{ik,\tau}, c_{ji,\tau}, c_{jj,\tau}, c_{jk,\tau}, c_{ki,\tau}, c_{kj,\tau}, c_{kk,\tau}\}$, $\{c^1_{ii,\tau}, c^1_{ij,\tau}, c^1_{ik,\tau}, c^2_{ii,\tau}, c^2_{ij,\tau}, c^2_{ik,\tau}, c^1_{ji,\tau}, c^1_{jj,\tau}, c^1_{jk,\tau}, c^2_{ji,\tau}, c^2_{jj,\tau}, c^2_{jk,\tau}, c^1_{ki,\tau}, c^1_{kj,\tau}, c^1_{kk,\tau}, c^2_{ki,\tau}, c^2_{kj,\tau}, c^2_{kk,\tau}\}$, and $|\nabla_M u|^n_\tau|$. We use an array to store the $|\nabla_M u|^n_\tau|$ values (for all triangles) and update them in each time step. The computation of $\nabla_M u|^n_{\tau=[v_i,v_j,v_k]}$ is through a linear combination of $\nabla_M\phi_i$, $\nabla_M\phi_j$, and $\nabla_M\phi_k$ with data u^n_i, u^n_j, and u^n_k at current time. For the linear model, the coefficient matrix is independent of time, and so are the two operations for PBCG. For the nonlinear models, the coefficient matrices depend on $|\nabla_M u|^n_\tau|$. However, the changes of the elements in the coefficient matrices merely include scalings (see (9), (11), (12), and (14)), and hence, the update of the coefficient matrices used in the two operations for PBCG is very fast and simple by the stored values $|\nabla_M u|^n_\tau|$ and the chosen function $g(|\nabla_M u|)$ (or $g_1(|\nabla_M u|)$ and $g_2(|\nabla_M u|)$ in the anisotropic model).

ACKNOWLEDGMENTS

The authors are supported by the National Key Basic Research Project of China (2004CB318000), the Outstanding Youth Grant of NSF of China (60225002), the NSF of China (60533060, 10671192, and 10701069), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20060358055), and the 111 Project (b07033).

REFERENCES

[1] "Compatible Spatial Discretizations" *The IMA Volumes in Mathematics and Its Applications,* vol. 142, D.N. Arnold, P.B. Bochev, R.B. Lehoucq, R.A. Nicolaides, and M. Shashkov, eds., Springer, 2006.

[2] O.K.C. Au, C.L. Tai, L.G. Liu, and H.B. Fu, "Dual Laplacian Editing for Meshes," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 3, pp. 386-395, May/June 2006.

[3] J.F. Aujol, G. Aubert, L.B. Feraud, and A. Chambolle, "Image Decomposition into a Bounded Variation Component and an Oscillating Component," *J. Math. Imaging and Vision,* vol. 22, no. 1, pp. 71-88, 2005.

[4] J.F. Aujol, G. Gilboa, T.F. Chan, and S. Osher, "Structure-Texture Image Decomposition-Modeling, Algorithms, and Parameter Selection," *Int'l J. Computer Vision,* vol. 67, no. 1, pp. 111-136, 2006.

[5] C.L. Bajaj and G. Xu, "Anisotropic Diffusion of Surfaces and Functions on Surfaces," *ACM Trans. Graphics,* vol. 22, no. 1, pp. 4-32, 2003.

[6] C.A.Z. Barcelos and M.A. Batista, "Image Inpainting and Denoising by Nonlinear Partial Differential Equations," *Proc. 16th Brazilian Symp. Computer Graphics and Image Processing (SIBGRAPI '03),* pp. 287-293, 2003.

[7] C.A.Z. Barcelos, M.A. Batista, A.M. Martins, and A.C. Nogueira, "Level Lines Continuation Based Digital Inpainting," *Proc. 17th Brazilian Symp. Computer Graphics and Image Processing (SIBGRAPI '04),* pp. 50-57, 2004.

[8] T. Barth and M. Ohlberger, "Finite Volume Methods: Foundation and Analysis," *Encyclopedia of Computational Mechanics.* John Wiley & Sons, 2004.

[9] M. Bertalmio, A.L. Bertozzi, and G. Sapiro, "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '01),* pp. 355-362, 2001.

[10] M. Bertalmio, L.T. Cheng, S. Osher, and G. Sapiro, "Variational Problems and Partial Differential Equations on Implicit Surfaces," *J. Computational Physics,* vol. 174, no. 2, pp. 759-780, 2001.

[11] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," *Proc. ACM SIGGRAPH '00,* pp. 417-424, 2000.

[12] A. Bossavit, "Generalized Finite Differences in Computational Electromagnetics," *Progress in Electromagnetics Research,* vol. 32, pp. 45-64, 2001.

