

6 Isosurfaces and Level-Sets

ROSS T. WHITAKER

School of Computing University of Utah Salt Lake City, UT

Abstract

This chapter describes the basic differential geometry of isosurfaces and the method of manipulating the shapes of isosurfaces within volumes, called *level sets*. Deformable isosurfaces, implemented with level-set methods, have demonstrated a great potential in visualization for applications such as segmentation, surface processing, and surface reconstruction. This chapter begins with a short introduction to isosurface geometry, including curvature. It continues with a short explanation of the level-set partial differential equations. It also presents some practical details for how to solve these equations using upwind-scheme and sparse-calculation methods. This paper presents a series of examples of how level-set surface models are used to solve problems in graphics and vision.

6.1 Introduction

6.1.1 Motivation

This chapter describes mechanisms for analyzing and processing volumes in a way that deals specifically with *isosurfaces*. The underlying philosophy is to use isosurfaces as a modeling technology that can serve as an alternative to parameterized models for a variety of important applications in visualization and computer graphics. Level-set methods [1] rely on partial differential equations (PDEs) to model deforming isosurfaces. These methods have applications in a wide range of fields, such as visualization, scientific computing, computer graphics, and computer vision [2,3]. Applications in visualiza-

tion include volume segmentation [4,5,6], surface processing [7,8], and surface reconstruction [9,10].

This chapter presents the mathematics and numerical techniques for describing the geometry of isosurfaces and manipulating their shapes in prescribed ways. It starts with a basic introduction into the notation and fundamental concepts and then presents the geometry of isosurfaces. It then describes the method of level sets, i.e., moving isosurfaces, and presents the mathematical and numerical methods they entail.

6.1.2 Isosurfaces

6.1.2.1 Modeling Surfaces With Volumes

When considering surface models for graphics and visualization, one is faced with a staggering variety of options including meshes, spline-based patches, constructive solid geometry, implicit blobs, and particle systems. These options can be divided into two basic classes—explicit (parameterized) models and implicit models. With an implicit model, one specifies the model as a *level set* of a scalar function,

$$\phi: \begin{matrix} U & \mapsto \mathbb{R} \\ x, y, z & k \end{matrix} \quad (6.1)$$

where $U \subset \mathbb{R}^3$ is the domain of the volume (and the *range* of the surface model). Thus, a surface \mathcal{S} is

$$\mathcal{S} = \{x | \phi(x) = k\} \quad (6.2)$$

The choice of k is arbitrary, and ϕ is sometimes called the *embedding*. Notice that surfaces defined in this way divide U into a clear inside and outside—such surfaces are always closed wherever they do not intersect the boundary of the domain.

Choosing this implicit strategy begs the question of how to represent ϕ . Historically, implicit models are represented using linear combinations of *basis functions*. These basis or potential functions usually have several degrees of freedom, including 3D position, size, and orientation. By combining these functions, one can create complex objects. Typical models might contain several hundred to several thousand such primitives. This is the strategy behind the *blobby models* proposed by Blinn [11].

While such an implicit modeling strategy offers a variety of new modeling tools, it has some limitations. In particular, the global nature of the potential functions limits one's ability to model *local* surface deformations. Consider a point $x \in \mathcal{S}$ where \mathcal{S} is the level surface associated with a model $\phi = \sum_i \alpha_i$, and α_i is one of the individual potential functions that comprise that model. Suppose one wishes to move the surface at the point x in a way that maintains continuity with the surrounding neighborhood. With multiple, global basis functions one must decide which basis function or combination of basis functions to alter and at the same time control the effects on other parts of the surface. The problem is generally ill posed—there are many ways to adjust the basis functions so that x will move in the desired direction, and yet it may be impossible to eliminate the effects of those movements on other disjoint parts of the surface. These prob-

lems can be overcome, but the solutions usually entail heuristics that tie the behavior of the surface deformation to, for example, the choice of representation [12].

An alternative to using a small number of *global* basis functions is to use a relatively large number of *local* basis functions. This is the principle behind using a volume as an implicit model. A volume is a discrete sampling of the embedding ϕ . It is also an implicit model with a very large number of basis functions, as shown in Fig. 6.1. The total number of basis functions is fixed; their positions (grid-points) and extent are also fixed. One can change only the magnitude of each basis function, i.e., each basis function has only one degree of freedom. A typical volume of size $128 \times 128 \times 128$ contains over a million such basis functions. The shape of each basis function is open to interpretation—it depends on how one interpolates the values between the grid-points. A trilinear interpolation, for instance, implies a basis function that is a piece-wise cubic polynomial with a value of one at the grid-point and zero at neighboring grid-points. Another advantage of using volumes as implicit models is that for the purposes of analysis we can treat the volume as a continuous function whose values can be *set* at each point according to the application. Once the continuous analysis is complete, we can map the algorithm into the discrete domain using standard methods of numerical analysis. The sections that

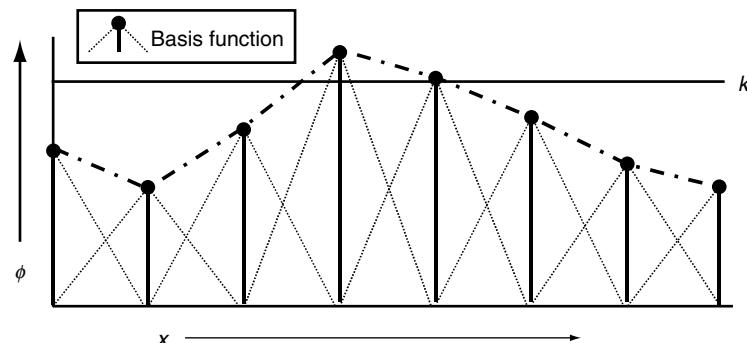


Figure 6.1 A volume can be considered as an implicit model with a large number of local basis functions.

follow discuss how to compute the geometry of surfaces that are represented as volumes and how to manipulate the shapes of those surfaces by changing the gray-scale values in the volume.

6.1.2.2 Isosurface Extraction and Visualization

This paper addresses the question of how to use volumes as surface models. Depending on the application, however, a 3D grid of data (i.e., a volume) may not be a suitable model representation. For instance, if the goal is to make measurements of an object or visualize its shape, an explicit model might be necessary. In such cases it is beneficial to convert between volumes and other modeling technologies.

For instance, the literature proposes several methods for scan-converting polygonal meshes or solid models [13]. Likewise, a variety of methods exists for extracting parametric models of isosurfaces from volumes. The most prevalent method is to locate isosurface crossings along grid lines in a volume (between voxels along the 3 cardinal directions) and then to link these points together to form triangles and meshes. This is the strategy of *marching cubes* [14] and other related approaches. However, extracting a parametric surface is not essential for visualization, and a variety of direct methods [15,16] are now computationally feasible and arguably superior in quality. This chapter does not address the issue of extracting or rendering isosurfaces, but rather studies the geometry of isosurfaces and how to manipulate them directly by changing the grey-scale values in the underlying volume. Thus, we propose volumes as a mechanism for studying and deforming surfaces, regardless of the ultimate form of the output. There are many ways of rendering or visualizing them, and these techniques are beyond the scope of this discussion.

