Geometric Surface Processing via Normal Maps[a]

Tolga Tasdizen
Univ. of Utah
tolga@cs.utah.edu

Ross Whitaker
Univ. of Utah
whitaker@cs.utah.edu

Paul Burchard
UCLA
burchard@pobox.com

Stanley Osher
UCLA
sjo@math.ucla.edu

UUCS-02-03

[a]Submitted to SIGGRAPH'02 for review as a research paper.

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

January 17, 2002

Abstract

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 framework, smooth surfaces minimize the norm of the derivative of the surface normals—i.e. total curvature. Penalty functions on the surface normals are computed using geometry-based shape metrics and minimized using gradient descent. This produces a set of partial differential equations (PDE). In this paper, we introduce a novel framework for implementing geometric processing tools for surfaces using a two step algorithm: (i) operating on the normal map of a surface, and (ii) manipulating the surface to fit the processed normals. The computational approach uses level set surface models; therefore, the processing does not depend on any underlying parameterization. Iterating this two-step process, we can implement geometric fourth-order flows efficiently by solving a set of coupled second-order PDEs. This paper will demonstrate that the framework 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.

Chapter 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 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.

The state of the art in surface processing includes a number of very useful tools for processing meshes. However, to date there *is no general framework for geometric surface processing*. By *general* we mean two things. First, the framework should provide a broad variety of capabilities, including surface processing tools that resemble the state of the art in image processing algorithms. Second the framework should apply to a general class of surfaces. Users should be able to process complex surfaces of arbitrary 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.

This paper presents a framework that 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. In this light, the differences between surface processing and image processing are threefold. Normals live on a manifold (the surface) and cannot necessarily be processed using a flat metric, as is typically done with

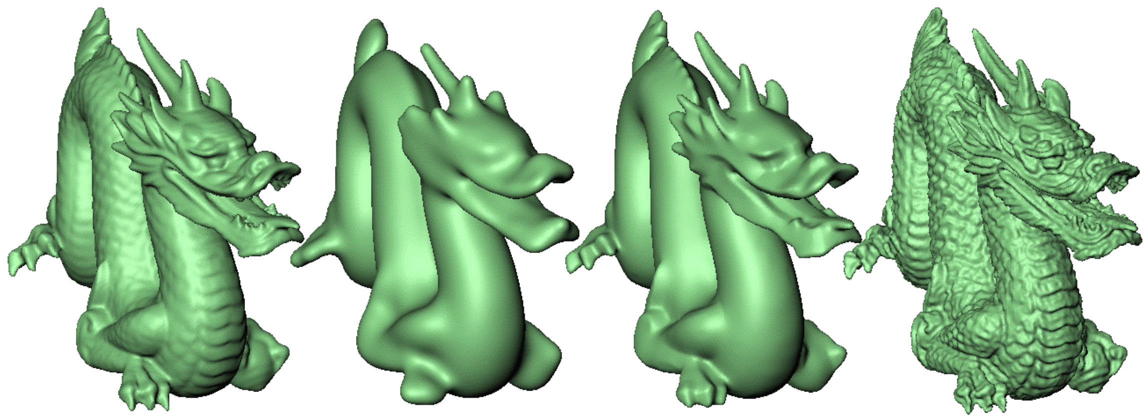


Figure 1.1: Surface processing examples.

images. Normals are vector valued and constrained to be unit length; the processing techniques must accommodate this. Normals are coupled with the surface shape, and 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 partial differential equations (PDE). This strategy enables us to achieve the "black box" behavior, which is reflected in the nature of the results. Results in this paper will show a level of complexity in the models and the processing algorithms that has not yet been demonstrated in the literature. This generality 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.

Figure 1 shows several results of processing a 3D surface model with different algorithms. These algorithms consist of smoothing, feature-preserving smoothing, and surface enhancement. All three processes are consistent mathematical generalizations of their image-processing counterparts. Note that all of the surfaces in this paper are represented volumetrically and rendered using the marching cubes algorithm [1].

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 [2], which are not yet possible with the proposed representation. However, in other applications, such as 3D segmentation and surface reconstruction [3, 4], the processing is data driven, and surfaces can deform quite far from their initial shapes and even change topology. Furthermore, when considering processes other than isotropic smoothing, such as nonlinear smoothing or high-boost filtering, the creation or sharpening of small features can exhibit noticeable effects of the mesh topology—features that are aligned with the mesh are treated differently than those that are not. The techniques presented in this paper offer a new set of capabilities that are especially interesting when processing measured data—as are all of the examples show in this paper.

The specific contributions of this paper are:

- a novel framework for geometric processing of surfaces that relies on surface normals;
- a numerical method for solving fourth-order level set equations in two simpler steps, thereby avoiding the explicit computation of unstable high-order derivatives; and
- examples of three geometric processing algorithms with applications to data sets that are more complex than those previously demonstrated in the literature.

Chapter 2

Related Work

The majority of surface processing research has been in the context of *surface fairing* with the motivation of smoothing surface models to create aesthetically pleasing surfaces using triangulated meshes [5, 6, 7, 8]. Surface fairing typically operate by minimizing a fairness or penalty function that favors smooth surfaces [9, 10, 11, 5]. Fairness functionals can depend on the geometry of the surface or the parameterization. Geometric functionals 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. Simpler parameterization dependent functionals are linear approximations to geometric invariants. Such functionals can be equivalent to geometric invariants when the surface parameterization is *isometric*) or they can be poor approximations when the parameterization is irregular and non-differentiable. An *isometric* surface parameterization requires the two parameter coordinate axis to be orthogonal and arc-length parameterized. In the context of surface fairing with meshes these concepts are also referred to as geometric and parameterization smoothness [7] or outer and inner fairness [12].

One way to smooth a surface is to incrementally reduce its surface area. This can be accomplished by mean curvature flow (MCF) at every point S:

$$\frac{\partial S}{\partial t} = \mathrm{H}\vec{\mathrm{N}} \tag{2.1}$$

where H is the mean curvature of the surface, $\vec{N}$ is the surface normal, and t is the time evolution of the surface shape. For parameterized surfaces the surface area translates to the membrane energy functional

$$\int_\Omega X_u^2 + X_v^2 \; du \; dv \tag{2.2}$$

where $X(u,v)$ and Ω are the surface parameterization and its domain, respectively. For an isometric parameterization $X_u^2 + X_v^2 = 1$; therefore, (2.2) reduces to surface area. However, for larger and smaller $X_u^2 + X_v^2$, the approximation to surface area is distorted proportionally. The variational derivative of (2.2) is the Laplacian,

$$\Delta X = X_{uu} + X_{vv}, \tag{2.3}$$

which is equivalent to mean curvature in the isometric case. Laplacian, or mean curvature flow, is closely tied to Gaussian filtering—a standard method of smoothing images. Clarenz *et al.* [13] propose a meshed-based intrinsic flow that incorporates a weighted sum principle curvatures that depends on the local surface shape.

A second-order penalty function is *total curvature*

$$\int_S \kappa_1^2 + \kappa_2^2 \; dS \tag{2.4}$$

which has been shown to deform surfaces into spheres [14]. Total curvature is a geometric (invariant) property of the surface. The mesh fairing approach of [5] which minimizes (2.4) involves fitting 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 basis provide equally good representations. In [8] the authors search directly for an intrinsic PDE that produces fair surfaces instead of deriving the PDEs from a variational framework. They propose the Laplacian of mean curvature $\Delta_B \mathrm{H} = 0$ for meshes where Δ_B is the Laplace-Beltrami operator, i.e. the Laplacian for parameterized surfaces. Their approach is not sufficiently general to satisfy the goals of this paper, but is closely related to the proposed method. We will discuss it further in Sec. 3.

If we penalize the parameterization (i.e. non-geometric), equation (2.4) becomes the thin plate energy functional

$$\int_\Omega X_{uu}^2 + 2X_{uv}^2 + X_{vv}^2 \; du \; dv \tag{2.5}$$

where X and Ω are as defined for (2.2). Thin plate energy was used in [10] for surface fairing. The variational derivative of (2.5) is the biharmonic operator, which is linear:

$$\Delta^2 X = X_{uuuu} + 2X_{uuvv} + X_{vvvv}. \tag{2.6}$$

These linear energy functionals underly the signal processing approach to surface fairing pioneered in [6], who derived the natural vibration frequencies of a surface from the Laplacian operator. Taubin observes that Gaussian filtering causes shrinkage. He eliminates this problem by designing a low pass filter using a weighted average of the Laplacian and the biharmonic operator. The weights have to 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. Indeed, any polynomial transfer function in the frequency domain can be implemented with this method [15]. [16] describe a related approach in which surface are smoothed by simultaneously solving the membrane (2.2) and thin plate (2.5) energy functionals.

