Geometric Surface Processing via Normal Maps

Tolga Tasdizen
School of Computing, University of Utah
and
Ross Whitaker
School of Computing, University of Utah
and
Paul Burchard
Department of Mathematics, UCLA
and
Stanley Osher
Department. of Mathematics, UCLA

We propose that the generalization of signal and image processing to surfaces entails filtering the normals of the surface, rather than filtering the positions of points on a mesh. Using a variational strategy, penalty functions on the surface geometry can be formulated as penalty functions on the surface normals, which are computed using geometry-based shape metrics and minimized using fourth-order gradient descent partial differential equations (PDE). In this paper, we introduce a two step approach to implementing geometric processing tools for surfaces: (i) operating on the normal map of a surface, and (ii) manipulating the surface to fit the processed normals. Iterating this two-step process, we can efficiently implement geometric fourth-order flows by solving a set of coupled second-order PDEs. The computational approach uses level set surface models; therefore, the processing does not depend on any underlying parameterization. This paper will demonstrate that the proposed strategy provides for a wide range of surface processing operations, including edge-preserving smoothing and high-boost filtering. Furthermore, the generality of the implementation makes it appropriate for very complex surface models, e.g. those constructed directly from measured data.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid and object representations.*

Additional Key Words and Phrases: surface fairing, geometric surface processing, anisotropic diffusion, high-boost filtering, level sets.

1. INTRODUCTION

The fundamental principles of signal processing give rise to a wide range of useful tools for manipulating and transforming signals and images. The generalization of these principles

Author's address: Tolga Tasdizen, 50 S. Central Campus Drive, School of Computing, University of Utah Salt Lake City, UT 84112-9205. Phone: (801) 585-3742. E-mail: tolga@sci.utah.edu

to the processing of 3D surfaces has become an important problem in computer graphics, visualization, and vision. For instance, 3D range sensing technologies produce high resolution descriptions of objects, but they often suffer from noise. Medical imaging modalities such as MRI and CT scans produce large volumes of scalar or tensor measurements, but surfaces of interest must be extracted through some segmentation process or fitted directly to the measurements. These surfaces typically contain topological artifacts such as holes and unconnected pieces.

The goal of this paper is to introduce a new surface processing strategy that is flexible, general, and geometric. By *flexible* we mean that the framework should provide a basis for a broad variety of capabilities, including surface processing tools that resemble the state-of-the-art in image processing algorithms. The proposed methods should apply to a *general* class of surfaces. Users should be able to process complex surfaces of arbitrary and changing topology, and obtain meaningful results with very little *a priori* knowledge about the shapes. By *geometric* we mean that output of surface processing algorithms should depend on *surface shape and resolution*, but should be independent of arbitrary decisions about the representation or parameterization.

The work presented in this paper is based on the proposition that the natural generalization of image processing to surfaces is via the *surface normal vectors*. Thus, a smooth surface is one that has smoothly varying normals. Penalty functions on the surface normals typically give rise to fourth-order partial differential equations (PDE). Our strategy is to use a two step approach: (i) operating on the normal map of a surface, and (ii) manipulating the surface to fit the processed normals. Iterating this two-step process, we can efficiently implement fourth-order flows by solving a set of coupled second-order PDEs. In this light, the differences between surface processing and image processing are threefold:

(1) Normals are vector valued and constrained to be unit length; the processing techniques must accommodate this.

(2) Normals live on a manifold (the surface) and cannot be processed using a flat metric, as is typically done with images.

(3) Normals are coupled with the surface shape, thus the normals should drag the surface along as their values are modified during processing.

This paper presents an implementation that represents surfaces as the level sets of volumes and computes the processing of the normals and the deformation of the surfaces as solutions to a set of PDEs. In some applications, such as animation, models are manually generated by a designer, and the parameterization is not arbitrary but is an important aspect of the geometric model. In these cases, mesh-based processing methods offer a powerful set of tools, such as hierarchical editing [Guskov et al. 1999], which are not yet possible with the proposed representation. However, in other applications, such as 3D segmentation and surface reconstruction [Malladi et al. 1995; Whitaker 1998], the processing is data driven, surfaces can deform quite far from their initial shapes and change topology; hence, user intervention is not practical. Furthermore, when considering processes other than isotropic smoothing, such as nonlinear smoothing, the creation or sharpening of small features can exhibit noticeable effects of the mesh topology—the creation of new features requires changes in the mesh parameterization. In contrast, the underlying grid for level sets is independent of the surface shape; therefore, the only limitation for the creation of new features is the resolution of the grid. Hence, the use of a level set formulation en-

ables us to achieve a "black box" behavior and build surface processing techniques that are especially useful when processing measured data.

We have introduced an anisotropic diffusion for surfaces based on processing the normal map in [Tasdizen et al. 2002]. This paper discusses the mathematical foundations of the normal map processing strategy in detail and provides details of the numerical implementation as well as introducing high-boost filtering of normals as a new surface processing tool. The specific contributions are:

(1) a novel approach based on surface normals for geometric processing of surfaces;
(2) a numerical method for solving geometric fourth-order level set equations for surfaces in two simpler steps, thereby avoiding the explicit computation of unstable high-order derivatives; and
(3) examples of three geometric surface processing algorithms with applications to complex data sets.

The rest of this paper is organized as follows. We will discuss related surface processing work in Section 2. In Section 3, we formulate our splitting approach for solving geometric fourth-order level set equations for surfaces. In the limit, this approach is equivalent to solving the full, fourth-order flow, Appendix (A), but it generalizes to a wide range of processes and makes no assumptions about the shapes of the solutions. In Section 4, we show results for isotropic and anisotropic diffusion. To demonstrate the flexibility of the proposed framework, we will also show results of high-boost surface filtering implemented with our framework in Section 5. The numerical implementations of our approach is covered in Appendix (B). Conclusions and directions for future work will be discussed in Section 6.

2. RELATED WORK

The majority of surface processing research has been in the context of *surface fairing* with the motivation of smoothing surfaces to create aesthetically pleasing models. Surface fairing can be accomplished either by minimizing an energy function that favors smooth surfaces [Moreton and Séquin 1992; Welch and Witkin 1992; Halstead et al. 1993; Welch and Witkin 1994] or by applying smoothing filters [Taubin 1995; Desbrun et al. 1999; Guskov et al. 1999]. Energy minimization is a global method whereas filtering uses local neighborhoods. In the rest of this section, we review related work in these two categories. An approach that falls between these two extremes is based on Wiener filtering which utilizes arbitrary local spectral properties of the mesh [Alexa 2002].

Energy functions can depend on the geometry of the surface or the parameterization. Geometric functions make use of invariants such as principal curvatures, which are parameterization independent, intrinsic properties of the surface. Therefore, geometric approaches produce results that are not affected by arbitrary decisions about the parameterization; however, geometric invariants are nonlinear functions of surface derivatives that are computationally expensive to evaluate. Parameterization dependent functions are linear substitutes for geometric invariants.

One way to smooth a surface is to incrementally reduce its surface area. This can be accomplished by mean curvature flow (MCF), a second-order PDE,

$$\frac{\partial \mathbf{x}}{\partial t} = -H\mathbf{N} = -\left(\frac{\kappa_1 + \kappa_2}{2}\right)\mathbf{N} \tag{1}$$

where κ_1, κ_2 are the principal curvatures and H is the mean curvature at a point $\mathbf{x}$ on the surface, $\mathbf{N}$ is the surface normal, and the parameter t tracks the deforming surface shape. For parameterized surfaces, the membrane energy function, a linear substitute for surface area, is

$$\int_\Omega X_u^2 + X_v^2 \, du \, dv \tag{2}$$

where $X(u,v)$ and Ω are surface parameterization and its domain, respectively. The variational derivative of (2) is the Laplacian

$$\Delta X = X_{uu} + X_{vv} \, , \tag{3}$$

which is a linear substitute for mean curvature; however, they are equivalent only if the parameterization is orthonormal everywhere. These methods generally produce unsatisfactory results due to inherent limitations such as inability to preserve features, a systematic shrinking, and the introduction of high-curvature singularities.

A second-order energy function is the integral of *total curvature* over the surface S

$$\int_S \kappa_1^2 + \kappa_2^2 \, dS \tag{4}$$

which has been shown to deform surfaces into spheres when minimized [Polden 1997]. We will refer to (4) as the *total curvature penalty* which should not be confused with the local quantity *total curvature*. The total curvature penalty is a geometric (invariant) property of the surface that can be minimized by a fourth-order PDE which is very difficult to solve. The mesh fairing approach of [Welch and Witkin 1994], which minimizes (4), fits local polynomial basis functions to local neighborhoods for the computation of total curvature. These polynomial basis functions range from full quadratic polynomials to constrained quadratics and planar approximations. Depending on the complexity of the local neighborhood, the algorithm must choose, at each location, which basis to employ. Ambiguities result at locations where multiple bases provide equally good representations.

If we penalize the parameterization (i.e. non-geometric), equation (4) becomes the thin plate energy function

$$\int_\Omega X_{uu}^2 + 2X_{uv}^2 + X_{vv}^2 \, du \, dv \tag{5}$$

where X and Ω are as defined for (2). The variational derivative of (5) is the linear biharmonic operator

$$\Delta^2 X = X_{uuuu} + 2X_{uuvv} + X_{vvvv} \tag{6}$$

which is a fourth-order operator used for surface fairing [Welch and Witkin 1992].