6.2 Surface Normals

The surface normal of an isosurface is given by the normalized gradient vector. Typically, we

identify a surface normal with a point in the volume domain D . That is

$$\mathbf{n}(x) = \frac{\nabla\phi(x)}{|\nabla\phi(x)|} \text{ where } x \in D. \quad (6.3)$$

The convention regarding the direction of this vector is arbitrary; the negative of the normalized gradient magnitude is also normal to the isosurface. The gradient vector points toward that side of the isosurface that has greater values (i.e., is brighter). When rendering, the convention is to use *outward-pointing* normals, and the sign of the gradient must be adjusted accordingly. However, for most applications any consistent choice of normal vector will suffice. On a discrete grid, one must also decide how to approximate the gradient vector (i.e., first partial derivatives). In many cases central differences will suffice. However, in the presence of noise, especially when volume-rendering, it is sometimes helpful to compute first derivatives using some smoothing filter (e.g., convolution with a Gaussian). Alternatively, when calculating high-order geometry, one should use a polynomial or spline, with the appropriate degree of continuity [17]. When using the normal vector to solve certain kinds of partial differential equations, it is sometimes necessary to approximate the gradient vector with discrete, one-sided differences, as discussed in successive sections.

Q1

Note that a single volume contains families of nested isosurfaces, arranged like the layers of an onion. We specify the normal to an isosurface as a function of the position within the volume. That is, $\mathbf{n}(x)$ is the normal of the (single) isosurface that passes through the point x . The k value associated with that isosurface is $\phi(x)$.

6.3 Second-Order Structure

In differential geometric terms, the second-order structure of a surface is characterized by a quadratic patch that shares first-and second-order contact with the surface at a point (i.e., tangent plane and osculating circles). The *principal directions* of the surface are those associated with the quadratic approximation, and

the *principal curvatures* k_1, k_2 , are the curvatures in those directions.

As described in Kindlmann et al. [17], the second structure of the isosurface can be computed from the first-and second-order structure of the embedding, ϕ . All of the isosurface shape information is contained in a field of normals given by $\mathbf{n}(\mathbf{x})$. The 3×3 matrix of derivatives of this vector is

$$\mathbf{N} = -[n_x n_y n_z] \quad (6.4)$$

The projection of this derivative onto the tangent plane of the isosurface gives the shape matrix, β . Let P denote the normal projection operator, which is defined as

$$\mathbf{P} = \mathbf{n} \otimes \mathbf{n} = \frac{1}{\|\nabla\phi\|^2} \begin{pmatrix} \phi_x^2 & \phi_x\phi_y & \phi_x\phi_z \\ \phi_y\phi_x & \phi_y^2 & \phi_y\phi_z \\ \phi_z\phi_x & \phi_z\phi_y & \phi_z^2 \end{pmatrix} \quad (6.5)$$

The tangential projection operator is $I - P$, and thus the shape matrix is

$$\beta = NT = TH_\phi T \quad (6.6)$$

where H_ϕ is the Hessian of ϕ . The shape matrix β has 3 real eigenvalues, which are

$$e_1 = k_1, e_2 = k_2, e_3 = 0 \quad (6.7)$$

The corresponding eigenvectors are the principle directions of the surface (i.e., in the tangent plane) and the normal, respectively.

The *mean curvature* is the mean of the two principal curvatures, which is one-half of the trace of β , which is equal to the trace of N :

$$\begin{aligned} H &= \frac{k_1 + k_2}{2} = \frac{1}{2} \text{Tr}(\mathbf{N}) \\ &= \frac{\phi_x^2(\phi_{yy} + \phi_{zz}) + \phi_y^2(\phi_{xx} + \phi_{zz}) + \phi_z^2(\phi_{xx} + \phi_{yy}) - 2\phi_x\phi_y\phi_{xy} - 2\phi_x\phi_z\phi_{xz} - 2\phi_y\phi_z\phi_{yz}}{2(\phi_x^2 + \phi_y^2 + \phi_z^2)^{3/2}} \end{aligned} \quad (6.8)$$

The *Gaussian curvature* is the product of the principal curvatures:

$$\begin{aligned} K &= k_1 k_2 = e_1 e_2 + e_1 e_3 + e_2 e_3 = 2\text{Tr}(\mathbf{N})^2 - \frac{1}{2} \|\mathbf{N}\|^2 \\ &= \frac{\phi_x^2(\phi_{xx}\phi_{yy} - \phi_{xy}\phi_{xy}) + \phi_y^2(\phi_{xx}\phi_{zz} - \phi_{xz}\phi_{xz}) + \phi_z^2(\phi_{yy}\phi_{zz} - \phi_{yz}\phi_{yz}) + 2(\phi_x\phi_y(\phi_{xz}\phi_{yz} - \phi_{xy}\phi_{zz}) + \phi_x\phi_z(\phi_{xy}\phi_{xz} - \phi_{yz}\phi_{xx}))}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^2} \end{aligned} \quad (6.9)$$

The total curvature, also called the deviation from flatness D [18], is the root sum of squares of the two principal curvatures, which is the Euclidean norm of the matrix β .

Notice these measures exist at every point in U , and at each point they describe the geometry of the particular isosurface that passes through that point. All of these quantities can be computed on a discrete volume using finite differences, as described in successive sections.

6.4 Deformable Surfaces

This section begins with mathematics for describing geometric surface deformations on parametric models. The result is an evolution equation for a surface. Any term in this geometric evolution equation can be reexpressed in a way that is independent of the parameterization. Finally, the evolution equation for a parametric surface gives rise to an evolution equation (differential equation) on a volume, which encodes the shape of that surface as a level set.

6.4.1 Surface Deformation

A regular surface $S \subset \mathbb{R}^3$ is a collection of points in 3D that can be represented locally as a continuous function. In geometric modeling a surface is typically represented as a two-parameter object in a 3D space, i.e., a surface is local to a mapping S :

$$\begin{aligned} S: V \times V &\mapsto \mathbb{R}^3 \\ r \quad S &\quad x, y, z' \end{aligned} \tag{6.10}$$

where $V \times V \mathbb{R}^2$, and the bold notation refers specifically to a parameterized surface (vector-valued function). A deformable surface exhibits some motion over time. Thus $\mathbf{S} = \mathbf{S}(r, s, t)$, where $t \in \mathbb{R}^+$. We assume second-order-continuous, orientable surfaces; therefore, at every point on the surface (and in time) there is surface normal $\mathbf{N} = \mathbf{N}(r, s, t)$. We use \mathcal{S}_t to refer to the entire set of points on the surface.

Local deformations of \mathbf{S} can be described by an evolution equation, i.e., a differential equation on \mathbf{S} that incorporates the position of the surface, local and global shape properties, and responses to other forcing functions. That is,

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{G}(\mathbf{S}, \mathbf{S}_r, \mathbf{S}_s, \mathbf{S}_{rr}, \mathbf{S}_{rs}, \mathbf{S}_{ss}, \dots) \tag{6.11}$$

where the subscripts represent partial derivatives with respect to those parameters. The evolution of \mathbf{S} can be described by a sum of terms that depends on both the geometry of \mathbf{S} and the influence of other functions or data.

There are a variety of differential expressions that can be combined for different applications. For instance, the model could move in response to some directional *forcing* function [19,20], $\mathbf{F}: U \mapsto \mathbb{R}^3$, that is

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{F}(\mathbf{S}) \tag{6.12}$$

Alternatively, the surface could expand and contract with a spatially varying speed. For instance,

$$\frac{\partial \mathbf{S}}{\partial t} = G(\mathbf{S})\mathbf{N} \tag{6.13}$$

where $G: \mathbb{R}^3 \mapsto \mathbb{R}$ is a signed speed function. The evolution might also depend on the surface geometry itself. For instance,

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{S}_{rr} + \mathbf{S}_{ss} \tag{6.14}$$

describes a surface that moves in a way that becomes more *smooth* with respect to its own parameterization. This motion can be combined

with the motion of Equation 6.12 to produce a model that is pushed by a forcing function but maintains a certain smoothness in its shape and parameterization. There are myriad terms that depend on both the differential geometry of the surface and outside forces or functions to control the evolution of a surface.