The signal processing approach uses the umbrella operator which is a discretization of the Laplacian. The edge lengths connecting the nodes of the mesh and the angles between adjacent edges around a node, also known as face angles, introduce parameterization dependencies. By setting the edge weights in the umbrella operator to the reciprocal of the edge length, the dependency on edge length can be removed [6], but the dependency on the face angles remain. A scale dependent intrinsic umbrella operator is defined in [7] that removes both dependencies. The time steps in explicitly integrating a scale dependent umbrella operator are proportional to the square of the shortest edge length. Desbrun *et al.* overcome this limitation by introducing an implicit integration scheme. Nevertheless, the weights for the umbrella operator must be recomputed at each iteration to maintain its intrinsic property. A non-uniform relaxation operator is introduced in [2] to minimize a locally weighted quadratic energy of second order differences.

Moreton and Sequin [9] propose a geometric fairness functional that penalizes the variation of principle curvatures—a third-order, geometric penalty function (corresponding to a *sixth-order* variational derivative), which requires very large computation times. The analysis and implementation of general penalty functions above second order remains an open problem, which is beyond the scope of this paper. Evidence in this paper and elsewhere [7, 8] suggests that fourth-order geometric flows form a sufficient foundation for a general, geometric surface processing system.

This work in this paper is also related to that of Chopp & Sethian [17], who derive the intrinsic Laplacian of curvature for an implicit curve, and solve the resulting fourth-order nonlinear PDE. However, 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. One of the results of this paper is to solve this equation more effectively and to demonstrate that this is only one example of a more general class of useful surface processing techniques. Joint interpolation of vector fields and gray level functions was used for succesfully filling-in missing parts of images in [18].

Chapter 3

Geometric Surface Processing

One of the underlying strategies of this paper 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 [12], where the influence of the mesh parameterization on surface fairing results is clearly shown, and higher-order geometric flows, such as the intrinsic Laplacian of curvature, are proposed as the solution.

As an illustration of the importance of higher-order geometric processing, consider the results in Fig. 3.1, which demonstrates the differences between processing surfaces with mean curvature flow (MCF) and intrinsic Laplacian of mean curvature flow (ILMCF). The amount of smoothing for MCF and ILMCF was chosen to be qualitatively similar, and yet important differences can be observed on the smaller features of the original 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. Some authors [19] have proposed volume preserving forms of second-order flows, but these processes compensate by enlarging *the object as a whole*, which exhibits, qualitatively, the same behavior on small features. Intrinsic Laplacian of mean curvature flow, in Fig. 3.1, preserves the structure of these features much better *while* smoothing them.

An alternative to solving a fourth-order equation directly is to decouple it into a pair of second-order equations. For instance, a two-step solution to ILMCF for meshes is proposed in [8]. However, this approach works only for meshes, and relies on analytic properties of the steady-state solutions, $\Delta H = 0$, by fitting surface primitives that have those properties.

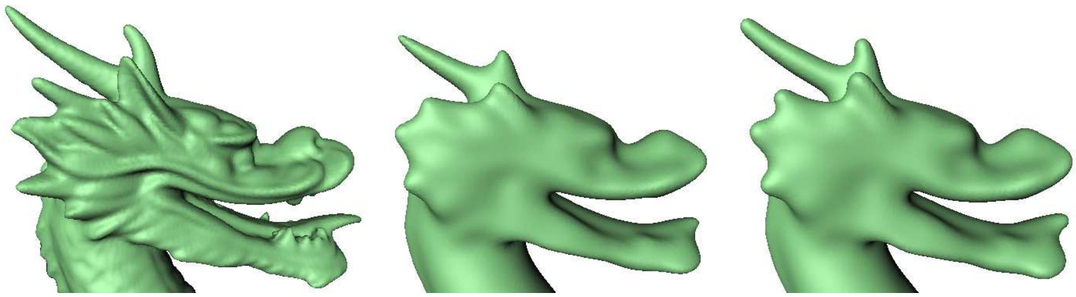


Figure 3.1: Second- and fourth-order surface smoothing. From left to right: Original model, mean curvature flow, and intrinsic Laplacian of mean curvature flow.

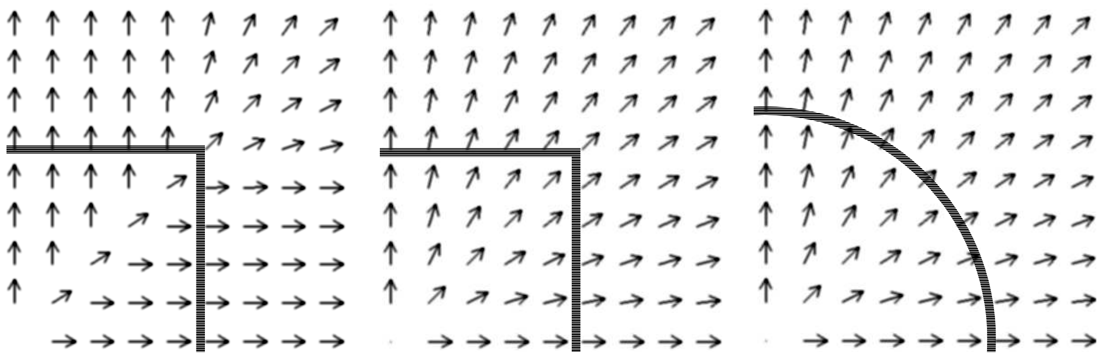


Figure 3.2: Shown here in 2D, the process begins with a shape and constructs a normal map from the distance transform (left), modifies the normal map according to a PDE derived from a penalty function (center), and re-fits the shape to the normal map (right).

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 those that minimize nonlinear feature-preserving penalties.

In [18], 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. This forms a high-order interpolation function, which is shown to be useful for image inpainting—recovering missing patches of data in 2D images. This 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. The formulation proposed in this paper 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.

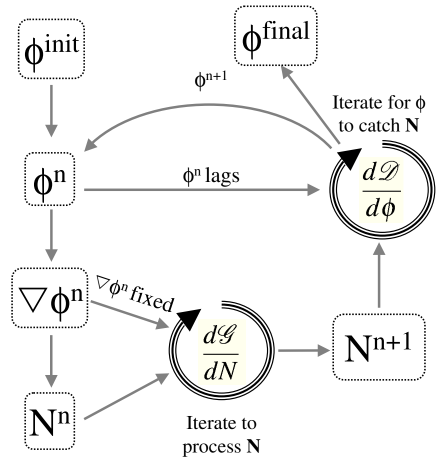


Figure 3.3: Flow chart

In this section, we will generalize [17] to the intrinsic Laplacian of mean curvature for level set surfaces, and introduce a method for breaking it into two simpler PDEs. This pair of equations is solved by allowing the surface shape to lag the normals as they are filtered and then catch up by a separate process. Figure 3.2 shows these three step process graphically in 2D—shapes give rise to normal maps, which, when filtered, give rise to new shapes. In the limit, this approach is equivalent to solving the full-blown, intrinsic fourth-order flow, but it generalizes to a wide range of processes and makes no assumptions about the shapes of the solutions. In Sec. 4, we show results for several different kinds of surface filters. The appendix describes a numerically stable discrete solver for the system of equations.

3.1 Level set methods

The remainder of this paper addresses the higher-order geometry of implicit surfaces, i.e. level sets. Even though very good surface fairing results have been obtained with meshes, there are some drawbacks to using meshes and other parametric models for purposes other than simple smoothing (i.e. low pass filtering). We believe that level set methods [20, 21] are better suited for a wide range of surface-processing for the following reasons.

1. Some surface processes, such as anisotropic diffusion and high-boost filtering, have the capability of introducing new features—a fundamental difference from smoothing. Meshes do not form discontinuities well unless the edges of the face triangles coincide with the edges of the surface.