Moreton and Séquin propose a geometric energy function that penalizes the variation of principle curvatures [1992]. This function has a sixth-order variational derivative which requires very large computation times. The analysis and implementation of general energy functions above second order remains an open problem, which is beyond the scope of this paper. Evidence in this paper and elsewhere [Desbrun et al. 1999; Schneider and Kobbelt 2000] suggests that fourth-order geometric flows form a sufficient foundation for a general, geometric surface processing system.

Taubin pioneers the linear filter based approaches to surface fairing. He observes that simple Gaussian filtering associated with the membrane energy causes shrinkage [1995].

He eliminates this problem by designing a low pass filter using a weighted average of the Laplacian (3) and the biharmonic operator (6). The weights must be fine-tuned to obtain the non-shrinking property. Analyzed in the frequency domain, this low-pass filter can be seen as a Gaussian smoothing shrinking step followed by an unshrinking step. Taubin shows that any polynomial transfer function in the frequency domain can be implemented with this method [1996]. A related approach in which surfaces are smoothed by simultaneously solving the membrane (2) and thin plate (5) energy functions is proposed in [Kobbelt et al. 1998]. Desbrun et al. use implicit integration to build a computationally efficient method for mesh fairing [1999]. Guskov et al. defines down- and up-sampling tools and smoothing filters for irregular meshes to build a multiresolution mesh processing framework [1999]. Their work is based on a generalized low pass filter which uses a non-uniform relaxation operator that minimizes a locally weighted quadratic energy of second-order differences on the mesh.

The techniques proposed in this paper are also related to that of [Chopp and Sethian 1999], who derive the intrinsic Laplacian of curvature for an implicit curve, and solve the resulting fourth-order nonlinear PDE. However, their method does not generalize to implicit surfaces. Moreover, they argue that the numerical methods used to solve second-order flows are not practical, because they lack long term stability. They propose several new numerical schemes, but none are found to be completely satisfactory due to their slow computation and inability to handle singularities. As a generalization of this PDE for surfaces, Schneider and Kobbelt propose using the intrinsic Laplacian of mean curvature, $\Delta_B H$, for meshes, where Δ_B is the Laplace-Beltrami operator, i.e. the Laplacian for parameterized surfaces [2000]. However, that approach works only for meshes, and relies on analytic properties of the steady-state solutions, $\Delta_B H = 0$, by fitting surface primitives that have those properties. Thus, the formalism does not generalize well to applications, such as surface reconstruction, where the solution is a combination of measured data and a fourth-order smoothing term. Also, it does not apply to other types of smoothing processes, such as anisotropic diffusion that minimizes nonlinear feature-preserving penalties. We solve a more general class of surface flows with a variational basis in an effective, stable splitting method.

An example of a splitting strategy can be found in [Ballester et al. 2001], where the authors penalize the smoothness of a vector field while simultaneously forcing the gradient directions of a gray scale image to closely match the vector field. The penalty function on the normal field is proportional to the divergence of the normal vectors. It forms a high-order interpolation function, which is shown to be useful for image inpainting—recovering missing patches of data in 2D images. The strategy of simultaneously penalizing the divergence of a normal field and the mismatch of this field with the image gradient is closely related to the total curvature penalty function used in this paper. However, our formulation emphasizes the processing of normals on an arbitrary surface manifold (rather than the flat geometry of an image), with an explicit relationship to fourth-order surface flows. Furthermore, this paper establishes new directions for surface flows— toward edge-preserving surface smoothing and feature enhancement. Our proposed PDE splitting approach is related to the methods in [Schneider and Kobbelt 2000], which we discuss in detail in Section 3.

Our splitting method requires diffusing unit-length vectors on a non-flat manifold. Perona proposes a method for diffusing orientation-like quantities on flat manifolds [1998]. This method solves the problem of diffusing 2D unit vectors as a 1D problem of angle

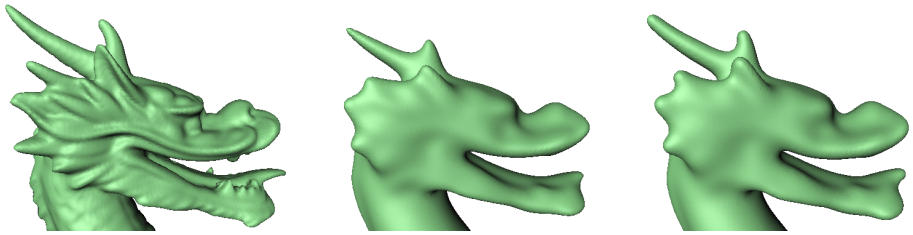


Fig. 1. Second- vs. fourth-order surface smoothing. From left to right: Original model, mean curvature flow, and isotropic fourth-order surface flow.

diffusion; however, it is not rotationally invariant and it does not generalize to the diffusion of unit vectors in higher dimensions. Several authors use harmonic maps theory to solve the diffusion of unit vectors defined on higher dimensional non-flat manifolds, e.g. surfaces [Bertalmio et al. 2001; Tang et al. 2000]. Hence, their methods are closely related to some of the problems we solve in this, but they do not provide a solution to the coupling between surface shape and the unit vectors because their goal is not surface processing.

3. MINIMIZING TOTAL CURVATURE

One of the underlying strategies of our approach is to use *geometric surface processing*, where the output of the process depends only on the shape of the input surface, and does not contain artifacts from the underlying parameterization. The motivation for this strategy is discussed in detail in [Schneider and Kobbelt 2001], where the influence of the parameterization on surface fairing results is clearly shown, and higher-order nonlinear geometric flows are proposed as the solution.

As an illustration of the importance of higher-order geometric processing, consider the results in Figure 1, which demonstrates the differences between processing surfaces with mean curvature flow (MCF) and the isotropic fourth-order PDE that minimizes the total curvature penalty (4). The amount of smoothing for both processes in this example were chosen to be qualitatively similar, and yet important differences can be observed on the smaller features of this model. MCF has shortened the horns of the original model, and yet they remain sharp—not a desirable behavior for a "smoothing" process. This behavior for MCF is well documented as a pinching off of cylindrical objects and is expected from the variational point of view: MCF minimizes surface area and therefore will quickly eliminate smaller parts of a model. Sapiro discusses volume preserving forms of second-order flows [2001], but these processes compensate by enlarging *the object as a whole*, which exhibits, qualitatively, the same behavior on small features. The isotropic fourth-order PDE, on the other hand, preserves the structure of these features much better *while* smoothing them as can be seen in Figure 1. Note that all of the surfaces in this paper are represented and processed volumetrically. To display the results, we render a mesh obtained with the Marching Cubes algorithm [Lorenson and Cline 1987].

We propose a two-step solution based on letting the surface shape to lag the normals as they are filtered and then refitting the surface to the normals by a separate process. For general fourth-order surface flows such as isotropic and anisotropic diffusion, both of these steps involve solving second-order PDEs. The first second-order PDE is used for minimizing a penalty function on the normals. The other second-order PDE minimizes the discrepancy between the modified normals and the surface; in other words, it refits the

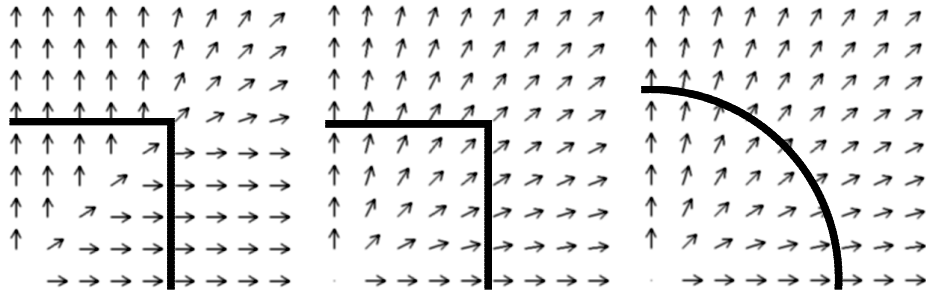


Fig. 2. Shown here in 2D, the process begins with a shape and constructs a normal map from the distance transform (left), modifies the normal map according to a PDE derived from a penalty function (center), and re-fits the shape to the normal map (right).

surface to the normals. Figure 2 shows this three step process graphically in 2D—shapes give rise to normal maps, which, when filtered give rise to new normal maps, which finally give rise to new shapes. The rest of this section is organized as follows. Level set methods are introduced in Section 3.1. We formulate total curvature as a function of the normal map and derive gradient descent minimizations for general functions of total curvature in Section 3.2; this gives rise to the first PDE mentioned above. The surface refitting process is discussed in Section 3.3; this gives rise to the other second-order PDE.