6.5 Deformation: The Level-Set Approach

The method of level-sets, proposed by Osher and Sethian [21] and described extensively in Sethian [2], provides the mathematical and numerical mechanisms for computing surface deformations as time-varying iso-values of ϕ by solving a partial differential equation on the 3D grid. That is, the level-set formulation provides a set of numerical methods that describe how to manipulate the greyscale values in a volume, so that the isosurfaces of ϕ move in a prescribed manner (Fig. 6.2).

We denote the movement of a point on a surface as it deforms as dx/dt , and we assume that this motion can be expressed in terms of the position of $x \in U$ and the geometry of the surface at that point. In this case, there are generally two options for representing such surface movements implicitly:

Static: A single, static $\phi(x)$ contains a family of level-sets corresponding to surfaces as different times t . That is,

$$\phi(x(t)) = k(t) \Rightarrow \nabla \phi(x) \cdot \frac{\partial x}{\partial t} = \frac{dk(t)}{dt} \tag{6.15}$$

To solve this static method requires constructing a ϕ that satisfies Equation 6.15. This is a boundary-value problem, meaning it can be solved somewhat efficiently, starting with a single surface using the fast marching method of Sethian [22]. This representation has some significant limitations, however, because (by definition) a surface cannot pass back over itself over time, i.e., motions must be strictly monotonic—inward or outward.

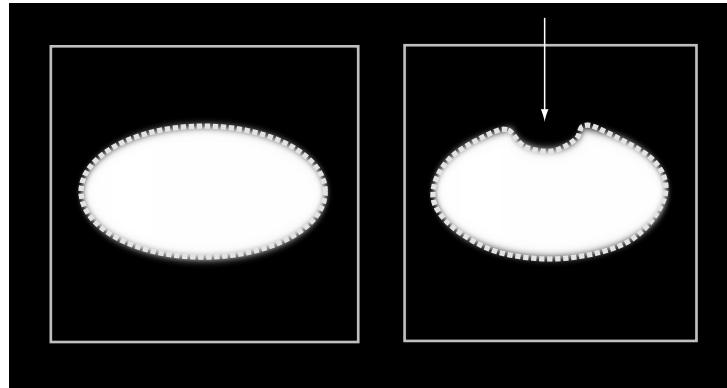


Figure 6.2 Level-set models represent curves and surfaces implicitly using greyscale images; a) an ellipse is represented as the level set of an image, b) to change the shape we modify the greyscale values of the image.

Dynamic: The approach is to use a one-parameter *family* of embeddings, in which $\phi(x, t)$ changes over time, x remains on the k level set of ϕ as it moves, and k remains constant. The behavior of ϕ is obtained by setting the total derivative of $\phi(x(t), t) = k$ to zero. Thus,

$$\phi(x(t), t) = k \Rightarrow \frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{dx}{dt} \quad (6.16)$$

This approach can accommodate models that move forward and backward and cross back over their own paths (over time). However, to solve this requires solving the initial value problem (using finite forward differences) on $\phi(x, t)$ —a potentially large computational burden. The remainder of this discussion focuses on the dynamic case, because of its superior flexibility.

All surface movements depend on position and geometry, and the level-set geometry is expressed in terms of the differential structure of ϕ . Therefore, the dynamic formulation from Equation 6.16 gives a general form of the partial differential equation on ϕ :

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= -\nabla \phi \cdot \frac{dx}{dt} \\ &= -\nabla \phi \cdot \mathbf{F}(x, D\phi, D^2\phi, \dots) \end{aligned} \quad (6.17)$$

where $D^n\phi$ is the set of order- n derivatives of ϕ evaluated at x . Because this relationship applies to every level-set of ϕ , i.e., all values of k , this equation can be applied to all of U , and therefore the movements of *all* the level-set surfaces embedded in ϕ can be calculated from Equation 6.17.

The level-set representation has a number of practical and theoretical advantages over conventional surface models, especially in the context of deformation and segmentation. First, level-set models are topologically flexible; they can easily represent complicated surface shapes that can, in turn, form holes, split to form multiple objects, or merge with other objects to form a single structure. These models can incorporate many (millions) of degrees of freedom, and therefore they can accommodate complex shapes. Indeed, the shapes formed by the level sets of ϕ are restricted only by the resolution of the sampling. Thus, there is no need to reparameterize the model as it undergoes significant deformations.

Such level-set methods are well documented in the literature [21,23] for applications such as computational physics [24], image processing [25,26], computer vision [27,5], medical image analysis [4,5], and 3D reconstruction [28,29].

For instance, in computational physics, level-set methods are a powerful tool for modeling moving boundaries between different materials (see Osher and Fedkiw [24] for a nice overview of recent results). Examples are water-air and water-oil interfaces. In such cases, level-set methods can be used to compute deformations that minimize surface area while preserving volumes for materials that *split and merge* in arbitrary ways. The method can be extended to multiple, nonoverlapping objects [30].

Level-set methods have also been shown to be effective in extracting surface structures from biological and medical data. For instance, Malladi et al. [5] propose a method in which the level-sets form an expanding or contracting contour that tends to *cling* to interesting features in 2D angiograms. At the same time the contour is also influenced by its own curvature, and therefore remains smooth. Whitaker et al. [4,31] have shown that level sets can be used to simulate conventional deformable surface models, and demonstrated this by extracting skin and tumors from *thick-sliced* (e.g., clinical) MR data, and by reconstructing a model of a fetus from 3D ultrasound. A variety of authors [32,33,26,34] have presented variations on the method and presented results for 2D and 3D data. Sethian [2] gives several examples of level-set curves and surfaces for segmenting CT and MR data.

6.5.1 Deformation Modes

In the case of parametric surfaces, one can choose from a variety of different expressions to construct an evolution equation that is appropriate for a particular application. For each of those parametric expressions, there is a corresponding expression that can be formulated on ϕ , the volume in which the level-set models are embedded. In constructing evolutions of levels-sets, there can be no reference to the underlying surface parameterization (terms depending on r and s in Equations 6.10 through

6.14). This has two important implications: i) only those surface movements that are normal to the surface are represented—any other movement is equivalent to a reparameterization; ii) all of the derivatives with respect to surface parameters r and s must be expressed in terms of invariant surface properties that can be derived without a parameterization.

Consider the term $S_{rr} + S_{ss}$ from Equation 6.14. If r, s is an orthonormal parameterization, the effect of that term is based purely on surface shape, not on the parameterization, and the expression $S_{rr} + S_{ss}$ is twice the *mean curvature*, H , of the surface. The corresponding level-set formulation is given by Equation 6.8.

Table 6.1 shows a list of expressions used in the evolution of parameterized surfaces and their equivalents for level-set representations. Also given are the assumptions about parameterization that give rise to the level-set expressions.

Q3

6.6 Numerical Methods

By taking the strategy of embedding surface models in volumes, we have converted equations that describe the movement of surface points to nonlinear, partial differential equations defined on a volume, which is generally a rectilinear grid. The expression $u_{i,j,k}^n$ refers to the n th time step at position i, j, k , which has an associated value in the 3D domain of the continuous volume $\phi(x_i, y_j, z_k)$. The goal is to solve the differential equation consisting of terms from Table 5.1 on the discrete grid $u_{i,j,k}^n$.

Q4

The discretization of these equations raises two important issues. First is the availability of accurate, stable numerical schemes for solving these equations. Second is the problem of computational complexity and the fact that we have converted a *surface* problem to a *volume* problem, increasing the dimensionality of the domain over which the evolution equations must be solved.