2. Anything more than a modest amount of smoothing is likely to evolve the surface far from its original shape. This requires the creation and deletion of faces in meshes to maintain the original resolution. Implicit surfaces do not exhibit this problem.
3. For surface reconstruction, such as from range imagery or tomographic data, the evolving surface can undergo topological changes [3, 4]. Surface meshes do not (readily) allow topological changes, whereas level set surfaces do.

To facilitate the discussion, we use the Einstein notation convention, where the subscripts indicate tensor indices, and repeated subscripts within a product represent a summation over the index (across the dimensions of the underlying space). Furthermore, we use the convention that subscripts on quantities represent derivatives, except where they are in parenthesis, in which case they refer to a vector-valued variable. Thus, ϕ_i is the gradient vector of a scalar quantity $\phi : \mathbb{R}^n \mapsto \mathbb{R}$. The Hessian is ϕ_{ij}, and the Laplacian is ϕ_{ii}. A vector field is $v_{(i)}$, where $v : \mathbb{R}^n \mapsto \mathbb{R}^n$, and the divergence of that field is $v_{(i)i}$. Scalar operators, such as differentials behave in the usual way. Thus, gradient magnitude is $|\phi_i| = \sqrt{\phi_i \phi_i}$ and the differential for a coordinate system is $dx_{(i)} = dx_1 dx_2 \ldots dx_n$.

Level set surface models rely on the notion of a regular surface, which is a collection of 3D points, $\mathscr{S}$, with a topology that allows each point to be modeled *locally* as a function of two variables. We can describe the deformation of such a surface using the 3D velocity of each of its constituent points, i.e., $\partial s_{(i)}(t)/\partial t$ for all $s_{(i)} \in \mathscr{S}$. If we represent the surface implicitly at each time t, then

$$\mathscr{S} = \left\{ s_{(i)}(t) \mid \phi\left(s_{(i)}(t), t\right) = 0 \right\}. \tag{3.1}$$

Notice that 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 $\mathscr{S}$, 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

$$\frac{\partial \phi}{\partial t} = -\phi_j \frac{\partial s_{(j)}}{\partial t} \tag{3.2}$$

Notice that ϕ_j is proportional to the surface normal, and thus $\partial s_{(j)}/\partial t$ affects ϕ only in the direction of the surface normal— surface motion in any other direction is merely a change in the parameterization. Using this framework, the PDE on ϕ that describes the motion of a surface by mean curvature, as in (2.1), is

$$\frac{\partial \phi}{\partial t} = \phi_{ii} - \frac{\phi_{ij}\phi_i\phi_j}{\phi_k\phi_k}. \tag{3.3}$$

The following sections give the mathematical description of the process outlined in Fig. 3.3. First we will derive the energy functional for minimizing the total curvature of a normal map. The first variation of the total curvature results in a second-order PDE on the normals. This process is denoted in Fig. 3.3 as the $d\mathscr{G}/d\mathbf{N}$ loop. Next, in Sec. 3.3, we show another energy functional for fitting a surface to an evolved normal map, resulting in a second-order PDE on ϕ. This is the $d\mathscr{D}/d\phi$ loop in Fig. 3.3. Finally, we will show that the overall process of simultaneously solving these two PDEs as shown in Fig. 3.3 is equivalent to ILMCF. This establishes the mathematical foundation of the proposed method.

3.2 Intrinsic Laplacian of mean curvature flow for normal maps

When using implicit representations one must account for the fact derivatives of functions defined on the surface are computed by projecting their 3D derivatives onto the tangent plane of the surface. Let $N_{(i)} : \mathbb{R}^3 \to S^3$ be the normal map, which is a field of normals that are everywhere perpendicular to the family of embedded isosurfaces of ϕ—thus $N_{(i)} = \phi_i / \sqrt{\phi_k \phi_k}$. The 3×3 projection matrix for the implicit surface normal is $P_{(ij)} = N_{(i)} N_{(j)}$, and $P_{(ij)} V_{(i)}$ returns the projection of $V_{(i)}$ onto $N_{(i)}$. Let $I_{(ij)}$ be the identity matrix. Then the projection onto the plane that is perpendicular to the vector field $N_{(i)}$ is the *tangent projection operator*, $T_{(ij)} = I_{(ij)} - P_{(ij)}$. Under normal circumstances the normal map $N_{(i)}$ is derived from the gradient of ϕ, and $T_{(ij)}$ projects vectors onto the tangent planes of the level sets of ϕ. However, the computational strategy we are proposing allows ϕ to lag the normal map. Therefore, the tangent projection operators differ, and we use $T^{\phi}_{(ij)}$ to denote projections onto the tangent planes of the level sets of ϕ and $T^{N}_{(ij)}$ to denote projections onto planes perpendicular to the normal map.

The shape matrix [22] of a surface describes its curvature independent of the parameterization. The shape matrix of an implicit surface is obtained by differentiating the normal map and projecting that derivative onto the tangent plane of the surface. The Euclidean norm of the shape matrix is the total curvature, κ. Thus

$$\kappa^2 = \left|\left| N_{(i)j} T^{\phi}_{(jk)} \right|\right|^2. \tag{3.4}$$

If $N_{(i)}$ is derived directly from ϕ, this gives

$$\kappa^2 = \left|\left| T^{\phi}_{(il)} \phi_{lj} T^{\phi}_{(jk)} \right|\right|^2. \tag{3.5}$$

The strategy is to treat these expressions as penalty functions, and develop surface processing methods by solving the PDEs that result from the first variation of surface integrals over these penalty functions. Notice, that if we take the first variation of (3.5) we obtain a fourth-order PDE on ϕ, but if we take the first variation of (3.4), with respect to $N_{(i)}$, allowing ϕ to remain fixed, we obtain a second-order PDE on $N_{(i)}$. To see the first variation of (3.4) we re-express the norm, using the identity $||A||^2 = \mathrm{Trace}[\mathrm{A}^\mathrm{T}\mathrm{A}]$. This gives

$$\kappa^2 = \left|\left|N_{(i)j}\right|\right|^2 - \frac{\left|\left|N_{(i)j}\phi_j\right|\right|^2}{\phi_k \phi_k}, \tag{3.6}$$

and thus the penalty function for the total surface curvature is

$$\mathscr{G}\left(N_{(i)}\right) = \int_{\mathscr{S}} \left[\left|\left|N_{(i)j}\right|\right|^2 - \frac{\left|\left|N_{(i)j}\phi_j\right|\right|^2}{\phi_k \phi_k}\right] dx_{(m)}. \tag{3.7}$$

As we process the normal map $N_{(i)}$, letting ϕ lag, we must ensure that it maintains the unit length constraint, $N_{(i)}N_{(i)} = 1$. This is expressed in the penalty function using Lagrange multipliers. The constrained penalty is

$$\mathscr{G}\left(N_{(i)}\right) + \int_{\mathscr{S}} \lambda(x_{(l)}) \left(N_{(k)}N_{(k)} - 1\right) dx_{(l)}, \tag{3.8}$$

where $\lambda(x_{(l)})$ is the Lagrange multiplier at $x_{(l)}$. Using the first variation and solving for λ introduces a projection operator $T^N_{(ij)}$ on the first variation of $\mathscr{G}$, which keeps $N_{(i)}$ unit length. Thus the first variation of (3.8) with respect to $N_{(i)}$ is

$$-T^N_{(ij)} \frac{d\mathscr{G}}{dN_{(j)}} = -T^N_{(ij)} \left[N_{(j)k} - \frac{N_{(j)l}\phi_l \phi_k}{\phi_m \phi_m}\right]_k. \tag{3.9}$$

A gradient descent on this metric $\partial N_{(i)}/\partial t = -d\mathscr{G}/dN_{(i)}$, results in a PDE that a smoothes the normal map by minimizing total curvature while maintaining the unit normal constraint. Notice that this is precisely the same as solving the constrained diffusion equation on $N_{(i)}$ using the method of solving PDEs on implicit manifolds described in [23]. This derivation is what we consider the *base case*—subsequent sections of this paper will introduce other processes on the normal maps.