3.1 Level set methods

In this section, we briefly introduce the notation of level set methods. We can describe the deformation of a regular surface using the 3D velocity of each of its constituent points, *i.e.*, $\partial \mathbf{x}(t)/\partial t$ for all $\mathbf{x} \in S$. We represent the deforming surfaces implicitly as a function of the parameter t

$$S = \{\mathbf{x}(t) \mid \phi(\mathbf{x}(t), t) = 0\}, \tag{7}$$

where ϕ is the embedding function. Surfaces defined in this way divide a volume into two parts: inside ($\phi > 0$) and outside ($\phi < 0$). It is common to choose ϕ to be the signed distance transform of S, or an approximation thereof. The surface remains a level set of ϕ over time, and thus taking the total derivative with respect to time (using the chain rule) gives

$$\partial \phi/\partial t = -\nabla \phi \cdot \partial \mathbf{x}/\partial t \tag{8}$$

Because $\nabla \phi$ is proportional to the surface normal, $\partial \mathbf{x}/\partial t$ affects ϕ only in the direction of the normal— motion in any other direction is merely a change in the parameterization. This family of PDEs and the upwind scheme for solving them on a discrete grid is the methods of *level sets* [Osher and Sethian 1988]. For instance, using this framework and $\partial \mathbf{x}/\partial t$ from (1), the PDE on ϕ that describes the motion of a surface by mean curvature is

$$\partial \phi/\partial t = -\nabla \phi \cdot H\mathbf{N} = -\|\nabla \phi\| H. \tag{9}$$

Surface integrals of penalty functions have corresponding volume integrals which quantify the associated properties for the embedded surfaces of ϕ. A general surface penalty function based on the total curvature penalty (4) can be written for level set surfaces as

$$\mathscr{G}_\phi = \int_U G\left(\kappa_1^2 + \kappa_2^2\right) \| \nabla \phi \| \, d\mathbf{x}, \tag{10}$$

where $U \subset \Re^3$ is the volumetric domain of ϕ. When G is the identity function, (10) reduces to the total curvature penalty. The rest of this paper discusses methods for minimizing this penalty function in a stable and computationally efficient manner.

3.2 Total curvature of normal maps

In this section, we formulate total curvature of a surface from its normal map. Then, we derive the variational PDEs on the normal map that minimize functions of total curvature. When using implicit representations, one must account for the fact that derivatives of functions defined on the surface are computed by projecting their 3D derivatives onto the surface tangent plane. The 3×3 projection matrix for the implicit surface normal is

$$\mathbf{P} = \nabla\phi \otimes \nabla\phi / ||\nabla\phi||^2, \tag{11}$$

where $\otimes$ is the tensor product defined as $\mathbf{a} \otimes \mathbf{a} = \mathbf{a}\mathbf{a}^T$. Consequently, the projection matrix onto the surface tangent plane is $\mathbf{I} - \mathbf{P}$, where $\mathbf{I}$ is the identity matrix.

The local geometry of a surface can be described with the first and second fundamental forms, (I) and (II), respectively [DoCarmo 1976]. The eigenvalues of the matrix $(I)^{-1}(II)$, which we refer to as the *shape matrix*, are the principal curvatures of the surface independent of the parameterization. For an implicit surface, the shape matrix is obtained by differentiating the normal map and projecting the derivative onto the surface tangent plane. We define the differential of the normal map

$$\nabla\mathbf{N} = \left(\nabla\mathbf{N}_{(1)} \;\; \nabla\mathbf{N}_{(2)} \;\; \nabla\mathbf{N}_{(3)}\right)^T, \tag{12}$$

as the matrix whose rows are the gradient vectors of the components of $\mathbf{N}$ which we have denoted by $\mathbf{N}_{(i)}$ for $i = 1, 2$, and 3. Then the shape matrix is the projection $\nabla\mathbf{N}(\mathbf{I} - \mathbf{P})$, which measures the intrinsic change in the normals by mapping the differentials of $\mathbf{N}$ on to the tangent planes of ϕ. The Euclidean norm of the shape matrix is the sum of squared principal curvatures, i.e. total curvature,

$$\kappa^2 = \kappa_1^2 + \kappa_2^2 = ||(\nabla\mathbf{N})(\mathbf{I} - \mathbf{P})||^2. \tag{13}$$

We can use (13) to define an energy of the normal map that is analogous to the general energy function of ϕ defined in (10)

$$\mathscr{G}_{\mathbf{N}} = \int_U G(||(\nabla\mathbf{N})(\mathbf{I} - \mathbf{P})||^2)\, d\mathbf{x}. \tag{14}$$

The first variation of this energy with respect to the normals is a second order PDE. It is crucial to observe that, even though the projection operator $\mathbf{P}$ is a function of ϕ, it independent of $\mathbf{N}$ because we fix ϕ as we process $\mathbf{N}$. Hence, $\mathbf{P}$ does not increase the order of the first variation of (14). In contrast, taking the first variation of (10) with respect to ϕ directly, would have yielded a much harder to solve fourth order PDE on ϕ.

As we process the normal map to minimize (14), letting ϕ lag, we must ensure that the normal vectors maintain the unit length constraint. Solutions to constrained optimization problems defined on non-flat manifolds are discussed in [Bertalmio et al. 2001; Tang et al. 2000]. Using the method of Lagrange multipliers, we obtained the first variation of the constrained energy as

$$\frac{d\mathscr{G}}{d\mathbf{N}} = -(\mathbf{I} - \mathbf{N} \otimes \mathbf{N})\nabla \cdot \left[g\left(||(\nabla\mathbf{N})(\mathbf{I} - \mathbf{P})||^2\right)\nabla\mathbf{N}(\mathbf{I} - \mathbf{P})\right] \tag{15}$$

where g is the derivative of G with respect to its argument, κ^2. We will discuss several choices for G in Section 4. The projection operator, $\mathbf{I}-\mathbf{N}\otimes\mathbf{N}$, forces the changes to $\mathbf{N}$ to be perpendicular to itself in accordance with the unit length constraint. This operator is different from the other projection operator $\mathbf{I}-\mathbf{P}$ due to the decoupling of $\mathbf{N}$ and ϕ. Finally, the gradient descent PDE for the normals is $\partial\mathbf{N}/\partial t = -d\mathscr{G}/d\mathbf{N}$.

3.3 Surface evolution via normal maps

We have shown how to evolve the normals to minimize functions of total curvature; however, the final goal is to process the surface, which requires deforming ϕ. Therefore, the next step is to relate the deformation of the level sets of ϕ to the evolution of $\mathbf{N}$. Suppose that we are given the normal map $\mathbf{N}$ to some set of surfaces, but not necessarily level sets of ϕ—as is the case if we filter $\mathbf{N}$ and let ϕ lag. We can manipulate ϕ so that it fits the normal field $\mathbf{N}$ by minimizing a penalty function that quantifies the discrepancy between the gradient vectors of ϕ and the target normal map. That penalty function is

$$\mathscr{D}_\phi = \int_U D(\phi)\, d\mathbf{x}, \text{ where } D(\phi) = \sqrt{\nabla\phi\cdot\nabla\phi} - \nabla\phi\cdot\mathbf{N}. \tag{16}$$

The integrand, which is always a positive scalar, is proportional to the *sine* of the angle between the gradient vectors of ϕ and the target normal vectors.

The first variation of this penalty function with respect to ϕ is

$$\frac{d\mathscr{D}}{d\phi} = -\nabla\cdot\left[\frac{\nabla\phi}{||\nabla\phi||} - \mathbf{N}\right] = -\left[H^\phi - H^\mathbf{N}\right] \tag{17}$$

where H^ϕ is the mean curvature of the level set surface and $H^\mathbf{N}$ is half the divergence of the normal map. Then, the gradient descent PDE that minimizes (16) is $d\phi/dt = -||\nabla\phi||d\mathscr{D}/d\phi$. The factor of $||\nabla\phi||$, which is typical with level set formulations [Sethian 1999], comes from the fact that we are manipulating the shape of the level set, which is embedded in ϕ, as in (8). According to (17), the surface moves as the difference between its own mean curvature and that of the normal field.

The proposed splitting strategy for solving fourth-order level-set flows entails processing the normals and allowing ϕ to lag and then be refitted later, in a separate process. We have derived a gradient descent for the normal map based on a certain class of penalty functions that use the total curvature defined in Section 3.2. This process is denoted in Figure 3 as the $d\mathscr{G}/\mathbf{N}$ loop. The surface refitting to the normal map is formulated as a gradient descent in (17). This process is the $d\mathscr{D}/d\phi$ loop in Figure 3. The overall algorithm shown in Figure 3 repeats these two steps to minimize the penalty functions in terms of the surface. We refer to both of these processes, back-to-back, as one iteration of our algorithm. In Appendix (A) we will show that the overall process of concatenating these two second-order PDEs is equivalent to the fourth-order flow on the original surface. An alternate splitting approach for solving the same fourth-order level-set flows would be to simultaneously solve both second-order PDEs $d\mathscr{G}/d\mathbf{N}$ and $d\mathscr{D}/d\phi$ using a Lagrange multiplier instead of concatenating them as we have done. This approach was taken by Ballester et al. to solve the image inpainting problem [2001]. However, this approach uses a weighted sum of the two second-order PDEs; therefore, it is not clear whether it solves the original fourth-order flow. Moreover, in our case, due to the significant computational overhead of setting up the diffusion of normal vectors, it is more efficient to concatenate the two second-order PDEs and to do multiple consecutive iterations of $d\mathscr{G}/d\mathbf{N}$.