The level-set terms in Table 6.1 are combined, based on the needs of the application, to create a partial differential equation on $\phi(\mathbf{x}, t)$. The solutions to these equations are computed using finite differences. Along the time axis, solutions are obtained using finite *forward* differences, beginning with an initial model (i.e., volume) and stepping sequentially through a series of discrete time steps (which are denoted as superscripts on u). Thus, the update equation is

$$u_{i,j,k}^{n+1} = u_{i,j,k}^n + \Delta t \Delta u_{i,j,k}^n \quad (6.18)$$

The term $\Delta u_{i,j,k}^n$ is a discrete approximation to $\partial\phi/\partial t$, which consists of a weighted sum of terms such as those in Table 5.1. Those terms must, in turn, be approximated using finite differences on the volume grid.

Q5

6.6.1 Upwind Schemes

The terms in Table 6.1 fall into two basic categories: the first-order terms (items 1 and 2) and the second-order terms (items 3 through 5). The first-order terms describe a moving wave front with a space-varying velocity (expression 1) or speed (expression 2). Equations of this form cannot be solved with a simple finite forward difference scheme. Such schemes tend to overshoot, and they are unstable. To address this issue, Osher and Sethian [1] proposed an *upwind* scheme. The upwind method relies on a one-sided derivative that looks in the upwind direc-

tion of the moving wave front, and thereby avoids the overshooting associated with finite forward 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, differences in the x direction on a discrete grid $u_{i,j,k}$ with domain X and uniform spacing h are defined as

$$\delta_x^{(+)} u_{i,j,k} \triangleq (u_{i+1,j,k} - u_{i,j,k})/h, \quad (6.19)$$

$$\delta_x^{(-)} u_{i,j,k} \triangleq (u_{i,j,k} - u_{i-1,j,k})/h, \text{ and} \quad (6.20)$$

$$\delta_x u_{i,j,k} \triangleq (u_{i+1,j,k} - u_{i-1,j,k})/(2h), \quad (6.21)$$

(6.22)

where we have left off the time superscript for conciseness. Second-order terms are computed using the *tightest-fitting* central difference operators. For example,

$$\delta_{xx} u_{i,j,k} \triangleq (u_{i+1,j,k} + u_{i-1,j,k} - 2u_{i,j,k})/h^2, \quad (6.23)$$

$$\delta_{zz} u_{i,j,k} \triangleq (u_{i,j,k+1} + u_{i,j,k-1} - 2u_{i,j,k})/h^2, \text{ and} \quad (6.24)$$

$$\delta_{xy} u_{i,j,k} \triangleq \delta_x \delta_y u_{i,j,k} \quad (6.25)$$

The discrete approximation to the first-order terms in Table 5.1 are computed using the upwind scheme proposed by Osher and Sethian [21]. This strategy avoids overshooting by approximating the gradient of ϕ using a one-sided difference in the direction that

Table 6.1 A list of evolution terms for parametric models has a corresponding expression on the embedding, ϕ , associated with the level-set models.

	<i>Effect</i>	<i>Parametric Evolution</i>	<i>Level-Set Evolution</i>	<i>Parameter Assumptions</i>
1	External force	\mathbf{F}	$\mathbf{F} \cdot \nabla \phi$	None
2	Expansion/contraction	$G(\mathbf{x})\mathbf{N}$	$G(\mathbf{x}) \nabla \phi(\mathbf{x}, t) $	None
3	Mean curvature	$S_{rr} + S_{ss}$	$H \nabla \phi $	Orthonormal
4	Gauss curvature	$S_{rr} \times S_{ss}$	$K \nabla \phi $	Orthonormal
5	Second order	S_{rr} or S_{ss}	$(H \pm \sqrt{H^2 - K}) \nabla \phi $	Principal curvatures

Q6

Q7

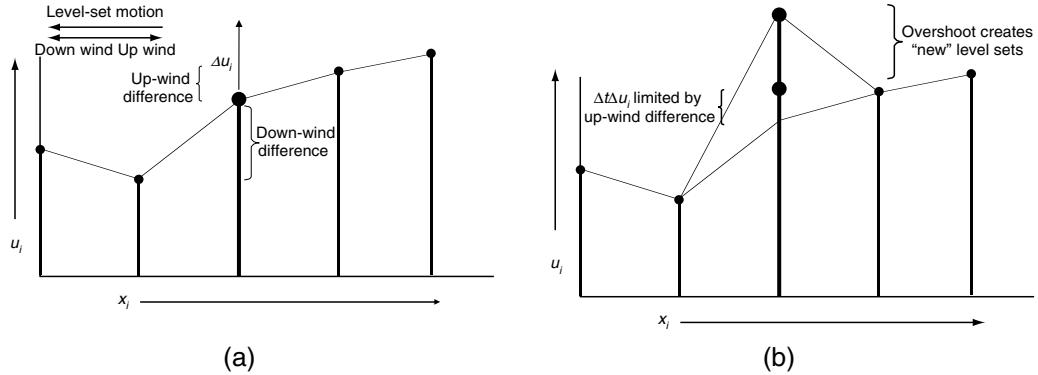


Figure 6.3 The upwind numerical scheme uses one-sided derivatives to prevent overshooting and the creation of new level-sets.

is upwind of the moving level-set, thereby ensuring that no *new* contours are created in the process of updating $u_{i,j,k}^n$ (Fig. 6.3). The scheme is separable along each axis (x , y , and z).

Consider Term 1 in Table 5.1. If we use superscripts to denote the vector components, i.e.,

$$\mathbf{F}(x, y, z) = (F^{(x)}(x, y, z), F^{(y)}(x, y, z), F^{(z)}(x, y, z)), \quad (6.26)$$

the upwind calculation for a grid-point $u_{i,j,k}^n$ is

$$\begin{aligned} \mathbf{F}(x_i, y_i, z_i) \cdot \nabla \phi(x_i, y_j, z_k, t) &\approx \\ \sum_{q \in \{x, y, z\}} F^{(q)}(x_i, y_i, z_i) & \\ \begin{cases} \delta_q^+ u_{i,j,k}^n & F^{(q)}(x_i, y_i, z_i) > 0 \\ \delta_q^- u_{i,j,k}^n & F^{(q)}(x_i, y_i, z_i) < 0 \end{cases} \end{aligned} \quad (6.27)$$

The time steps are limited—the fastest-moving wave front can move only one grid unit per iteration. That is,

$$\Delta t_F \leq \frac{1}{\sum_{q \in \{x, y, z\}} \sup_{i,j,k \in X} \{|\nabla F^{(q)}(x_i, y_j, z_k)|\}} \quad (6.28)$$

Q8

For Term 2 in Table 5.1 the direction of the moving surface depends on the normal, and therefore the same upwind strategy is applied in a slightly different form.

$$\begin{aligned} G(x_i, y_j, z_k) |\nabla \phi(x_i, y_j, z_k, t)| &\approx \\ \sum_{q \in \{x, y, z\}} G(x_i, y_i, z_i) & \\ \begin{cases} \max^2(\delta_q^+ u_{i,j,k}^n, 0) + \min^2(\delta_q^- u_{i,j,k}^n, 0) & \\ G(x_i, y_i, z_i) > 0 \\ \min^2(\delta_q^+ u_{i,j,k}^n, 0) + \max^2(\delta_q^- u_{i,j,k}^n, 0) & \\ G(q)(x_i, y_i, z_i) < 0 \end{cases} \end{aligned} \quad (6.29)$$

The time steps are, again, limited by the fastest-moving wave front:

$$\Delta t_G \leq \frac{1}{3 \sup_{i,j,k \in X} \{|\nabla G(x_i, y_j, z_k)|\}} \quad (6.30)$$