3.3 Surface evolution via normal maps

We have shown how to evolve the normals to minimize total squared curvature; however, the final goal is the reconstruction of the surface which requires evolving ϕ, rather than the normal map. Hence, the next step is to relate the evolution of ϕ to the evolution of $N_{(i)}$. Suppose that we are given the normal map $N_{(i)}$ to some set of surfaces, but not necessarily level sets of ϕ—as is the case if we filter $N_{(i)}$ and let ϕ lag. We can manipulate ϕ so that it *catches* up to $N_{(i)}$ by minimizing a penalty function that quantifies the discrepancy. This penalty function is

$$\mathscr{D}(\phi) = \int_U \left[\phi_i\phi_i - \phi_i N_{(i)}\right] dx_{(j)}, \tag{3.10}$$

where $U \subset \mathbb{R}^3$ is the domain of ϕ. A gradient descent on ϕ that minimizes this penalty function is

$$\frac{\partial\phi}{\partial t} = -\frac{d\mathscr{D}}{d\phi} = ||\phi_k|| \left[\left(\frac{\phi_j}{\phi_m\phi_m}\right)_j - N_{(j)j}\right] = |\nabla\phi| \left[\mathrm{H}^\phi - \mathrm{H}^\mathrm{N}\right] \tag{3.11}$$

where H^ϕ is the mean curvature of the level set surface and H^N is the induced curvature of the normal map. Thus, the surface moves as the difference between its own curvature and that of the normal field. The factor of $|\nabla\phi|$, which is typical with level set formulations, comes from the fact that we are manipulating the shape of the level set, which is embedded in ϕ, as in (3.2).

We propose to solve fourth-order flows on the level sets of ϕ by a splitting strategy, which entails processing the normals and allowing ϕ to lag and then catch up later, in a separate process. This is only correct if we know that in the limit, as the lag becomes very small, we are solving the full fourth-order PDE. To show this we analyze one iteration of the main loop in Fig. 3.3. Before processing the normals, they are derived from ϕ^n, and we have $N^n_{(i)} = \phi^n_i/\sqrt{\phi^n_j\phi^n_j}$. Evolving the normal map once according to (3.9) for a small amount of time dt gives

$$N^{n+1}_{(i)} = N^n_{(i)} - T^N_{(ij)}\frac{d\mathscr{G}}{dN_{(j)}}dt. \tag{3.12}$$

If we immediately apply (3.11) to fit ϕ to this new normal map

$$\frac{\partial\phi}{\partial t} = \parallel \phi^n_k \parallel \left(\mathrm{H}^\phi - \left[\mathrm{N}^\mathrm{n}_{(\mathrm{i})} - \mathrm{T}^\mathrm{N}_{(\mathrm{ij})}\frac{\mathrm{d}\mathscr{G}}{\mathrm{dN}_{(\mathrm{j})}}\mathrm{dt}\right]_\mathrm{i}\right). \tag{3.13}$$

Because $N^n_{(i)}$ is derived directly from ϕ^n, we have $N^n_{(i)i} = \mathrm{H}^\phi$, which gives the expression for changes in ϕ in order to make up this infinitesimal lag:

$$\frac{\partial\phi}{\partial t} = - \parallel \phi^n_k \parallel \left[T^N_{(ij)}\frac{d\mathscr{G}}{dN_{(j)}}\right]_i. \tag{3.14}$$

We can express the penalty function $\mathscr{G}$ in terms of either $N_{(i)}$ or ϕ, and take the first variation with respect to either of these quantities. The relationship between $d\mathscr{G}/dN_{(i)}$ and $d\mathscr{G}/d\phi$ is established by integrating (by parts)—it is

$$\frac{d\mathscr{G}}{d\phi} = -\left[T^{\phi}_{ij} \frac{d\mathscr{G}}{dN_{(j)}} \right]_i , \tag{3.15}$$

where $T^{\phi}_{ij} = T^{N}_{ij}$ as $dt \to 0$. Thus the update on ϕ after it lags $N_{(i)}$ by some small amount is actually

$$\frac{\partial \phi}{\partial t} = \parallel \phi_k \parallel \frac{d\mathscr{G}}{d\phi}, \tag{3.16}$$

which is a gradient descent, of the level sets of ϕ, on $\mathscr{G}$— the total curvature of the surface.

This analysis shows that the strategy depicted in Figs. 3.2 and 3.3 is a valid mechanism for implementing the base-case, which is ILMCF on implicit, level set surfaces. However, this strategy has broader implications. Sec. 4.2 will show that other choices of $\mathscr{G}$ (there are many options from the image processing literature) will produce different kinds of PDE-based surface processing algorithms. Furthermore, Sec. 4.2.1 will demonstrate that the general strategy of processing normals separately from surfaces, in a two phase process, offers a set of useful tools that go beyond the strict variational framework.

Chapter 4

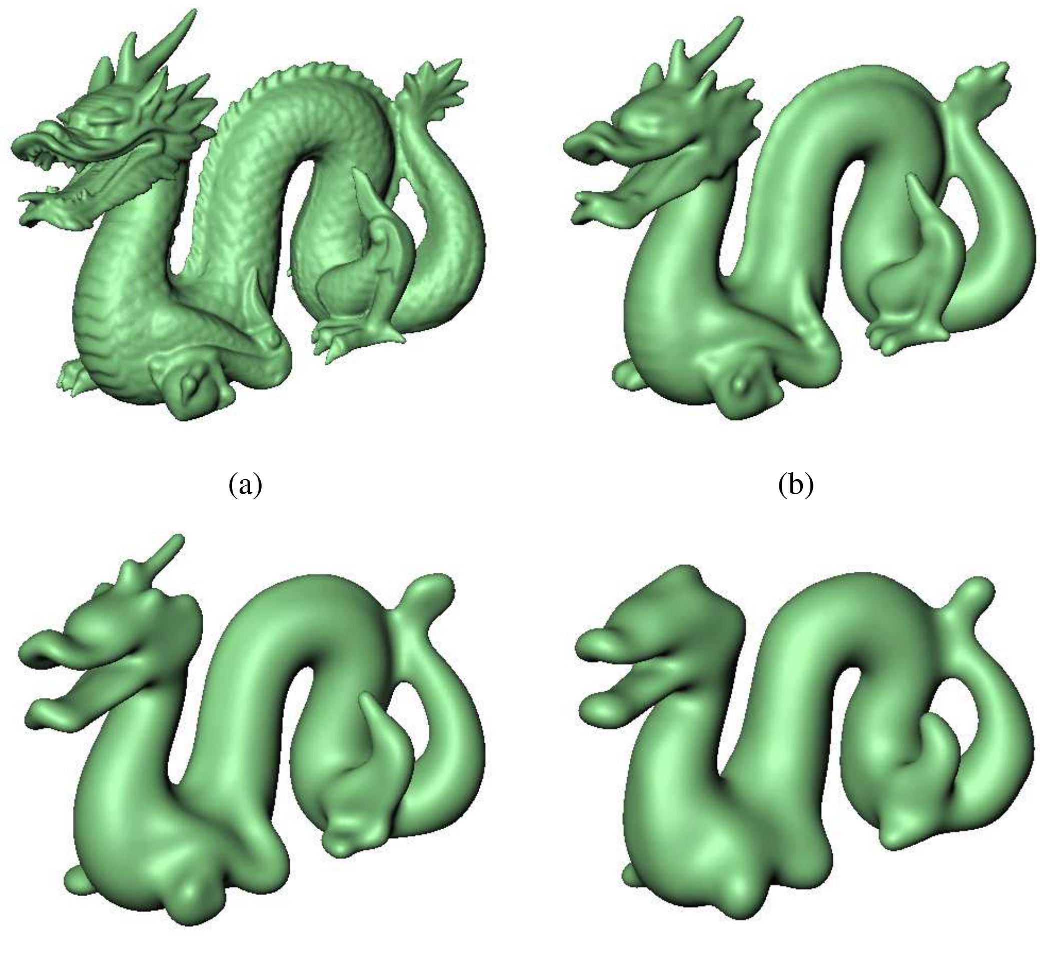
Applications

4.1 Isotropic Diffusion

We have discussed ILMCF in detail in Sec. 3. Here we will present some results of this diffusion process. Figure 4.1 shows model at various stages of the diffusion process. Two iterations of the main processing loop in Fig. 3.3 are enough to remove the smallest scale features from the model. Later iterations start smoothing the global shape of the model. The model shown in this example consists of a $356 \times 161 \times 251$ volume. The computation time required for one iteration of the main processing loop for this model is around 30 minutes on a *1.7 Ghz Intel processor*. The models used in the examples in the rest of this paper approximately have the same level of complexity.