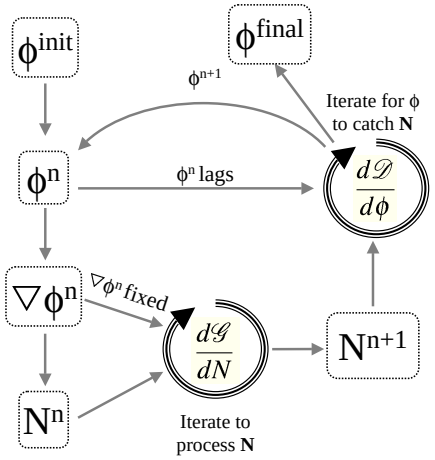


Fig. 3. Flow chart

4. ISOTROPIC AND ANISOTROPIC DIFFUSION

The flexible normal map energy minimization and surface refitting methodology introduced in Section 3 allows us to experiment with various forms of G in (14) that give rise to different classes of penalty functions. The choice of $G(\kappa^2) = \kappa^2$ leads to an isotropic diffusion. This choice works well for smoothing surfaces and eliminating noise, but it also deforms or removes important features. This type of smoothing is called isotropic because it corresponds to solving the heat equation on the normal map with a constant, scalar conduction coefficient, which is the same as Gaussian smoothing, for images. Isotropic diffusion is not particularly effective if the goal is to denoise a surface that has an underlying structure with fine features. This scenario is common when extracting surfaces from 3D imaging modalities, such as magnetic resonance imaging (MRI), in which the measurements are inherently noisy.

The problem of preserving features while smoothing away noise has been studied extensively in computer vision. Anisotropic diffusion introduced in [Perona and Malik 1990] has been very successful in dealing with this problem in a wide range of images. Perona & Malik (P&M) proposed to replace Laplacian smoothing, which is equivalent to the heat equation $\partial I/\partial t = \nabla \cdot \nabla I$, with a nonlinear PDE

$$\partial I/\partial t = \nabla \cdot \left[g\left(\| \nabla I \|^2 \right) \nabla I \right], \tag{18}$$

where I is generally the grey-level image. This PDE is the first variation of

$$\int_U G\left(\| \nabla I \|^2 \right) \, dx \, dy, \tag{19}$$

where g in (18), the derivative of G with respect to $\| \nabla I \|^2$, is the edge stopping function, and U is the image domain. P&M suggested using $g(x) = e^{-\|\nabla I\|^2/2\mu}$, where μ is a positive, free parameter that controls the level of contrast of edges that can affect the smoothing process. Notice that $g(\| \nabla I \|)$ approaches 1 for $\| \nabla I \| \ll \mu$ and 0 for $\| \nabla I \| \gg \mu$. Edges are generally associated with large image gradients, and thus diffusion across edges is

stopped while regions that are relatively flat undergo smoothing. Solutions to (18) can actually exhibit an inverse diffusion near edges, and can enhance or sharpen smooth edges that have gradients greater than μ [Perona and Malik 1990].

A great deal of research has focused on modified second-order flows that produce better results than MCF. Using level set models, several authors have proposed smoothing surfaces by weighted combinations of principle curvatures. For instance, Whitaker has proposed a nonlinear reweighting scheme that favors the smaller curvature and preserves cylindrical structures [1994]. Lorigo et al. propose a smoothing by the minimum curvature [2000]. A variety of other combinations have been proposed [Sapiro 2001]. A similar set of curvature-based algorithms have been developed for surface meshes. For instance, Clarenz et al. propose a modified MCF as an anisotropic diffusion of the surface [2000]. They threshold a weighted sum of the principle curvatures to determine the surface locations where edge sharpening is needed. Tangential displacement is added to the standard MCF at these locations for sharpening the edges. Although, this flow produces results that tend to preserve sharp features, it is not a strict generalization of [Perona and Malik 1990] anisotropic diffusion from images to surfaces. Another mesh-based modified MCF is proposed in [Ohtake et al. 2000] where a threshold on the mean curvature is used to stop over-smoothing. Taubin proposes a "linear anisotropic" Laplacian operator for meshes that is based on a separate processing of the normals [2001]. It is essentially a reweighting of the Laplacian. In a different context, anisotropic diffusion as a modified surface area minimization for height functions was proposed in [Desbrun et al. 2000].

These level set and mesh based methods are all modifications of curvature flows, and are therefore all second-order processes. Because they are based on reweightings of curvature, these methods always smooth the surface in one direction or another. They do not exhibit a sharpening of details, which is achieved by the P&M equation (for images) through an inverse diffusion process. Hence, these methods are not satisfactory generalizations of the P&M anisotropic diffusion equation. The generalization of P&M diffusion to surfaces requires a higher-order geometric flow which is achieved from variational principles by choosing the appropriate function of total curvature in (14). For instance,

$$G(\kappa^2) = 2\mu^2\left(1 - e^{-\frac{\kappa^2}{2\mu^2}}\right), \text{ and } g(\kappa^2) = e^{-\frac{\kappa^2}{2\mu^2}}, \tag{20}$$

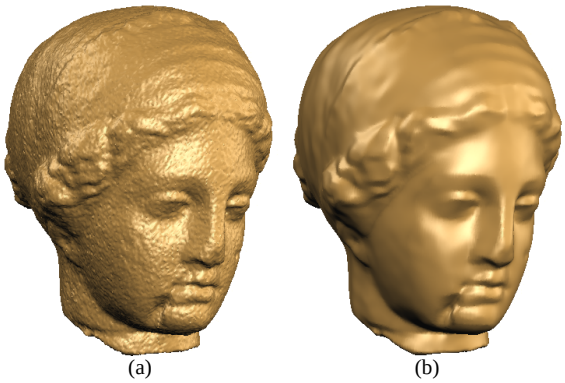
where g is the derivative of G with respect to κ^2. The first variation with respect to the surface normals gives a vector-valued anisotropic diffusion on the level set surface—a straightforward generalization of (18). This flow preserves or enhances areas of high curvature, which we will call *creases*. Creases are the generalization of edges in images to surfaces [Eberly 1996].

4.1 Results

Figure 4(a) illustrates an example of the skin surface, which was extracted, via isosurfacing, from an MRI data set. Notice that the roughness of the skin is noise, an artifact of the measurement process. This model is also quite complex because, despite our best efforts to avoid it, the isosurfaces include many convoluted passages up in the sinuses and around the neck. Isotropic diffusion, shown in Figure 4(b), is marginally effective for denoising the head surface. Notice that the sharp edges around the eyes, nose, lips and ears are lost in this process. The differences between anisotropic diffusion and isotropic diffusion can clearly be observed in Figure 4(c). There is no noticeable difference in the results of the

(a) (b) (c)

Fig. 4. Processing results on the MRI head model: (a) original isosurface, (b) isotropic diffusion, and (c) anisotropic diffusion. The small protrusion under the nose is a physical marker used for registration.



(a) (b)

Fig. 5. (a) Noisy venus head model, and (b) smoothed version after three iterations of anisotropic diffusion.

two processes around the smooth areas of the original model such as the forehead and the cheeks; however, very significant differences exist around the lips and the eyes. The creases in these areas, which have been eliminated by isotropic diffusion, are preserved by the anisotropic process. Note that the free parameter μ in (20) was fixed at 0.1 for all of the results shown in this paper. Unlike, in P&M image diffusion, this parameter does not need to be changed for different surface models. In the context of P&M image diffusion, the units of μ are in gray levels; consequently, the optimal choice of μ is image dependent. However, for surfaces, the units are in curvature, which is data independent. This makes it possible to choose a μ value that gives consistent results over a broad range of surfaces.

The computation time required for one iteration of the main processing loop operating on this model is approximately 15 minutes on a *1.7 Ghz Intel processor* for both isotropic and anisotropic diffusion. The results shown in Figure 4(b) and (c) are both after three iterations which translates to around 45 minutes of processing time. The generality of the proposed approach comes at the cost of significant computation time. However, the method is practical with state-of-the-art computers and is well-poised to benefit from parallel computing architectures, due to its reliance on local, iterative computations.

Another example of denoising by anisotropic diffusion is shown in Figure 5. Noise was added to the original model, which in this case is a $221 \times 221 \times 161$ volume. After three iterations of the main processing loop the noise was successfully removed while preserving

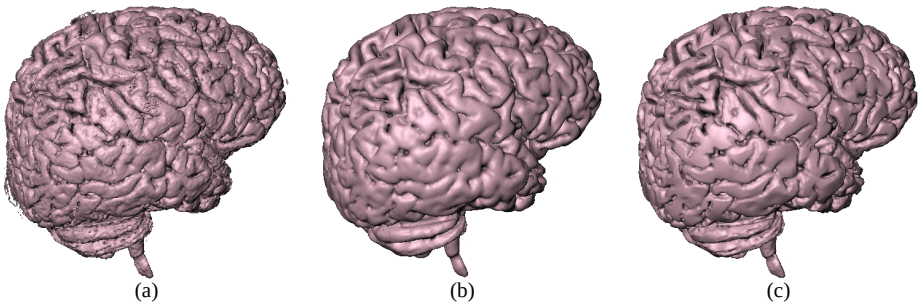


Fig. 6. (a) Original brain isosurface from MRI data set, (b) result of MCF, and (c) after 5 iterations of anisotropic diffusion.

the features of the original model. The quality of these results compares favorably with results from the same model shown in [Clarenz et al. 2000]. Our results demonstrate better preservation of fine, sharp details, such as those around the eyes and in the hair. The computation times per iteration for this example are approximately five minutes compared to 15 minutes per iteration for the example in Figure 4. This is indicative of the relatively high degree of complexity of the MRI based model in the previous example.