To compute the approximation of the update to the second-order terms in Table 5.1 requires only central differences. Thus, the mean curvature is approximated as

$$\begin{aligned} H_{i,j,k}^n &= \\ \frac{1}{2} \left((\delta_x u_{i,j,k}^n)^2 + (\delta_y u_{i,j,k}^n)^2 + (\delta_z u_{i,j,k}^n)^2 \right)^{-1} \times & \\ \left[\left((\delta_y u_{i,j,k}^n)^2 + (\delta_z u_{i,j,k}^n)^2 \right) \delta_{xx} u_{i,j,k}^n \right. & \\ + \left((\delta_z u_{i,j,k}^n)^2 + (\delta_x u_{i,j,k}^n)^2 \right) \delta_{yy} u_{i,j,k}^n & \\ + \left((\delta_x u_{i,j,k}^n)^2 + (\delta_y u_{i,j,k}^n)^2 \right) \delta_{zz} u_{i,j,k}^n & \\ - 2\delta_x u_{i,j,k}^n \delta_y u_{i,j,k}^n \delta_{xy} u_{i,j,k}^n - 2\delta_y u_{i,j,k}^n \delta_z u_{i,j,k}^n \delta_{yz} u_{i,j,k}^n & \\ - 2\delta_z u_{i,j,k}^n \delta_x u_{i,j,k}^n \delta_{zx} u_{i,j,k}^n \right] & \end{aligned} \quad (6.31)$$

Q9

Such curvature terms can be computed by using a combination of forward and backward differences as described by Whitaker and Xue [35]. In some cases this is advantageous, but the details are beyond the scope of this paper.

For the second-order terms, the time steps are limited, for stability, by the diffusion number, to

$$\Delta t_H \leq \frac{1}{6} \quad (6.32)$$

When combining terms, the maximum number of time steps for each term is scaled by one over the weighting coefficient for that term.

6.6.2 Narrow-Band Methods

If one is interested in only a *single-level set*, the formulation described previously is not efficient. This is because solutions are usually computed over the entire domain of ϕ . The solutions, $\phi(x, y, z, t)$ describe the evolution of an embedded family of contours. While this dense family of solutions might be advantageous for certain applications, there are other applications that require only a single surface model. In such applications the calculation of solutions over a dense field is an unnecessary computational burden, and the presence of contour families can be a nuisance because further processing might be required to extract the level-set that is of interest.

Fortunately, the evolution of a single level-set, $\phi(x, t) = k$, is not affected by the choice of embedding. The evolution of the level-sets is such that they evolve independently (to within the error introduced by the discrete grid). Furthermore, the evolution of ϕ is important only in the vicinity of that level-set. Thus, one should perform calculations for the evolution of ϕ only in a neighborhood of the surface $S = \{\mathbf{x} | \phi(\mathbf{x}) = k\}$. In the discrete setting, there is a particular subset of grid-points whose values control a particular level set (Fig. 6.4). Of course, as the surface moves, that subset of grid points must change to account for its new position.

Adalsteinson and Sethian [36] propose a *narrow-band* approach, which follows this line of reasoning. The narrow-band technique con-

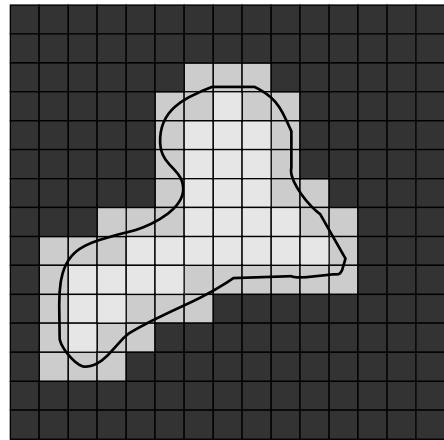


Figure 6.4 A level curve of a 2D scalar field passes through a finite set of cells. Only those grid-points nearest to the level curve are relevant to the evolution of that curve.

structs an embedding of the evolving curve or surface via a signed distance transform. The distance transform is truncated, i.e., computed over a finite width of only m points that lie within a specified distance to the level set. The remaining points are set to constant values to indicate that they do not lie within the narrow band, or *tube*, as they call it. The evolution of the surface (they demonstrate it for curves in the plane) is computed by calculating the evolution of u only on the set of grid-points that are within a fixed distance to the initial level-set, i.e., within the narrow band. When the evolving level-set approaches the edge of the band (Fig. 6.5), they calculate a new distance transform (e.g., by solving the Eikonal equation with the fast marching method), which creates a new embedding, and they repeat the process. This algorithm relies on the fact that the embedding is not a critical aspect of the evolution of the level-set. That is, the embedding can be transformed or recomputed at any point in time, so long as such a transformation does not change the position of the k th level set, and the evolution will be unaffected by this change in the embedding.

Despite the improvements in computation time, the narrow-band approach is not optimal

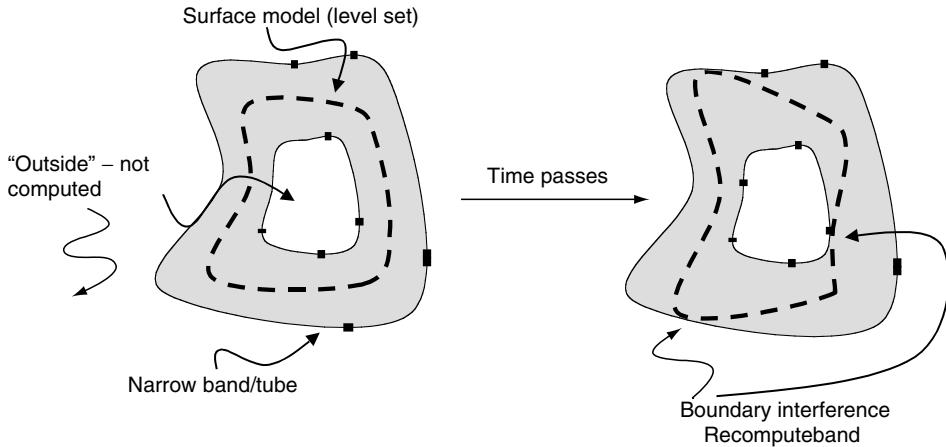


Figure 6.5 The narrow-band scheme limits computation to the vicinity of the specific level-set. As the level-set moves near the edge of the band, the process is stopped and the band recomputed.

for several reasons. First it requires a band of significant width ($m = 12$ in the examples of Adalsteinson and Sethian [36]) where one would like to have a band that is only as wide as necessary to calculate the derivatives of u near the level-set (e.g. $m = 2$). The wider band is necessary because the narrow-band algorithm trades off two competing computational costs. One is the cost of stopping the evolution and computing the position of the curve and distance transform (to sub-cell accuracy) and determining the domain of the band. The other is the cost of computing the evolution process over the entire band. The narrow-band method also requires additional techniques, such as smoothing, to maintain the stability at the boundaries of the band, where some grid points are undergoing the evolution and nearby neighbors are static.

6.6.3 The Sparse-Field Method

The basic premise of the narrow-band algorithm is that computing the distance transform is so costly that it cannot be done at every iteration of the evolution process. Another strategy is to use an approximation to the distance transform that makes it feasible to recom-

pute the neighborhood of the level-set model at each time step. Computation of the evolution equation is done on a band of grid-points that is only one point wide. The embedding is extended from the active points to a neighborhood around those points that is precisely the width needed at each time. This extension is done via a fast distance transform approximation.

This approach has several advantages. First, the algorithm does precisely the number of calculations needed to compute the next position of the level curve. It does not require explicit recalculation of the positions of level sets and their distance transforms. Because the number of points being computed is so small, it is feasible to use a linked list to keep track of them. Thus, at each iteration the algorithm visits only those points adjacent to the k -level curve. For large 3D data sets, the very process of incrementing a counter and checking the status of all of the grid-points is prohibitive.