4.2 Anisotropic Diffusion

Minimizing the total squared curvature of a surface 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. This is equivalent to a convolution of the surface normals with a Gaussian kernel that is isotropic (using a metric that is flat on the surface). Isotropic diffusion is not particularly effective if the goal is to de-noise 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),



(a) (b)

(c) (d)

Figure 4.1: Various stages of isotropic diffusion: (a) original model, (b) after 2 iterations, (c) after 8 iterations, and (d) after 25 iterations.

in which the measurements are inherently noisy. Figure 4.2(a) is 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. As an indication of this complexity, consider that marching cubes produces 543,000 triangles from this volume. This is over 10 times the number of faces in many of the "standard" models used to demonstrate surface processing algorithms [2, 7]. The strategy in this paper is to treat such surfaces as they are measured, in their volumetric representation, rather than process this data indirectly via a simplified, fitted surface mesh [24].

Isotropic diffusion, shown in Fig. 4.2(b), is marginally effective for de-noising the head surface. Notice that the sharp edges around the eyes, nose, lips and ears are lost in this process. The problem of preserving features while smoothing away noise has been studied extensively in computer vision. Anisotropic diffusion introduced in [25] has been very successful in dealing with this problem in a wide range of images. Perona & Malik proposed to replace Laplacian smoothing, which is equivalent to the heat equation $\partial I/\partial t = \nabla \cdot \nabla I$, with a nonlinear PDE:a

$$\partial I/\partial t = \nabla \cdot [g\,(\| \nabla I \|)\, \nabla I], \tag{4.1}$$

where I is generally the grey-level image, and $g(\cdot)$ is the edge stopping function, which should be a decreasing sigmoidal function. Perona & Malik 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. A mathematical analysis shows that solutions to (4.1) can actually exhibit an inverse diffusion near edges, and can enhance or sharpen smooth edges that have gradients greater than μ [19].

Using anisotropic diffusion for surface processing with meshes was proposed in [13]. In that work, the authors move the surface according to a weighted sum of principle curvatures, where the weighting depends on the surface geometry. This is not a variational formulation, and is thereby not a generalization of (4.1). Because it performs a convex re-weighting of principle curvatures, it can reduce smoothing, but cannot exhibit sharpening of features. Furthermore, the level set implementation we are proposing allows us to produce results on significantly more complex surface shapes.

The generalization of anisotropic diffusion as in (4.1) to surfaces is achieved from variational principles by minimizing the penalty function

$$\mathscr{G} = \int_{\mathscr{S}} e^{-\frac{\kappa^2}{2\mu}} dx_{(i)}, \tag{4.2}$$

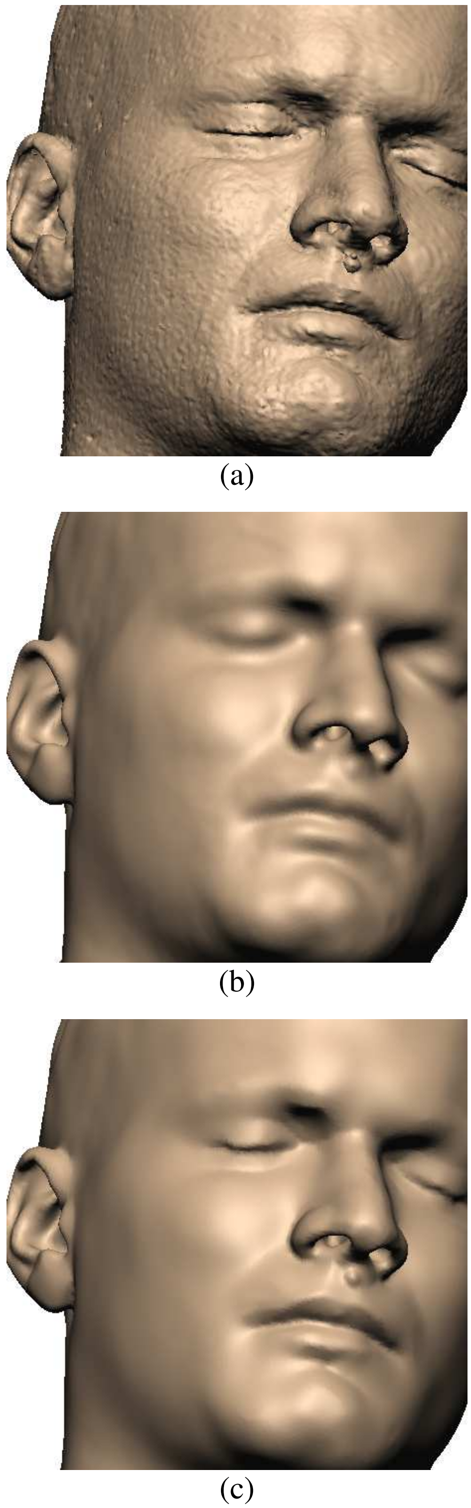


Figure 4.2: Processing results on the MRI head model: (a) original isosurface, (b)) isotropic diffusion of surface normals, and (c) anisotropic diffusion of surface normals. The small protrusion under the nose is a physical marker used for registration.

where κ^2 is computed from $N_{(i)}$ according to (3.4). The first variation with respect to the surface normals gives gives a vector-valued anisotropic diffusion on the level set surface—a straightforward generalization of (4.1). If we take the first variation with respect to ϕ, we obtain a fourth-order PDE, which is a modified version of ILMCF that preserves or enhances areas of high curvature, which we will call *creases*. Creases are the generalization of edges to surfaces. We will solve this variational problem by solving the second-order PDE on the normals using the framework introduced in Sec. 3. The strategy is the same as that in Fig. 3.3, where we replace $d\mathscr{G}/dN$ with the first variation of the penalty function from (4.2). This gives

$$T^N_{(ij)} \frac{d\mathscr{G}}{dN_{(j)}} = T^N_{(ij)} \left[g(\kappa) \left(N_{(j)k} - \frac{N_{(j)l}\phi_l\phi_k}{\phi_m\phi_m} \right) \right]_k \tag{4.3}$$

where κ is the total curvature as in (3.4), and the edge-stopping function becomes

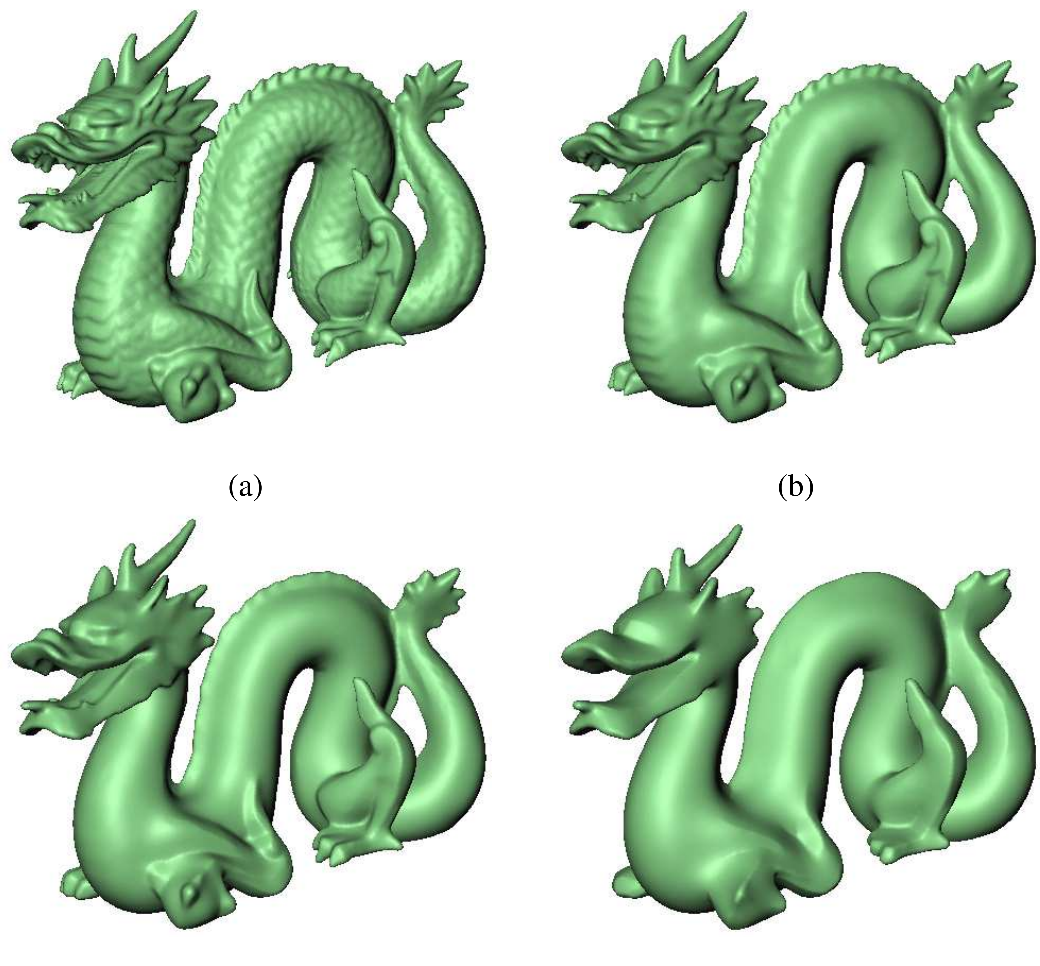
$$g(\kappa) = e^{-\frac{\kappa^2}{2\mu^2}}. \tag{4.4}$$