[13] A. Bossavit, *Computational Electromagnetism.* Academic Press, 2004.

[14] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *Int'l J. Computer Vision,* vol. 22, pp. 61-79, 1997.

[15] E. Catmull and J. Clark, "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes," *Computer-Aided Design,* vol. 10, no. 6, pp. 350-355, 1978.

[16] T.F. Chan, S.H. Kang, and J. Shen, "Total Variation Denoising and Enhancement of Color Images Based on the CB and HSV Color Models," *J. Visual Comm. and Image Representation,* vol. 12, pp. 422-435, 2001.

[17] T.F. Chan, S. Osher, and J. Shen, "The Digital TV Filter and Nonlinear Denoising," *IEEE Trans. Image Processing,* vol. 10, no. 2, pp. 231-241, 2001.

[18] T.F. Chan and F. Park, "Data Dependent Multiscale Total Variation Based Image Decomposition and Contrast Preserving Denoising," Technical Report UCLA CAM Report 04-15, UCLA CAM, 2004.

[19] T.F. Chan and J. Shen, "Mathematical Models for Local Nontexture Inpaintings," *SIAM J. Applied Math.,* vol. 62, no. 3, pp. 1019-1043, 2001.

[20] T.F. Chan, J. Shen, and L. Vese, "Variational PDE Models in Image Processing," *Notice of Am. Math. Soc.,* vol. 50, pp. 14-26, 2003.

[21] T.F. Chan and L.A. Vese, "An Active Contour Model without Edges," *Lecture Notes in Computer Science,* vol. 1682, pp. 141-151, Springer, 1999.