The *sparse-field* algorithm is analogous to a locomotive engine that lays tracks before it and picks them up from behind. In this way the number of computations increases with the surface area of the model rather than the resolution of the embedding. Also, the sparse-field approach identifies a single level-set with a specific

set of points whose values control the position of that level-set. This allows one to compute external forces to an accuracy that is better than the grid spacing of the model, resulting in a modeling system that is more accurate for various kinds of *model-fitting* applications.

The sparse-field algorithm takes advantage of the fact that a k -level surface, S , of a discrete image u (of any dimension) has a set of cells through which it passes, as shown in Fig. 6.4. The set of grid-points adjacent to the level set is called the *active set*, and the individual elements of this set are called *active points*. As a first-order approximation, the distance of the level set from the center of any active point is proportional to the value of u divided by the gradient magnitude at that point. Because all of the derivatives (up to second order) in this approach are computed using nearest-neighbor differences, only the active points and their neighbors are relevant to the evolution of the level-set at any particular time in the evolution process. The strategy is to compute the evolution given by Equation 6.17 on the active set and then update the neighborhood around the active set using a fast distance transform. Because active points must be adjacent to the level-set model, their positions lie within a fixed distance to the model. Therefore, the values of u for locations in the active set must lie within a certain range. When active-point values move out of this *active range*, they are no longer adjacent to the model. They must be removed from the set, and other grid-points, those whose values are moving into the active range, must be added to take their place. The precise ordering and execution of these operations is important to the operation of the algorithm.

The values of the points in the active set can be updated using the upwind scheme for first-order terms and central differences for the mean-curvature flow, as described in the previous sections. In order to maintain stability, one must update the neighborhoods of active grid-points in a way that allows grid-points to enter and leave the active set without those changes

in status affecting their values. Grid-points should be removed from the active set when they are no longer the nearest grid-point to the zero crossing. If we assume that the embedding u is a discrete approximation to the distance transform of the model, then the distance of a particular grid-point $x_m = (i, j, k)$, to the level-set is given by the value of u at that grid-point. If the distance between grid-points is defined to be unity, then we should remove a point from the active set when the value of u at that point no longer lies in the interval $[-\frac{1}{2}, \frac{1}{2}]$ (Fig. 6.6). If the neighbors of that point maintain their distance of 1, then those neighbors will move into the active range just as x_m is ready to be removed.

There are two operations that are significant to the evolution of the active set. First, the values of u at active points change from one iteration to the next. Second, as the values of active points pass out of the active range, they are removed from the active set and other, neighboring grid-points are added to the active set to take their place. Whitaker [29] gives some formal definitions of active sets and the operations that affect them, definition which show that active sets will always form a boundary between positive and negative regions in the image, even as control of the level-set passes from one set of active points to another.

Because grid-points that are near the active set are kept at a fixed value difference from the active points, active points serve to control the behavior of nonactive grid-points to which they are adjacent. The neighborhoods of the active set are defined in *layers*, $L_{+1}, \dots L_{+N}$ and $L_{-1}, \dots L_{-N}$, where the i indicates the distance (city-block distance) from the nearest active grid-point, and negative numbers are used for the outside layers. For notational convenience, the active set is denoted L_0 .

The number of layers should coincide with the size of the footprint or neighborhood used to calculate derivatives. In this way, the inside and outside grid-points undergo no changes in their values that affect or distort the evolution of the zero set. Most of the level-set work relies

on surface normals and curvature, which require only second-order derivatives of ϕ . Second-order derivatives are calculated using a $3 \times 3 \times 3$ kernel (city-block distance 2 to the corners). Therefore, only five layers are necessary (2 inside layers, 2 outside layers,

and the active set). These layers are denoted L_1, L_2, L_{-1}, L_{-2} , and L_0 .

The active set has grid-point values in the range $[-\frac{1}{2}, \frac{1}{2}]$. The values of the grid-points in each neighborhood layer are kept 1 unit from the layer next closest to the active set (Fig. 6.6).

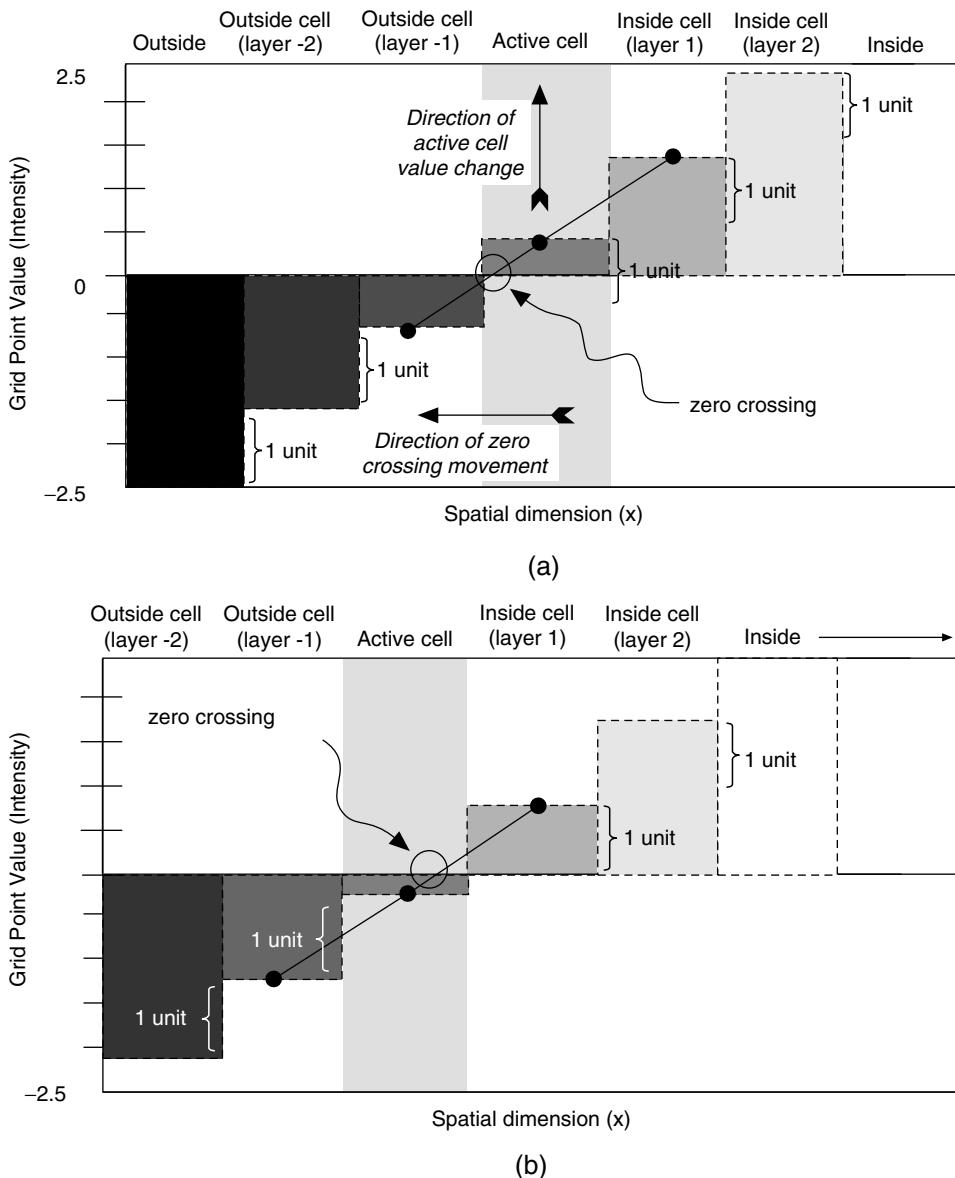


Figure 6.6 The status of grid-points and their values at two different points in time show that as the zero crossing moves, activity is passed from one grid-point to another.