The rest of the process shown in Fig. 3.3 remains unchanged.

De-noising of measurement data is one of the most important applications of anisotropic diffusion. The differences between anisotropic diffusion and isotropic diffusion can clearly be observed in Fig. 4.2(c). Around the smooth areas of the original model such as the forehead and the cheeks, there is no noticeable difference in the results of the two processes. 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.

Running anisotropic diffusion for more time steps produces surfaces that have piecewise smooth normals (analogous to the behavior of the Perona and Malik equation for images), which corresponds to smooth, almost planar, surface patches bounded by sharp creases. Figure 4.3 shows the evolution of the dragon model with anisotropic diffusion. After a few iterations of the overall loop in Fig. 3.3 the smallest details, such as the scales of the dragon's skin, disappear, as in Fig. 4.3(b). After some more iterations in Fig. 4.3(c), the finest details of the face and the spikes on the ridge of the back have been suppressed. Going even further, observe that the surface is starting to become polyhedral in Fig. 4.3(d).

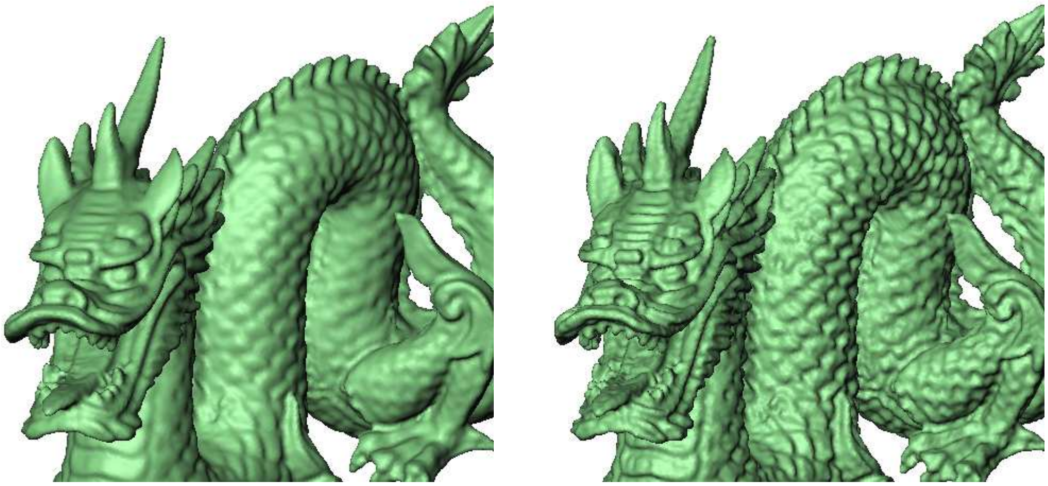
The non-linear progression of elimination of details from the smallest scale to the largest in Fig. 4.3 suggests a possibility for improved surface compression strategies, where details above a desired scale are kept effectively unchanged. This is in sharp contrast to isotropic diffusion and mean curvature flow, which distorts all features and generates sharp discontinuities as features disappear.



(a) (b)

(c) (d)

Figure 4.3: Various stages of anisotropic diffusion: (a) original model, (b) after 2 iterations, (c) after 6 iterations, and (d) after 26 iterations.



(a) (b)

Figure 4.4: Enhanced surface (a) after 1, and (b) 2 iterations of high-boost filtering.

These results support our proposition that processing the normals of a surface is the natural generalization of image processing. Processing normals on the surface is the same as processing a vector-valued image (such as a color image) with a distance metric that varies over the domain. This is in contrast to processing the points on a surface directly with lower-order flows, which offers no clear analogy to methods established in image processing.

4.2.1 High-boost filtering

The surface processing framework introduced in Sec. 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 to surfaces.

A high-boost filter has a frequency transform that amplifies high frequency components. In image processing this can be achieved by *unsharp masking* [26]. Let the low-pass filtered version of an image I be $\tilde{I}$. The high-frequency components are $\hat{I} = I - \tilde{I}$. The high-boost output is sum of the input image and some fraction of its high-frequency components:

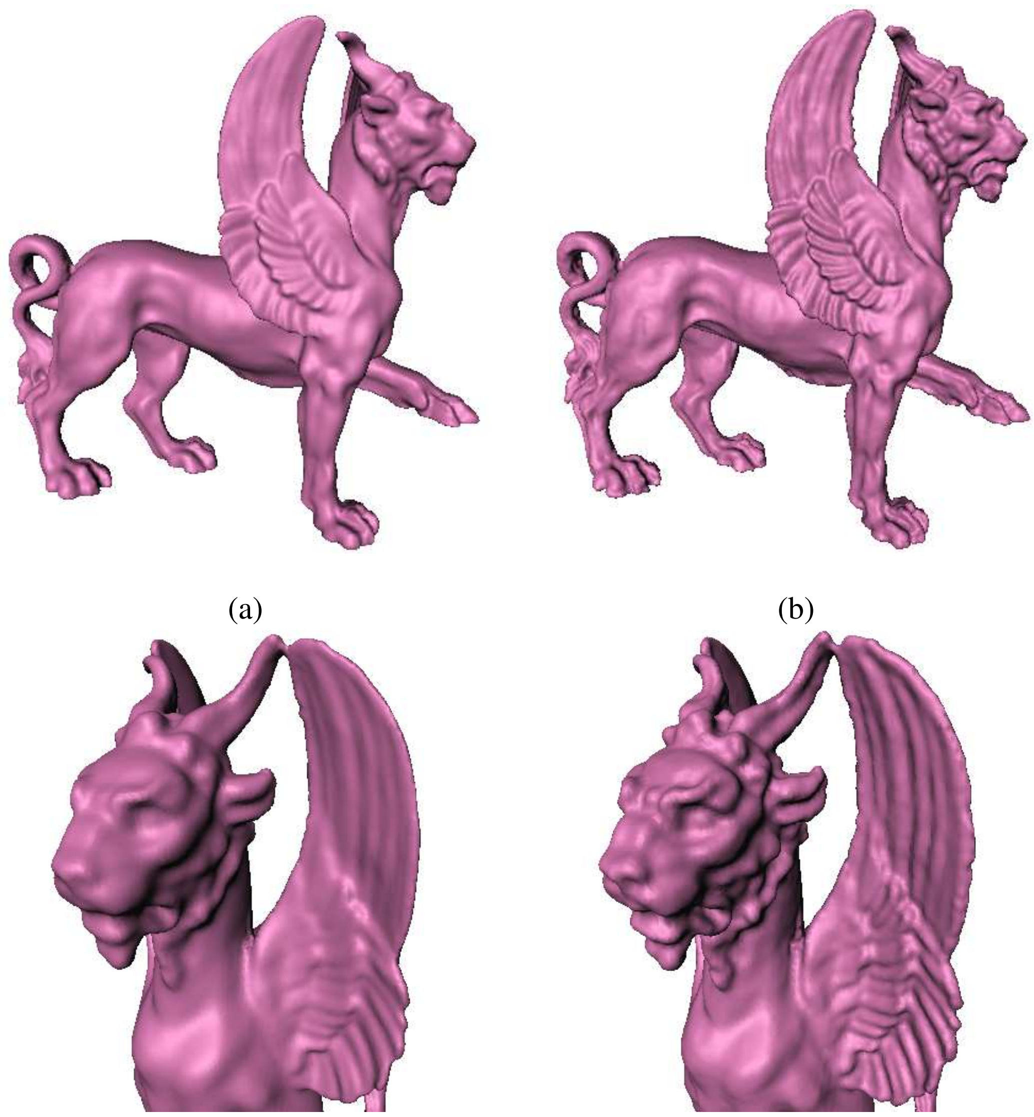
$$I_{\text{out}} = I + \alpha \hat{I} = (1+\alpha)I - \alpha \tilde{I}, \tag{4.5}$$