Figure 6(a) shows a different isosurface (the cortex) extracted from the same MRI scan as the model in Figure 4. The complexity of this model, i.e. the many tightly nested folds, make it ill-suited for mesh based deformations. Also, the main cortical surface has many detached pieces, an artifact of the segmentation process. As an indication of this complexity, we note that object enclosed by the cortical surface has more than 700 connected components. The approach proposed in this paper can automatically simplify topologically noisy features due to the level set implementation — an important aspect of denoising measured surfaces.

The examples in Figure 7 demonstrates another aspect of the proposed method. Although the original model in Figure 7(a) was constructed as a volume directly from 3D range data [Curless and Levoy 1996], it does not exhibit significant noise. Hence, smoothing is done with the purpose of simplification of the original model rather than denoising in these examples. Running isotropic diffusion for many steps creates a linear scale space where details in the model are progressively eliminated in accordance to their scale; the scales on the skin and the horns have been eliminated in Figure 7(b) and Fig 7(c), respectively. When running the proposed method for anisotropic diffusion, however, surfaces *tend* toward solutions that have piecewise smooth normals with sharp discontinuities in the normal map—analogous to the behavior of the P&M equation for intensity images. Such properties in the normal map correspond to surfaces consisting of planar patches and smooth patches bounded by sharp creases. Thus, the proposed method generates a feature preserving scale space, very much like that of P&M for images. These results, which are shown in Figure 7(d) and (e), support our proposition that processing the normals of a surface is the natural generalization of image processing. The non-linear progression of elimination of details from the smallest scale to the largest also suggests applications of this method to surface compression and multi-resolution modeling.

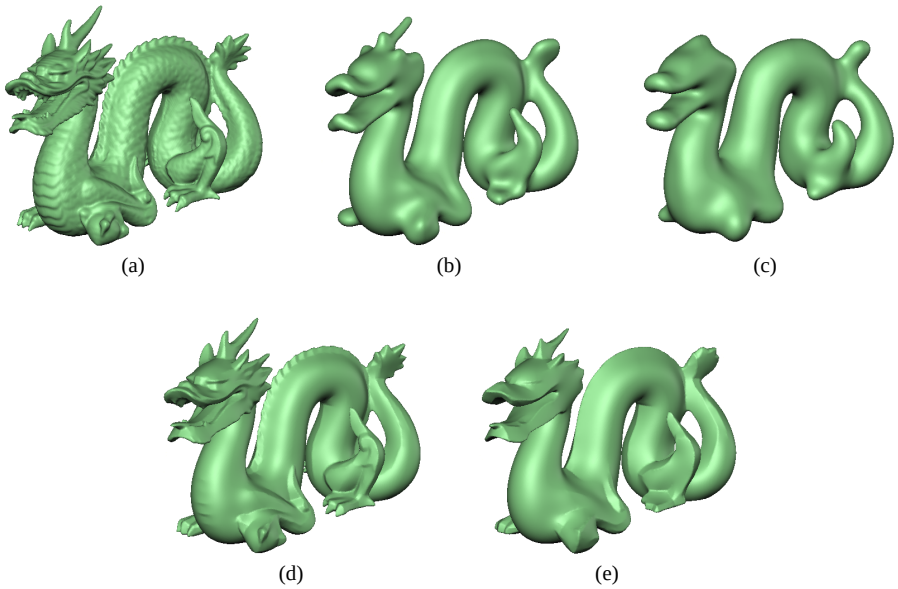


Fig. 7. (a) Original model. Isotropic diffusion: (b) after 10 iterations, and (c) after 20 iterations. Anisotropic diffusion: (a) after 10 iterations, and (b) after 20 iterations.

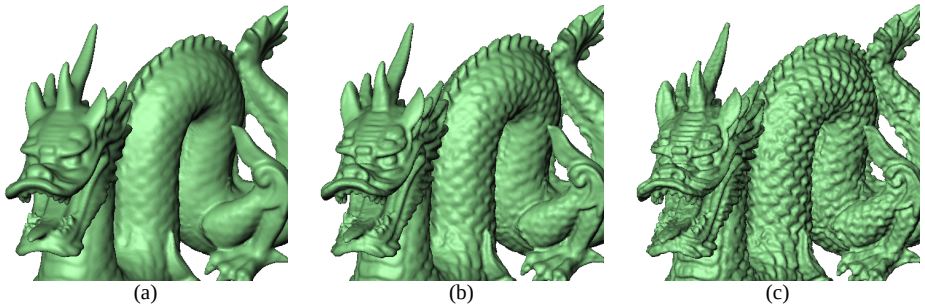


Fig. 8. (a) Original model, (b) after 1, and (c) 2 iterations of high-boost filtering.

5. HIGH-BOOST FILTERING

The surface processing framework introduced in Section 3 is flexible and allows for the implementation of even more general image processing methods. We demonstrate this by describing how to generalize image enhancement by high-boost filtering of surfaces.

A high-boost filter has a frequency transform that amplifies high frequency components. In image processing, this can be achieved by *unsharp masking* [Gonzalez and Woods 1992]. Let the low-pass filtered version of an image I be $\tilde{I}$. The high-frequency components are $I_{hf} = I - \tilde{I}$. The high-boost output is the sum of the input image and some fraction of its high-frequency components:

$$I_{\text{out}} = I + \alpha I_{hf} = (1+\alpha)I - \alpha\tilde{I}, \tag{21}$$

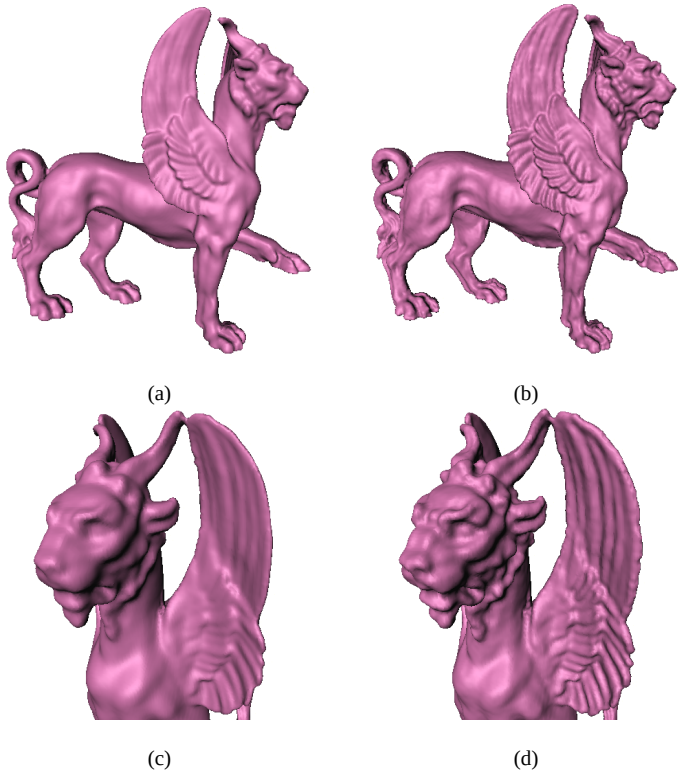


Fig. 9. High-boost filtering: (a) original model, (b) after filtering, (c) close-up of original, and (d) filtered model.

where α is a positive constant that controls the amount of high-boost filtering.

This same algorithm applies to surface normals by a simple modification to the flow chart in Figure 3. Recall that the $d\mathscr{G}/d\mathbf{N}$ loop produces $\mathbf{N}^{n+1}$. Define a new set of normal vectors by

$$\mathbf{N}' = \frac{(1+\alpha)\mathbf{N}^n - \alpha\mathbf{N}^{n+1}}{||(1+\alpha)\mathbf{N}^n - \alpha\mathbf{N}^{n+1}||}. \tag{22}$$

This new normal map is then input to the $d\mathscr{D}/d\phi$ refitting loop. The effect of (22) is to extrapolate from the previous set of normals in the direction opposite to the set of normals obtained by isotropic diffusion. Recall that isotropic diffusion will smooth areas with high curvature and not significantly affect already smooth areas. Processing the loop with the modification of (22) will have the effect of increasing the curvature in areas of high curvature, while leaving smooth areas relatively unchanged. Thus, we are able to obtain high quality surface enhancement on fairly complex surfaces of arbitrary topology, as can be observed in Figs. 8 and 9. The scales on the skin and the ridge back are enhanced. Also, note that different amounts of enhancement can be achieved by controlling the number of iterations of the main loop. The degree of low-pass filtering used to obtain $\mathbf{N}^{n+1}$ controls the size of the features that are enhanced. Figure 9 shows another example of high-boost filtering; notice the enhancement of features particularly on the wings.