Thus the values of layer L_i fall in the interval $[i - \frac{1}{2}, i + \frac{1}{2}]$. For $2N + 1$ layers, the values of the grid-points that are totally inside and outside are $N + \frac{1}{2}$ and $-N - \frac{1}{2}$, respectively. The procedure for updating the image and the active set based on surface movements is as follows:

1. For each active grid-point, $x_m = (i, j, k)$, do the following:
 - (a) Calculate the local geometry of the level-set.
 - (b) Compute the net change of u_{xm} , based on the internal and external forces, using some stable (e.g., upwind) numerical scheme where necessary.
2. For each active grid-point x_j add the change to the grid-point value and decide if the new value u_{xm}^{n+1} falls outside the $[-\frac{1}{2}, \frac{1}{2}]$ interval. If so, put x_m on lists of grid-points that are changing status, called the *status list*; S_1 or S_{-1} , for $u_{xm}^{n+1} > 1$ or $u_{xm}^{n+1} < -1$, respectively.
3. Visit the grid-points in the layers L_i in the order $i = \pm 1, \dots, \pm N$, and update the grid-point values based on the values (by adding or subtracting one unit) of the next inner layer, $L_{i\mp 1}$. If more than one $L_{i\mp 1}$ neighbor exists, then use the neighbor that indicates a level curve closest to that grid-point, i.e., use the maximum for the outside layers and minimum for the inside layers. If a grid-point in layer L_i has no $L_{i\mp 1}$ neighbors, then it gets demoted to $L_{i\pm 1}$, the next level away from the active set.
4. For each status list $S_{\pm 1}, S_{\pm 2}, \dots, S_{\pm N}$, do the following:
 - (a) For each element x_j on the status list S_i , remove x_j from the list $L_{i\mp 1}$ and add it to the list L_i , or, in the case of $i = \pm(N + 1)$, remove it from all lists.
 - (b) Add all $L_{i\mp 1}$ neighbors to the $S_{i\pm 1}$ list.

This algorithm can be implemented efficiently using linked-list data structures combined with arrays to store the values of the grid-points and their states, as shown in Fig. 6.7. This requires

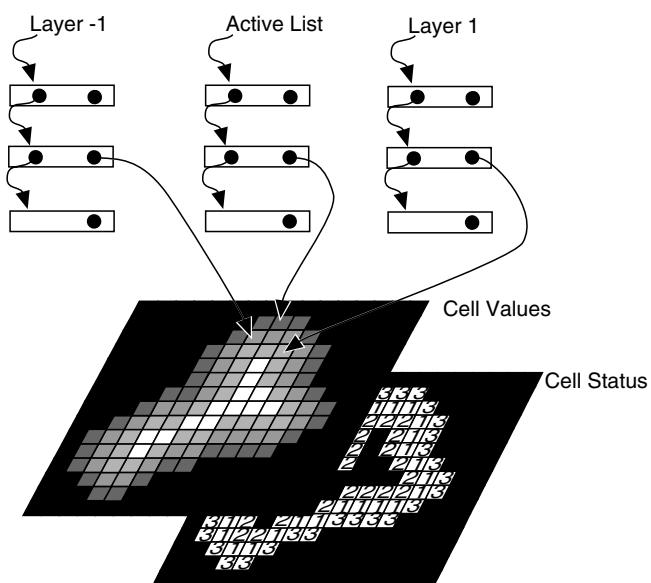


Figure 6.7 Linked-list data structures provide efficient access to those grid-points with values and status that must be updated.

only those grid-points whose values are changing, the active points and their neighbors, to be visited at each time step. The computation time grows as m^{n-1} , where m is the number of grid-points along one dimension of u (sometimes called the resolution of the discrete sampling). Computation time for dense-field approach increases as m^n . The m^{n-1} growth in computation time for the sparse-field models is consistent with conventional (parameterized) models, for which computation times increase with the resolution of the domain, rather than the range.

Another important aspect of the performance of the sparse-field algorithm is the larger time steps that are possible. The time steps are limited by the speed of the *fastest* moving level curve, i.e., the maximum of the force function. Because the sparse-field method calculates the movement of level-sets over a subset of the image, time steps are bounded from below by those of the dense-field case, i.e.,

$$\sup_{x \in \mathcal{A} \subset X} (g(x)) \leq \sup_{x \in X} (g(x)) \quad (6.33)$$

where $g(x)$ is the space-varying speed function and \mathcal{A} is the active set.

Results from previous work by Whitaker [29] have demonstrated several important aspects of the sparse-field algorithm. First, the manipulations of the active set and surrounding layers allow the active set to *track* the deformable surface as it moves. The active set always divides the inside and outside of the objects it describes (i.e., it stays closed). Empirical results show significant increases in performance relative to both the computation of full domain and the narrow-band method, as proposed in the literature. Empirical results also show that the sparse-field method is about as accurate as both the full, discrete solution and the narrow-band method. Finally, because the method positions level-sets to sub-voxel accuracy, it avoids aliasing problems and is more accurate than these other methods when it comes to *fitting* level-set models to other surfaces. This sub-voxel accuracy is an important aspect of the

implementation and will significantly impact the quality of the results for the applications that follow.

6.7 Applications

This section describes several examples of how level-set surface models can be used to address problems in graphics, visualization, and computer vision. These examples are a small selection of those available in the literature. All of these examples were implemented using the sparse-field algorithm and the VISPack library.

6.7.1 Surface Morphing

This section summarizes the work of Breen and Whitaker [8], which describes the use of level-set surface models to perform 3D shape metamorphosis. The *morphing* of 3D surfaces is the process of constructing a series of 3D models that constitute a smooth transition from one shape to another (i.e., a homotopy). Such a capability is interesting for creating animations and as a tool for geometric modeling [37,38,39,40,41,42].

Level-set models provide an algorithm for 3D morphing, which is a natural extension of the mathematical principles discussed in previous sections. The strategy is to allow a free-form deformation of one surface (called the *initial* surface) using the signed distance transform of a second surface (the *target* surface). This free-form deformation is combined with an underlying coordinate transformation that gives either a rough global alignment of the two surfaces, or one-to-one relationships between a finite set of landmarks on both the initial and the target surfaces. The coordinate transformation can be computed automatically or using user input.

The distance transform gives the nearest Euclidean distance to a set of points, curve, or surface. For closed surfaces in 3D, the signed distance transform gives a positive distance for points inside and negative for points outside

(one can also choose the opposite sign convention). If two connected shapes overlap, then the initial surface can expand or contract using the distance transform of the target. The steady state of such a deformation process is a shape consisting of the zero set of the distance transform of the target. That is, the initial object becomes the target. This is the basis of the proposed 3D morphing algorithm.