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 Fig. 3.3. Recall that the $d\mathscr{G}/dN$ loop produces $N^{n+1}_{(i)}$. Define a new set of normal vectors by

$$N'_{(i)} = \frac{(1+\alpha)N^{n}_{(i)} - \alpha N^{n+1}_{(i)}}{||(1+\alpha)N^{n}_{(j)} - \alpha N^{n+1}_{(j)}||}. \tag{4.6}$$

This new normal map is then input to the $d\mathscr{D}/d\phi$ catch-up loop. The effect of (4.6) 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 (4.6) 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 in Figs. 4.4 and 4.5. The effects of high-boost filtering can be observed by comparing the original dragon model in Fig. 4.3 with the high-boost filtered model in Fig. 4.4. 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 $N^{n+1}_{(i)}$ controls the size of the features that are enhanced. Figure 4.5 shows another example of high-boost filtering; notice the enhancement of features particularly on the wings.



(a) (b) (c) (d)

Figure 4.5: High-boost filtering: (a) original model, (b) after filtering, (c) close-up of original, and (d) filtered model.

Chapter 5

Conclusion

The *natural* generalization of image processing to surfaces is via the normals. Variational processes on the surface 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. In this framework, the diffusion of the surface normals (and corresponding motions of the surface) is equivalent to particular fourth-order flow, the intrinsic Laplacian of mean curvature. By processing the normals separately, we can solve a pair of coupled second-order equations instead of a four-order equation. Typically, we allow one equation (the surface) to lag the other, but as the lag gets very small, it should not matter. 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.

The method generalizes because 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 allow the surface to catch up. 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* [27] 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.

Bibliography

[1] W. Lorenson and H. Cline, “Marching cubes: A high resolution 3d surface reconstruction algorithm,” in *Proceedings of SIGGRAPH’87*, pp. 163–169, 1987.

[2] I. Guskov, W. Sweldens, and P. Schroder, “Multiresolution signal processing for meshes,” in *Proceedings of SIGGRAPH’99*, pp. 325–334, 1999.

[3] R. Malladi, J. A. Sethian, and B. C. Vemuri, “Shape modeling with front propagation: A level set approach,” *IEEE Trans. on PAMI*, vol. 17, no. 2, pp. 158–175, 1995.

[4] R. T. Whitaker, “A level-set approach to 3D reconstruction from range data,” *Int. J. Computer Vision*, vol. 29, no. 3, pp. 203–231, 1998.

[5] W. Welch and A. Witkin, “Free-form shape design using triangulated surfaces,” in *Proceedings of SIGGRAPH’94*, pp. 247–256, 1994.

[6] G. Taubin, “A signal processing approach to fair surface design,” in *Proceedings of SIGGRAPH’95*, pp. 351–358, 1995.

[7] M. Desbrun, M. Meyer, M. Schroder, and A. Barr, “Implicit fairing of irregular meshes using diffusion and curvature flow,” in *Proceedings of SIGGRAPH’99*, pp. 317–324, 1999.

[8] R. Schneider and L. Kobbelt, “Generating fair meshes with g^1 boundary conditions,” in *Geometric Modeling and Processing Proceedings*, pp. 251–261, 2000.

[9] H. P. Moreton and C. H. Sequin, “Functional optimization for fair surface design,” in *Proceedings of SIGGRAPH’92*, pp. 167–176, 1992.

[10] W. Welch and A. Witkin, “Variational surface modeling,” in *Proceedings of SIGGRAPH’92*, pp. 157–166, July 1992.

[11] M. Halstead, M. Kass, and T. DeRose, “Efficient, fair interpolation using catmull-clark surface,” in *Proceedings of SIGGRAPH’93*, pp. 35–44, 1993.

[12] R. Schneider and L. Kobbelt, "Geometric fairing of irregular meshes for free-form surface design." http://www-i8.informatik.rwth-aachen.de/geom_fair.pdf; To appear in Computer Aided Geometric Design, 2001.

[13] U. Clarenz, U. Diewald, and M. Rumpf, "Anisotropic geometric diffusion in surface processing," in *Proceedings of IEEE Visualization*, pp. 397–405, 2000.

[14] A. Polden, "Compact surfaces of least total curvature," tech. rep., University of Tubingen, Germany, 1997.

[15] G. Taubin, T. Zhang, and G. Golub, "Optimal surface smoothing as filter design," in *European Conf. on Computer Vision*, vol. 1, pp. 283–292, 1996.

[16] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive multi-resolution modeling on arbitrary meshes," in *Proceedings of SIGGRAPH'98*, pp. 105–114, 1998.

[17] D. L. Chopp and J. A. Sethian, "Motion by intrinsic laplacian of curvature," *Interfaces and Free Boundaries*, vol. 1, pp. 1–18, 1999.

[18] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. on Image Processing*, vol. 10, pp. 1200–1211, August 2001.

[19] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001.

[20] S. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulation," *J. Computational Physics*, vol. 79, pp. 12–49, 1988.

[21] J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Sciences*. Cambridge University Press, 1996.

[22] M. P. DoCarmo, *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[23] M. Bertalmio, L.-T. Cheng, S. Osher, and G. Sapiro, "Variational methods and partial differential equations on implicit surfaces," *J. Computational Physics*, vol. 174, pp. 759–780, 2001.

[24] N. Litke, A. Levin, and P. Schroder, "Fitting subdivision surfaces," in *Proceedings IEEE Visualization*, pp. 319–324, 2001.

[25] P. Perona and J. Malik, "Scale space and edge detection using anisotropic diffusion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629–639, July 1990.

[26] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[27] L. Rudin, S. Osher, and C. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D*, vol. 60, pp. 259–268, 1992.

[28] D. Adalsteinson and J. A. Sethian, “A fast level set method for propogating interfaces,” *J. Computational Physics*, pp. 269–277, 1995.

[29] D. Peng, B. Merriman, S. Osher, H.-K. Zhao, and M. Kang, “A pde based fast local level set method,” *J. Computational Physics*, vol. 155, pp. 410–438, 1999.

Appendix A

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. There are two issues with discretization: (i) the accuracy and stability of the numerical solutions, (ii) the increase in computational complexity introduced by the dimensionality of the domain.

A.1 Notation

For brevity, we will discuss the numerical implementation in 2D— the extension to 3D is straightforward. The function $\phi : U \mapsto \mathbb{R}$ 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. $\phi^n[p,q] = \phi(x_p, y_q, t_n)$. For a vector in 2-space v, we use $v_{(x)}$ and $v_{(y)}$ to refer to its components consistent with the notation of Sec. 3. 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

$$\delta_x^{(+)}\phi[p,q] \stackrel{\triangle}{=} \phi[p+1,q] - \phi[p,q],$$

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

where the time superscript has been left off for conciseness. The application of these difference operators to vector-valued functions denotes componentwise differentiation.