6. DISCUSSION

The *natural* generalization of image processing to surfaces is via the normals. The lowest-order differential invariant of images is the gradient magnitude, and minimizing a quadratic penalty of this quantity produces the diffusion equation, which gives rise to Gaussian blurring. The lowest-order differential invariants of surface shape are the principal curvatures. Likewise, the curvature of a 3D surface is a function of the gradient of the surface normal as shown in Section 3.2. In this light, total curvature, which is the Euclidean norm of the Jacobian of the vector field of surface normals, is the natural generalization of the squared-gradient-magnitude smoothness penalty for images. *Thus, for surfaces, first variation of the isotropic total curvature penalty, rather than MCF, is the equivalent of Gaussian blurring.*

Variational processes on the surface curvature have corresponding variational formulations on the surface normals. The generalization of image-processing to surface normals, however, requires that we process the normals using a metric on the surface manifold, rather than a simple, flat metric, as we do with images. By processing the normals separately, we can solve a pair of coupled second-order equations instead of a fourth-order equation. Typically, we allow one equation (the surface) to lag the other, but as the lag gets very small, it should not matter. In this framework, the diffusion of the surface normals (and corresponding motions of the surface) is equivalent to the particular fourth-order flow that minimizes the surface total curvature penalty function.

The method generalizes because we can do virtually anything we wish with the normal map. A generalization of anisotropic diffusion to a constrained, vector-valued function, defined on a manifold, gives us a smoothing process that preserves creases. If we want to enhance the surface, we can enhance the normals and refit the surface.

We solve these equations using implicit surfaces, representing the implicit function on a discrete grid, modeling the deformation with the method of level sets. This level set implementation allows us to separate the shape of the model from the processing mechanism. Because of the implementation, the method applies equally well to any surface that can be represented in a volume. Consequently, our results show a level of surface complexity that goes beyond that of previous methods.

Future work will study the usefulness of other interesting image processing techniques such as *total variation* [Rudin et al. 1992; Burchard 2002] and local contrast enhancement. To date, we have dealt with post processing surfaces, but future work will combine this method with segmentation and reconstruction techniques. The current shortcoming of this method is the computation time, which is significant. However, the process lends itself to parallelism, and the advent of cheap, specialized, vector-processing hardware promises significantly faster implementations.

ACKNOWLEDGMENTS

This work is supported by the Office of Naval Research under grant #N00014-01-10033 and the National Science Foundation under grant #CCR0092065.

REFERENCES

ADALSTEINSON, D. AND SETHIAN, J. A. 1995. A fast level set method for propogating interfaces. *J. Computational Physics*, 269–277.

ALEXA, M. 2002. Wiener filtering of meshes. In *Shape Modeling International.* 51–57.

BALLESTER, C., BERTALMIO, M., CASELLES, V., SAPIRO, G., AND VERDERA, J. 2001. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Trans. on Image Processing 10*, 8 (August), 1200–1211.

BERTALMIO, M., CHENG, L.-T., OSHER, S., AND SAPIRO, G. 2001. Variational problems and partial differential equations on implicit surfaces. *J. Computational Physics 174*, 759–780.

BURCHARD, P. 2002. Total variation geometry 1: Concepts and motivation. Tech. rep., UCLA CAM-02-01. January.

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

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *Proceedings of IEEE Visualization.* 397–405.

CURLESS, B. AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH'96*.

DESBRUN, M., MEYER, M., SCHRÖDER, M., AND BARR, A. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH'99*. 317–324.

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

DOCARMO, M. P. 1976. *Differential Geometry of Curves and Surfaces.* Prentice Hall.

EBERLY, D. 1996. *Ridges in Image and Data Analysis.* Kluwer Academic Publishers.

GONZALEZ, R. C. AND WOODS, R. E. 1992. *Digital Image Processing.* Addison-Wesley Publishing Company.

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

HALSTEAD, M., KASS, M., AND DEROSE, T. 1993. Efficient, fair interpolation using catmull-clark surface. In *Proceedings of SIGGRAPH'93*. 35–44.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH'98*. 105–114.

LORENSON, W. AND CLINE, H. 1987. Marching cubes: A high resolution 3d surface reconstruction algorithm. In *Proceedings of SIGGRAPH'87*. 163–169.

LORIGO, L., FAUGERAS, O., GRIMSON, E., KERIVEN, R., KIKINIS, R., NABAVI, A., AND WESTIN, C.-F. 2000. Co-dimension 2 geodesic active contours for the segmentation of tubular strucures. In *Proceedings of Computer Vision and Pattern Recognition.*

MALLADI, R., SETHIAN, J. A., AND VEMURI, B. C. 1995. Shape modeling with front propagation: A level set approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence 17*, 2, 158–175.

MORETON, H. P. AND SÉQUIN, C. H. 1992. Functional optimization for fair surface design. In *Proceedings of SIGGRAPH'92*. 167–176.

OHTAKE, Y., BELYAEV, A. G., AND BOGAEVSKI, I. A. 2000. Polyhedral surface smoothing with simultaneous mesh regularization. In *Geometric Modeling and Processing.*

OSHER, S. AND SETHIAN, J. A. 1988. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulation. *J. Computational Physics 79*, 12–49.

PENG, D., MERRIMAN, B., OSHER, S., ZHAO, H.-K., AND KANG, M. 1999. A pde based fast local level set method. *J. Computational Physics 155*, 410–438.

PERONA, P. 1998. Orientation diffusion. *IEEE Trans. on Image Processing 7*, 3, 457–467.

PERONA, P. AND MALIK, J. 1990. Scale space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence 12*, 7 (July), 629–639.

POLDEN, A. 1997. Compact surfaces of least total curvature. Tech. rep., University of Tubingen, Germany.

RUDIN, L., OSHER, S., AND FATEMI, C. 1992. Nonlinear total variation based noise removal algorithms. *Physica D 60*, 259–268.

SAPIRO, G. 2001. *Geometric Partial Differential Equations and Image Analysis.* Cambridge University Press, Chapter Geometric Curve and Surface Evolution.

SCHNEIDER, R. AND KOBBELT, L. 2000. Generating fair meshes with g^1 boundary conditions. In *Geometric Modeling and Processing Proceedings.* 251–261.

SCHNEIDER, R. AND KOBBELT, L. 2001. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design 18*, 359–379.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge University Press.

TANG, B., SAPIRO, G., AND CASELLES, V. 2000. Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case. *International Journal of Computer Vision 36,* 2, 149–161.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Geometric surface smoothing via anisotropic diffusion of normals. In *Proceedings of IEEE Visualization.* 125–132.

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

TAUBIN, G. 2001. Linear anisotropic mesh filtering. Tech. Rep. RC22213, IBM Research Division. October.

TAUBIN, G., ZHANG, T., AND GOLUB, G. 1996. Optimal surface smoothing as filter design. In *European Conf. on Computer Vision.* Vol. 1. 283–292.

WELCH, W. AND WITKIN, A. 1992. Variational surface modeling. In *Proceedings of SIGGRAPH'92*. 157–166.

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

WHITAKER, R. T. 1994. Volumetric deformable models: Active blobs. In *Visualization In Biomedical Computing,* R. A. Robb, Ed. SPIE, 122–134.

WHITAKER, R. T. 1998. A level-set approach to 3D reconstruction from range data. *Int. J. Computer Vision 29,* 3, 203–231.

A. MATHEMATICAL FOUNDATION

In this section, we will derive the equivalence of the proposed algorithm, in the limit, to minimizing the original energy function $\mathscr{G}_\phi$ defined in (10). Let us rewrite this energy function by observing that the principal curvatures are functions of the derivatives of ϕ

$$\mathscr{G}_\phi = \int_U G(\phi) \parallel \nabla\phi \parallel \, d\mathbf{x}. \tag{23}$$

Let $d\phi : \Re^3 \rightarrow \Re$ be a volume of incremental changes applied to $\phi : \Re^3 \rightarrow \Re$. The change to $\mathscr{G}$ induced by $d\phi$ can be expressed as the volume integral of the total derivative of the penalty function, $dG(\phi) \parallel \nabla\phi \parallel$, which is the product of $d\phi$ and the variation of the penalty function with respect to ϕ

$$d\mathscr{G}_\phi = \int_U \frac{d\left(G \parallel \nabla\phi \parallel\right)}{d\phi} \, d\phi \, d\mathbf{x}. \tag{24}$$

Applying the product rule to $\frac{d(G\|\nabla\phi\|)}{d\phi}$, we obtain

$$d\mathscr{G}_\phi = \underbrace{\int_U \frac{dG}{d\phi} \parallel \nabla\phi \parallel \, d\phi \, d\mathbf{x}}_{d\mathscr{G}_{\phi,1}} + \underbrace{\int_U G \frac{d \parallel \nabla\phi \parallel}{d\phi} \, d\phi \, d\mathbf{x}}_{d\mathscr{G}_{\phi,2}}. \tag{25}$$

The total derivative $dG(\phi) \parallel \nabla\phi \parallel$ can be written in terms of the surface normals by using the equality

$$\frac{dG}{d\phi} \, d\phi = \frac{dG}{d\mathbf{N}} \cdot d\mathbf{N}, \tag{26}$$

given that the normal map is a function of ϕ. Then, the first term in (25) can be written as

$$d\mathscr{G}_{\phi,1} = \int_U \frac{dG}{d\mathbf{N}} \cdot d\mathbf{N} \parallel \nabla\phi \parallel \, d\mathbf{x}. \tag{27}$$