[22] M. Desbrun, A.N. Hirani, M. Leok, and J.E. Marsden, *Discrete Exterior Calculus,* http://arxiv.org/abs/math.DG/0508341, 2005.
[23] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr, "Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow," *Proc. ACM SIGGRAPH,* 1999.
[24] Q. Du and L.L. Ju, "Finite Volume Methods on Spheres and Spherical Centroidal Voronoi Meshes," *SIAM J. Numerical Analysis,* vol. 43, no. 4, pp. 1673-1692, 2005.
[25] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, Circulation-Preserving, Simplicial Fluids," *ACM Trans. Graphics,* vol. 26, no. 1, 2007.
[26] P.M. Gandoin and O. Devillers, "Progressive Lossless Compression of Arbitrary Simplicial Complexes," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 372-379, 2002.
[27] G. Gilboa, N. Sochen, and Y.Y. Zeevi, "A Forward-and-Backward Diffusion Process for Adaptive Image Enhancement and Denoising," *IEEE Trans. Image Processing,* vol. 11, no. 7, pp. 689-703, 2002.
[28] A.N. Hirani, "Discrete Exterior Calculus," PhD dissertation, California Inst. Technology, 2003.
[29] A. Hummel, "Representations Based on Zero-Crossings in Scale-Space," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '86),* pp. 204-209, 1986.
[30] R. Kimmel, "Intrinsic Scale Space for Images on Surfaces: The Geodesic Curvature Flow," *Graphical Models and Image Processing,* vol. 59, no. 5, pp. 365-372, 1997.
[31] J. Koenderink, "The Structure of Images," *Biological Cybernetics,* vol. 50, pp. 363-370, 1984.
[32] G. Lin and P.Y. Yu, "An Improved Vertex Caching Scheme for 3D Mesh Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 4, pp. 640-648, July/Aug. 2006.
[33] M. Meyer, M. Desbrun, P. Schröder, and A. Barr, "Discrete Differential-Geometry Operator for Triangulated 2-Manifolds," *Visualization and Math. III,* H.-C. Hege and K. Polthier, eds., Springer, 2002.
[34] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer, 2002.
[35] S. Osher, A. Sole, and L. Vese, "Image Decomposition and Restoration Using Total Variation Minimization and the H^{-1} Norm," *Multiscale Modeling and Simulation,* vol. 1, pp. 349-370, 2003.
[36] P. Perona and J. Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 7, pp. 629-639, July 1990.
[37] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C,* second ed. Cambrige Univ. Press, 1992.
[38] E. Radmoser, O. Scherzer, and J. Weickert, "Scale-Space Properties of Regularization Methods," *Lecture Notes in Computer Science,* vol. 1682, pp. 211-222, Springer, 1999.
[39] J.R. Rommelse, H.X. Lin, and T.F. Chan, "A Robust Level Set Algorithm for Image Segmentation and Its Parallel Implementation," Technical Report UCLA CAM Report 03-05, UCLA CAM, 2003.
[40] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear Total Variation Based Noise Removal Algorithms," *Physica D,* vol. 60, pp. 259-268, 1992.
[41] J. Shen, "Inpainting and the Fundamental Problem of Image Processing," *SIAM News,* vol. 36, no. 5, 2003.
[42] L. Shi and Y. Yu, "Inviscid and Incompressible Fluid Simulation on Triangle Meshes," *J. Computer Animation and Virtual Worlds,* vol. 15, pp. 173-181, 2004.
[43] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, "Bounded Distortion Piecewise Mesh Parameterization," *Proc. IEEE Conf. Visualization (VIS '02),* pp. 355-362, 2002.
[44] A. Spira and R. Kimmel, "Enhancing Images Painted on Manifolds," *Lecture Notes in Computer Science,* vol. 3459, pp. 492-502, Springer, 2005.
[45] A. Spira and R. Kimmel, "Segmentation of Images Painted on Parametric Manifolds," *Proc. European Signal Processing Conf. (EUSIPCO '05),* Sept. 2005.
[46] J. Stam, "Flows on Surfaces of Arbitrary Topology," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 724-731, 2003.
[47] G. Taubin and J. Rossignac, "Geometric Compression through Topological Surgery," *ACM Trans. Graphics,* vol. 17, no. 2, pp. 84-115, 1998.
[48] A. Turing, "The Chemical Basis of Morphogenesis," *Philosophical Trans. Royal Soc. B,* vol. 237, pp. 37-72, 1952.
[49] G. Turk, "Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion," *Computer Graphics,* vol. 25, no. 4, pp. 289-298, 1991.
[50] L.A. Vese and T.F. Chan, "A Multiphase Level Set Framework for Image Segmentation Using the Mumford-Shah Model," *Int'l J. Computer Vision,* vol. 50, no. 3, pp. 271-293, 2002.
[51] J. Weickert and B. Benhamouda, "A Semidiscrete Nonlinear Scale-Space Theory and Its Relation to the Perona-Malik Paradox," *Advances in Computer Vision.* Springer, pp. 1-10, 1997.
[52] A. Witkin and M. Kass, "Reaction-Diffusion Textures," *Computer Graphics (Proc. ACM SIGGRAPH '91),* vol. 25, no. 4, pp. 299-308, 1991.
[53] C.L. Wu, J.S. Deng, and F.L. Chen, "Fast Data Extrapolating," *J. Computational and Applied Math.,* vol. 206, no. 1, pp. 146-157, 2007.
[54] C.L. Wu, J.S. Deng, W.M. Zhu, and F.L. Chen, "Inpainting Images on Implicit Surfaces," *Proc. 13th Pacific Conf. Computer Graphics and Applications (PG '05),* pp. 142-144, 2005.
[55] G. Xu, Q. Pan, and C.L. Bajaj, "Discrete Surface Modelling Using Partial Differential Equations," *Computer Aided Geometric Design,* vol. 23, no. 2, pp. 125-145, 2006.
[56] Y.Z. Yu, K. Zhou, D. Xu, X.H. Shi, H.J. Bao, B.N. Guo, and H.Y. Shum, "Mesh Editing with Poisson-Based Gradient Field Manipulation," *Proc. ACM SIGGRAPH '04,* pp. 641-648, 2004.
[57] E. Zhang, K. Mischaikow, and G. Turk, "Feature-Based Surface Parameterization and Texture Mapping," *ACM Trans. Graphics,* vol. 24, no. 1, pp. 1-27, 2005.
[58] H.K. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit and Nonparametric Shape Reconstruction from Unorganized Points Using a Variational Level Set Method," *Computer Vision and Image Understanding,* vol. 80, no. 3, pp. 295-319, 2000.
[59] D. Zorin and P. Schroder, "Subdivision for Modeling and Animation," *ACM SIGGRAPH '00 Course Notes,* 2000.

Chunlin Wu was born in Jiangxi, Peoples Republic of China, in 1982. He received the PhD degree from the University of Science and Technology of China, Hefei, Peoples Republic of China, in 2006. Currently, he is a postdoctoral in the Department of Mathematics, University of Science and Technology of China. His research interests are in computer graphics and image processing.

Jiansong Deng was born in Shangdong, Peoples Republic of China, in 1971. He received the PhD degree from the University of Science and Technology of China, Hefei, Peoples Republic of China, in 1998. Currently, he is a professor in the Department of Mathematics, University of Science and Technology of China. His research interests include computer-aided geometric design and computer graphics.

Falai Chen was born in Anhui, Peoples Republic of China, in 1966. He received the PhD degree from the University of Science and Technology of China, Hefei, Peoples Republic of China, in 1994. He is a professor in the Department of Mathematics, University of Science and Technology of China. His research interests include computer-aided geometric design and computer graphics.