A.2 Numerical solution to ILMCF

In describing the numerical solution to ILMCF, we will refer to the flow chart in Fig. 3.3 for one time step 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 (3.1). 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

$$\phi_i^n[p,q] \approx \left(\begin{array}{c} \delta_x\phi^n[p,q] \\ \delta_y\phi^n[p,q] \end{array} \right); \tag{A.2}$$

and the normal vectors are initialized as

$$N_{(i)}^{u=0}[p,q] = \phi_i^n[p,q] / \parallel \phi_k^n[p,q] \parallel . \tag{A.3}$$

Because ϕ^n is fixed and allowed to lag behind the evolution of $N_{(i)}$, the time steps in the evolution of $N_{(i)}$ are denoted with a different superscript, u. For this evolution, $\partial N_{(i)}/\partial t = -d\mathscr{G}/dN_{(i)}$ is implemented with smallest support area operators. For ILMCF $d\mathscr{G}/dN_{(i)}$ is given by (3.9) which we will rewrite here component by component. The Laplacian of a function can be applied in two steps, first the gradient and then the divergence. In 2D, the gradient of the normals produces a 2×2 matrix, and the divergence operator in (3.9) collapses this to a 2×1 vector. The diffusion of the normal vectors in the tangent plane of the level-sets of ϕ, requires us to compute the flux in the x and y directions, which we denote $M_{(ij)}^u = M_{(ix)}^u, M_{(iy)}^u$. The "columns" of the flux matrix are computed independently as

$$M_{(ix)}^u \approx \delta_x^{(+)} N_{(i)}^u - \left(N_{(i)j}^u \frac{\phi_j^n}{\parallel \phi_k^n \parallel} \right) \frac{\delta_x^{(+)}\phi^n}{\parallel \phi_k^n \parallel}, \tag{A.4}$$

$$M_{(iy)}^u \approx \delta_y^{(+)} N_{(i)}^u - \left(N_{(i)j}^u \cdot \frac{\phi_j^n}{\parallel \phi_k^n \parallel} \right) \frac{\delta_y^{(+)}\phi^n}{\parallel \phi_k^n \parallel} \tag{A.5}$$

where the time index n remains fixed as we increment u. Derivatives of the normal vectors are computed with forward differences; therefore they are staggered, located on a grid that

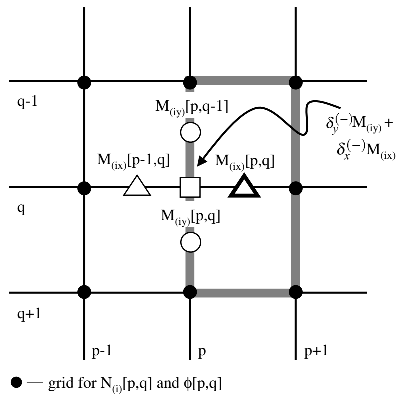


Figure A.1: Computation grid

is offset from the grid where ϕ and $N_{(i)}$ are defined, as shown Fig. A.1 for the 2D case. Furthermore, notice that since the offset is half pixel only in the direction of the differentiation, the locations of $\delta_x^{(+)} N_{(i)}$ and $\delta_y^{(+)} N_{(i)}$ are different, but are the same locations as $M_{(x)}^u$ and and $M_{(y)}^u$ respectively. To evaluate (A.4) and (A.5), $\phi_j^n \; N_{(i)j}^n$ must computed at $[p+1/2, q]$ and $[p, q+1/2]$, respectively. These computations are also done with the smallest support area operators, using the symmetric 2×3 grid of samples around each staggered point, as shown in Fig. A.1 with the heavy rectangle. For instance,

$$
\begin{aligned}
\phi_j^n[p+\frac{1}{2}, q] &\approx \begin{pmatrix} \delta_x^{(+)} \phi[p,q] \\ \frac{1}{2}\left(\delta_y \phi[p,q] + \delta_y \phi[p+1,q]\right) \end{pmatrix}, \text{ and} \\
\phi_j^n[p, q+\frac{1}{2}] &\approx \begin{pmatrix} \frac{1}{2}\left(\delta_x \phi[p,q] + \delta_x \phi[p,q+1]\right) \\ \delta_y^{(+)} \phi[p,q] \end{pmatrix}.
\end{aligned}
\tag{A.6}
$$

Backwards differences of the flux matrix are used to compute the divergence operation in (3.9)

$$
\left[\frac{d\mathcal{G}}{dN_{(i)}}\right]^u \approx -\delta_x^{(-)} M_{(ix)}^u + \delta_y^{(-)} M_{(iy)}^u \tag{A.7}
$$

Notice that backwards difference of $M_{(ix)}^u$, is defined at the original ϕ grid location $[p,q]$, and the same holds for $M_{(iy)}^u$. Thus all the components of $d\mathcal{G}/dN_{(i)}$ are located on the original grid for ϕ. Using the tangential projection operator in (3.9), the new set of normal vectors are computed as

$$
N_{(i)}^{u+1} \;\; = \;\; N_{(i)}^u + T^N \left[\frac{d\mathcal{G}}{dN_{(i)}}\right]^u
$$

$$= \quad N^u_{(i)} + \left[\frac{d\mathscr{G}}{dN_{(i)}}\right]^u - \left(\left[\frac{d\mathscr{G}}{dN_{(j)}}\right]^u N^u_{(j)}\right) N^u_{(i)}. \tag{A.8}$$

Starting with the initialization in (A.3) for $u = 0$, we iterate (A.8) for a fixed number of steps. In other words, we do not aim at minimizing the energy given in (3.8) in the $d\mathscr{G}/dN$ loop of Fig. 3.3; we only reduce it. The minimization of total mean curvature as a function of ϕ is achieved by iterating the main loop in (A.3). In Sec. 3.3, it was shown that to minimize total mean curvature, the $d\mathscr{G}/dN$ loop should be processed for only one time step before processing the $d\mathscr{D}/d\phi$ loop in every iteration of the main loop. However, the $d\mathscr{D}/d\phi$ loop is the most computationally expensive part of the algorithm and run-times can be reduced by running the main loop as few times as possible. To this effect, we found that the $d\mathscr{G}/dN$ loop can be processed for multiple time steps before moving onto the $d\mathscr{D}/d\phi$ loop. Moreover, the number of iterations of the main loop necessary to obtain the same result is reduced with this approach (25 iterations or less per main loop for the examples in this paper).

Once the evolution of N is concluded, ϕ is evolved to catch up with the new normal vectors according to (3.11). We denote the evolved normals by $N^{n+1}_{(i)}$. To solve (3.11) we must calculate H^ϕ and $H^{N^{n+1}}$. $H^{N^{n+1}}$ 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. Calculation of curvature from a field of normals is

$$H^{N^{n+1}} \approx \delta_x N^{n+1}_x + \delta_y N^{n+1}_y, \tag{A.9}$$

where we have used central differences on the components of the normal vectors. $H^{N^{n+1}}$ needs to be computed once at initialization as the normal vectors remain fixed during the cathc up 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:

$$H^{\phi^v} \approx \delta_x^{(-)} \frac{\delta_x^{(+)}\phi^v[p,q]}{\parallel \phi^v_j[p+\frac{1}{2},q] \parallel} + \delta_y^{(-)} \frac{\delta_y^{(+)}\phi^v[p,q]}{\parallel \phi^v_j[p,q+\frac{1}{2}] \parallel} \tag{A.10}$$

where the off-grid gradients of ϕ in the denominator are calculated using the 2×3 staggered neighborhood as in (A.6).

The PDE in (3.11) solved with a finite forward differences, but with the up-wind scheme for the gradient magnitude [20], 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 (3.11) over all of U is not necessary. Because different level sets evolve independently, 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 [28, 29]. See [21] for more details on numerical schemes and efficient solutions for level set methods.

Using the upwind scheme and narrow band methods, $\phi^{\nu+1}$ is computed from ϕ^{ν} according to (3.11) using the curvatures computed in (A.9) and (A.10). This loop is iterated until the energy in (3.10) ceases to decrease; let ν^{final} denote the final iteration of this loop. Then we set ϕ for the next iteration of the main loop (see Fig. 3.3) as $\phi^{n+1} = \phi^{\nu^{final}}$ and repeat the entire procedure. The number of iterations of the main loop is a free parameter that generally determines the extent of processing, i.e. the amount of smoothing for ILMCF.