To simplify (27), we derive $d\mathbf{N}$ as a function of $\mathbf{N}$ and $d\nabla\phi$

$$d\mathbf{N} = d\frac{\nabla\phi}{\| \nabla\phi \|} = d\frac{\nabla\phi}{(\nabla\phi \cdot \nabla\phi)^{1/2}} \tag{28}$$

$$= \frac{d\nabla\phi}{\| \nabla\phi \|} - \frac{(d\nabla\phi \cdot \mathbf{N})\,\mathbf{N}}{\| \nabla\phi \|}. \tag{29}$$

Equation (29) follows from (28) by using the chain rule for the differentiation, and substituting $\mathbf{N}$ back for $\nabla\phi / \| \nabla\phi \|$. Substituting (29) for $d\mathbf{N}$ in (27), we get

$$d\mathscr{G}_{\phi,1} = \int_U \left(\frac{dG}{d\mathbf{N}} \cdot d\nabla\phi - (d\nabla\phi \cdot \mathbf{N}) \ \frac{dG}{d\mathbf{N}} \cdot \mathbf{N} \right) \ d\mathbf{x}. \tag{30}$$

We are only interested in processes that maintain the unit length constraint of the normal map; therefore, $dG/d\mathbf{N} \cdot \mathbf{N} = 0$, and (30) is reduced to

$$d\mathscr{G}_{\phi,1} = \int_U \frac{dG}{d\mathbf{N}} \cdot \nabla d\phi \ d\mathbf{x}, \tag{31}$$

where we also use the linearity of differentiation to make the substitution $d\nabla\phi = \nabla d\phi$. We treat this energy minimization as an *adiabatic* problem, which means that energy flow across the boundary of U is zero. Hence, using *Neumann* boundary conditions for U and integration by parts, we obtain

$$d\mathscr{G}_{\phi,1} = \int_U \nabla \cdot \frac{dG}{d\mathbf{N}} \, d\phi \ d\mathbf{x}. \tag{32}$$

We now examine the second term in (25), $d\mathscr{G}_{\phi,2}$. As in Section 3.2, we treat G as a function of $\mathbf{N}$; therefore, due to the decoupling between $\mathbf{N}$ and ϕ, G can be considered independent of ϕ. Using this assumption, we can rewrite $d\mathscr{G}_{\phi,2}$ as

$$d\mathscr{G}_{\phi,2} = \int_U \frac{dG^{\mathbf{N}} \| \nabla\phi \|}{d\phi} \, d\phi \ d\mathbf{x}, \tag{33}$$

where the superscript on $G^{\mathbf{N}}$ is to mean that G is fixed with respect to ϕ. Taking the first variation of $dG^{\mathbf{N}} \| \nabla\phi \|$ yields

$$d\mathscr{G}_{\phi,2} = \int_U \nabla \cdot \left(G^{\mathbf{N}} \frac{\nabla\phi}{\| \nabla\phi \|} \right) \ d\phi \ d\mathbf{x}, \tag{34}$$

where we use the fact that $d \| \nabla\phi \| / d\nabla\phi = \nabla\phi / \| \nabla\phi \|$. Finally, combining equations (24), (32), and (34), we can derive the desired relationship between the variations with respect to ϕ and $\mathbf{N}$

$$\frac{d\,(G \| \nabla\phi \|)}{d\phi} = \nabla \cdot \left(\frac{dG}{d\mathbf{N}} + G^{\mathbf{N}} \frac{\nabla\phi}{\| \nabla\phi \|} \right). \tag{35}$$

Let us now consider the flow achieved by processing (15) and (17) back to back in one iteration of the main loop in Figure 3 again. At the beginning of iteration n, the normals are computed from ϕ^n. If we evolve the normals for one step according to (15), instead of processing them multiple iterations, the new normals are

$$\mathbf{N}^{n+1} = \mathbf{N}^n - \frac{dG}{d\mathbf{N}}, \tag{36}$$

where we write $\frac{dG}{d\mathbf{N}}$ instead of $\frac{d\mathscr{G}}{d\mathbf{N}}$ because we are referring to the update for $\mathbf{N}$ at a specific point in space. If we immediately apply (17) to fit ϕ to this new normal map, we get

$$\frac{dD}{d\phi} = H^{\phi^n} - \nabla \cdot \left(\mathbf{N}^n - \frac{dG}{d\mathbf{N}} \right), \tag{37}$$

where D is the local function defined in (16). Because $\mathbf{N}^n$ is derived directly from ϕ^n, we have $\nabla \cdot \mathbf{N} = H^{\phi^n}$, which gives the rule in our algorithm to make up this infinitesimal lag:

$$\frac{dD}{d\phi} = \nabla \cdot \frac{dG}{d\mathbf{N}}. \tag{38}$$

Comparing with (35), we find the rule to descend on the energy as a function of ϕ

$$\frac{d\left(G \parallel \nabla\phi \parallel\right)}{d\phi} = \frac{dD}{d\phi} + \nabla \cdot \left(G^{\mathbf{N}} \frac{\nabla\phi}{\parallel \nabla\phi \parallel} \right). \tag{39}$$

This establishes the mathematical foundation of our method. However, in our experiments, we have found that the contribution of the second term is very small and it does not change the results qualitatively. Therefore, we drop it for the sake of computational efficiency, and implement only $\frac{dD}{d\phi}$ as described in Section 3.

B. NUMERICAL IMPLEMENTATION

By embedding surface models in volumes, we have converted equations that describe the movement of surface points to nonlinear PDEs defined on a volume. The next step is to discretize these PDEs in space and time. In this paper, the embedding function ϕ is defined on the volume domain U and time. The PDEs are solved using a discrete sampling with forward differences along the time axis.

For brevity, we will discuss the numerical implementation in 2D— the extension to 3D is straightforward. The function $\phi : U \mapsto \Re$ has a discrete sampling $\phi[p,q]$, where $[p,q]$ is a grid location and $\phi[p,q] = \phi(x_p, y_q)$. We will refer to a specific time instance of this function with superscripts, i.e. $\phi^n[p,q] = \phi(x_p, y_q, t_n)$. In our calculations, we need three different approximations to first-order derivatives: forward, backward, and central differences. We denote the type of discrete difference using superscripts on a difference operator, i.e., $\delta^{(+)}$ for forward differences, $\delta^{(-)}$ for backward differences, and δ for central differences. For instance, the differences in the x direction on a discrete grid with unit spacing are

$$\begin{aligned} \delta_x^{(+)}\phi[p,q] &\triangleq \phi[p+1,q] - \phi[p,q], \\ \delta_x^{(-)}\phi[p,q] &\triangleq \phi[p,q] - \phi[p-1,q], \text{and} \\ \delta_x\phi[p,q] &\triangleq \frac{\phi[p+1,q] - \phi[p-1,q]}{2}. \end{aligned} \tag{40}$$

The application of these difference operators to vector-valued functions denotes componentwise differentiation.

In describing the numerical implementation, we will refer to the flow chart in Figure 3 for one iteration of the main loop. Hence, the first step in our numerical implementation is the calculation of the surface normal vectors from ϕ^n. Recall that the surface is a level set of ϕ^n as defined in (7). Hence, the surface normal vectors can be computed as the unit

vector in the direction of the gradient of ϕ^n. The gradient of ϕ^n is computed with central differences as

$$\nabla\phi^n \approx \begin{pmatrix} \delta_x\phi^n \\ \delta_y\phi^n \end{pmatrix}; \tag{41}$$

and the normal vectors are initialized as

$$\mathbf{N}^{u=0} = \nabla\phi^n / ||\nabla\phi^n|| . \tag{42}$$

Because ϕ^n is fixed and allowed to lag behind the evolution of $\mathbf{N}$, the time steps in the evolution of $\mathbf{N}$ are denoted with a different superscript, u. For this evolution, $\partial\mathbf{N}/\partial t = -d\mathscr{G}/d\mathbf{N}$, which is derived in (15).

We now describe how to numerically compute $d\mathscr{G}/d\mathbf{N}$. This computation is implemented with smallest support area operators, which use the smallest possible neighborhood of voxels to compute the required output. The Laplacian of a function can be computed in two steps by first applying the gradient operator and then the divergence operator. In 2D, the gradient of the normals produces a 2×2 matrix, which we call the *flux* matrix. Next, the divergence operator collapses the flux matrix to a 2×1 vector. The "columns" of the flux matrix are computed independently as

$$\overset{x+}{\mathbf{M}}{}^u \approx \delta_x^{(+)}\mathbf{N}^u - \left(\delta_x^{(+)}\phi^n\right)\overset{x+}{\mathbf{C}}{}^u, \tag{43}$$

$$\overset{y+}{\mathbf{M}}{}^u \approx \delta_y^{(+)}\mathbf{N}^u - \left(\delta_y^{(+)}\phi^n\right)\overset{y+}{\mathbf{C}}{}^u \tag{44}$$

where the time index n remains fixed as we increment u. The positions of $\mathbf{M}_u$, which is computed with forward differences, are staggered off the grid by half a pixel, see Figure 10. For instance, $\overset{x+}{\mathbf{M}}{}^u$ uses information from positions $[p+1,q]$ and $[p,q]$; hence, it exists at $[p+1/2,q]$. We use the following notation: for some function α, $\overset{x+}{\alpha}$, and $\overset{y+}{\alpha}$ will denote the function computed at $[p+1/2,q]$ and $[p,q+1/2]$, respectively.

To compute the intrinsic derivatives of $\mathbf{N}^u$ on the level sets of ϕ^n, $(\delta_x^{(+)}\phi^n)\,\overset{x+}{\mathbf{C}}{}^u$, and $(\delta_y^{(+)}\phi^n)\,\overset{y+}{\mathbf{C}}{}^u$ are subtracted from the regular derivatives of $\mathbf{N}^u$. The variables $\overset{x+}{\mathbf{C}}{}^u$ and $\overset{y+}{\mathbf{C}}{}^u$ are computed as follows

$$\overset{x+}{\mathbf{C}}{}^u = \overset{x+}{\nabla}\mathbf{N}^u \cdot \overset{x+}{\nabla}\phi^n \,/\, \|\overset{x+}{\nabla}\phi^n\|^2, \tag{45}$$

$$\overset{y+}{\mathbf{C}}{}^u = \overset{y+}{\nabla}\mathbf{N}^u \cdot \overset{y+}{\nabla}\phi^n \,/\, \|\overset{y+}{\nabla}\phi^n\|^2, \tag{46}$$

where the matrix $\nabla\mathbf{N}$ is as defined in (12). In (45), the dot product between the matrix $\overset{x+}{\nabla}\mathbf{N}^u$ and the vector $\overset{x+}{\nabla}\phi^n \,/\, \|\overset{x+}{\nabla}\phi^n\|^2$ denotes the vector whose components are the dot products of the rows of $\overset{x+}{\nabla}\mathbf{N}^u$ and the vector $\overset{x+}{\nabla}\phi^n \,/\, \|\overset{x+}{\nabla}\phi^n\|^2$. The same applies to (46). The variables (45) and (46) must be computed at the same locations as $\overset{x+}{\mathbf{M}}{}^u$ and $\overset{y+}{\mathbf{M}}{}^u$, respectively. These computations are done with the smallest support area operators, using the symmetric 2×3 grid of samples around each staggered point. For instance, the staggered gradients of ϕ,

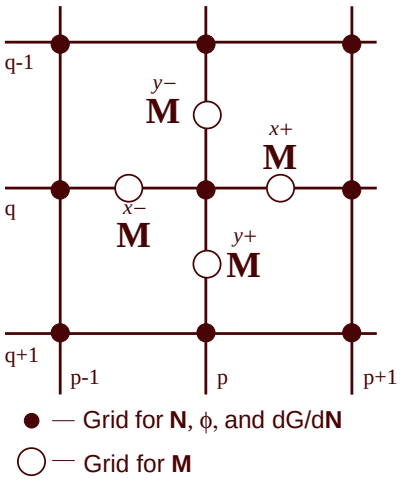


Fig. 10. Computational grid.

which are needed for the evaluating (45) and (46), are computed as

$$\begin{aligned}
\overset{x+}{\nabla}\phi^n = \nabla\phi^n[p+\frac{1}{2},q] &\approx \begin{pmatrix} \delta_x^{(+)}\phi[p,q] \\ \frac{1}{2}\left(\delta_y\phi[p,q]+\delta_y\phi[p+1,q]\right) \end{pmatrix}, \\
\overset{y+}{\nabla}\phi^n = \nabla\phi^n[p,q+\frac{1}{2}] &\approx \begin{pmatrix} \frac{1}{2}\left(\delta_x\phi[p,q]+\delta_x\phi[p,q+1]\right) \\ \delta_y^{(+)}\phi[p,q] \end{pmatrix}
\end{aligned} \tag{47}$$

The staggered gradient matrices of the normals, $\overset{x+}{\nabla}\mathbf{N}^u$ and $\overset{y+}{\nabla}\mathbf{N}^u$, which are also needed for evaluating (45) and (46), are computed with the same stencil.

After the computation of the flux, backwards differences are used to compute the divergence operation in (15). For isotropic diffusion,

$$\Delta^u = \delta_x^{(-)}\,\overset{x+}{\mathbf{M}}{}^u + \delta_y^{(-)}\,\overset{y+}{\mathbf{M}}{}^u, \tag{48}$$

and from (15)

$$-\left[\frac{d\mathscr{G}}{d\mathbf{N}}\right]^u \approx \left(\mathbf{I}-\mathbf{N}^u\otimes\mathbf{N}^u\right)\Delta^u = \Delta^u - \left(\Delta^u\cdot\mathbf{N}^u\right)\mathbf{N}^u. \tag{49}$$

The results of the backwards differencing are defined at the original ϕ grid location $[p,q]$ because they undo the forward staggering in the flux locations. Therefore, both components of Δ and thus $d\mathscr{G}/d\mathbf{N}$ are located on the original grid for ϕ.

To evaluate (15) for anisotropic diffusion, we also need to compute $g(\kappa^2)$ at the precise locations where the flux (43) and (44) are located. Hence, we compute the total intrinsic curvature of the normals

$$\begin{aligned}
\overset{x+}{\kappa}{}^2 &= \|\overset{x+}{\nabla}\mathbf{N}^u\|^2 - \overset{x+}{\mathbf{C}}{}^u\cdot\overset{x+}{\mathbf{C}}{}^u, \\
\overset{y+}{\kappa}{}^2 &= \|\overset{y+}{\nabla}\mathbf{N}^u\|^2 - \overset{y+}{\mathbf{C}}{}^u\cdot\overset{y+}{\mathbf{C}}{}^u,
\end{aligned} \tag{50}$$

where $\|\cdot\|^2$ is the Euclidean norm, the sum of the squares of all elements of the matrix.

Then, the divergence for anisotropic diffusion is computed as

$$\Delta^u = \delta_x^{(-)} \left[g(\overset{x+}{\kappa^2}) \overset{x+}{\mathbf{M}^u} \right] + \delta_y^{(-)} \left[g(\overset{y+}{\kappa^2}) \overset{y+}{\mathbf{M}^u} \right], \tag{51}$$

and the tangential projection is applied to this vector as in (49).

Starting with the initialization in (42) for $u = 0$, we iterate

$$\mathbf{N}^{u+1} = \mathbf{N}^u - \left[\frac{d\mathscr{G}}{d\mathbf{N}} \right]^u \tag{52}$$

for a fixed number of steps, 25 iterations for the examples in this paper. In other words, we do not aim at minimizing the energy given in (14) in the $d\mathscr{G}/d\mathbf{N}$ loop of Figure 3; we only reduce it. The minimization of total mean curvature as a function of ϕ is achieved by iterating the main loop in Figure 3.

Once the evolution of $\mathbf{N}$ is concluded, ϕ is refitted to the new normal vectors according to (17). We denote the evolved normals by $\mathbf{N}^{n+1}$. To solve (17) we must calculate H^ϕ and $H^{\mathbf{N}^{n+1}}$, which is the induced mean curvature of the normal map; in other words, it is the curvature of the hypothetical target surface that fits the normal map. Curvature from a field of normals is given by

$$H^{\mathbf{N}^{n+1}} \approx \delta_x \mathbf{N}_{(x)}^{n+1} + \delta_y \mathbf{N}_{(y)}^{n+1}, \tag{53}$$

where we have used central differences on the components of the normal vectors that are denoted by the subscripts (x) and (y). The quantity $H^{\mathbf{N}^{n+1}}$ needs to be computed once at initialization as the normal vectors remain fixed during the refitting phase. Let v be the time step index in the $d\mathscr{D}/d\phi$ loop. H^{ϕ^v} is the mean curvature of the moving level set surface at time step v and is calculated from ϕ with the smallest area of support

$$H^{\phi^v} \approx \delta_x^{(-)} \frac{\delta_x^{(+)} \phi^v}{\overset{x+}{\|\nabla \phi^v\|}} + \delta_y^{(-)} \frac{\delta_y^{(+)} \phi^v}{\overset{y+}{\|\nabla \phi^v\|}} \tag{54}$$

where the gradients in the denominators are staggered to match the locations of the forward differences in the numerator. The staggered gradients of ϕ in the denominator are calculated using the 2×3 neighborhood as in (47).

The PDE in (17) is solved with a finite forward differences, but with the upwind scheme for the gradient magnitude [Osher and Sethian 1988], to avoid overshooting and maintain stability. The up-wind method computes a one-sided derivative that looks in the up-wind direction of the moving wave front, and thereby avoids overshooting. Moreover, because we are interested in only a single level set of ϕ, solving (17) over all of U is not necessary. Different level sets evolve independently, and we can compute the evolution of ϕ only in a narrow band around the level set of interest and re-initialize this band as necessary [Adalsteinson and Sethian 1995; Peng et al. 1999]. See [Sethian 1999] for more details on numerical schemes and efficient solutions for level set methods.

Using the upwind scheme and narrow band methods, ϕ^{v+1} is computed from ϕ^v according to (17) using the curvatures computed in (53) and (54). This loop is iterated until the energy in (16) ceases to decrease; let v_{final} denote the final iteration of this loop. Then we set ϕ for the next iteration of the main loop (see Figure 3) as $\phi^{n+1} = \phi^{v_{final}}$ and repeat the entire procedure. The number of iterations of the main loop is a free parameter that generally determines the extent of processing.