Let $D(x)$ be the signed distance transform of the target surface, B , and let A be the initial surface. The evolution process, which takes a model S from A to B , is defined by

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{N}D(\mathbf{x}) \quad (6.34)$$

where $\mathbf{x}(t) \in S_t$ and $S_{t=0} = A$. The free-form deformations can be combined with an underlying coordinate transformation. The strategy is to use a coordinate transformation (e.g., a translation and rotation) to position the two surfaces near each other. These transformations can capture gross similarities in shape as well as user input. A coordinate transformation is given by

$$\mathbf{x}' = T(\mathbf{x}, \alpha) \quad (6.35)$$

where $0 \leq \alpha \leq 1$ parameterizes a continuous family of these transformations that begins with identity, i.e., $\mathbf{x} = T(\mathbf{x}, 0)$. The evolution equation for a parametric surface is

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{N}D(T(\mathbf{x}, 1)) \quad (6.36)$$

and the corresponding level-set equation is

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} = |\nabla \phi(\mathbf{x}, t)|D(T(\mathbf{x}, 1)) \quad (6.37)$$

This process produces a series of transition shapes (parameterized by t). The coordinate transformation can be a global rotation, translation, or scaling, or it might be a *warping of the underlying 3D space* [40]. Incorporating user input is important for any surface morphing technique, because in many cases finding the best set of transition surfaces depends on context. Only users can apply semantic considerations to the transformation of one object to another. However, this underlying coordinate

transformation can, in general, achieve only some finite similarity between the *warped* initial model and the target, and even this may require a great deal of user input. In the event that a user is not able or willing to define every important correspondence between two objects, some other method must *fill in* the gaps remaining between the initial and target surfaces. Lerios et al [40] propose alpha blending to achieve that smooth transition—really just a fading from one surface to the other. We are proposing the use of the free-form deformations, implemented with level-set models, to achieve a continuous transition between the shapes that result from the underlying coordinate transformation. We have also experimented with ways of automatically orienting and scaling objects, using 3D moments, in order to achieve a significant correspondence between two objects.

Fig. 6.8 shows a 3D model of a jet that was built using Clockworks [43], a CSG modeling system. Lerios et al. [40] demonstrate the transition of a jet to a dart, which was accomplished using 37 user-defined correspondences, roughly 100 user-defined parameters. Fig. 6.9 shows the use of level-set models to construct a set of transition surfaces between a jet and a dart. The triangle mesh is extracted from the volume using the method of marching cubes [14].

The application in this section shows how level-set models moving according to the first-order term given in expression 2 in Table 6.1 can fit other objects by moving with a speed that depends on the signed distance transform of the target object. The application in the next section relies on expression 5 of Table 6.1, a second-order flow that depends on the principal curvatures of the surface itself.

6.7.2 Surface Editing

This section gives a brief summary of the results in Museth et al. [44], who describe a system for surface editing based on level sets. The creation of complex models for such applications as movie special effects, graphic arts, and computer-aided design can be a time-consuming,

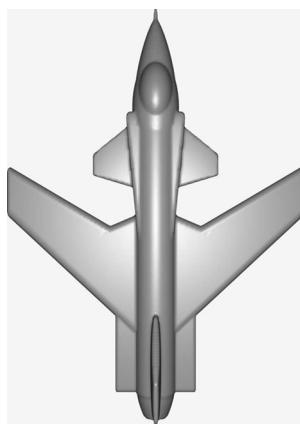


Figure 6.8 A 3D model of a jet that was built using Clockworks, a CSG modeling system.

Q11

tedious, and error-prone process. One of the solutions to the model creation problem is 3D photography [?], i.e., scanning a 3D object directly into a digital representation. However, the scanned model is rarely in a final desired form. The scanning process is imperfect and introduces errors and artifacts, or the object itself may be flawed. In order to overcome these difficulties, one can use a level-set approach to implementing operators for locally and globally editing closed surfaces.

In [44] they describe a number of surface-editing operators within the level-set framework by defining a collection of different level-set speed functions. The cut-and-paste operator gives the user the ability to copy, remove, and merge level-set models (using volumetric CSG operations) and automatically blends the intersection regions. The smoothing operator allows a user to define a region of interest and smooths the enclosed surface to a user-defined curvature value. They also describe a point-attraction operator, in which a regionally constrained portion of a level-set surface is attracted to a single point. By defining line segments, curves, polygons, patches, and 3D objects as densely sampled point sets, the single-point attraction operator may be combined to produce a more general surface-embossing operator. Morpho-

logical operators, such as opening and closing, can also be implemented in a level-set framework [45]. Such operations are useful for performing global blending (closing) and smoothing (opening) on level-set models. Because all of the operators accept and produce the same volumetric representation of closed surfaces, the operators may be applied repeatedly to produce a series of surface-editing operations, as shown in Fig. 6.10

6.7.3 Antialiasing Binary Volumes

This section presents a summary of results from Whitaker [46], whose article addresses the question of using level sets to reduce aliasing artifacts in binary volumes. Binary volumes are interesting for several practical reasons. First, in some cases, such as medical imaging, a volume dataset can be segmented to produce a set of voxels that correspond to some particular object (or anatomy). This segmentation can be manual, in which case a user identifies (usually with the aid of a GUI) all of the voxels that belong to a certain object. The segmentation can also be automated, relying on methods such as pattern classification, flood fill, and morphological watersheds, which produce segmentations that are *hard*, i.e., binary. Binary volumes are also important when using a 3D imaging device that produces data with very high contrast. In such cases the measured data is *essentially* binary with regard to both the information it contains and the problems it presents in rendering. Binary volumes can be important when visualizing mathematical expressions, such as fractals, that cannot be evaluated as continuous functions. Binary volumes are also interesting because they require so little data, and, with the use of run length encoding, are very well suited to compression.

The strategy presented here is related to the work of Gibson [47], who uses a deformable-surface approach to reducing aliasing artifacts. Gibson dealt with problem of extracting surfaces from binary volumes with an insightful strategy: *treat the binary data as a constraint on a surface that is subject to a regularization process*. She proposes a several-step algorithm

108 The Visualization Handbook

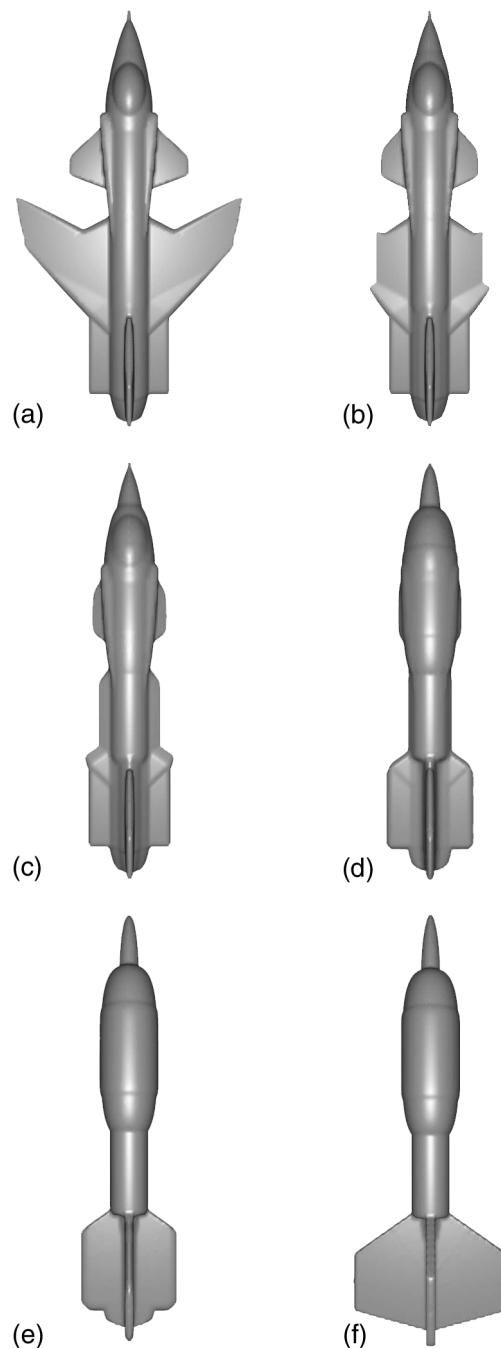


Figure 6.9 The deformation of the jet to a dart using a level-set model moving with a speed defined by the signed distance transform of the target object.

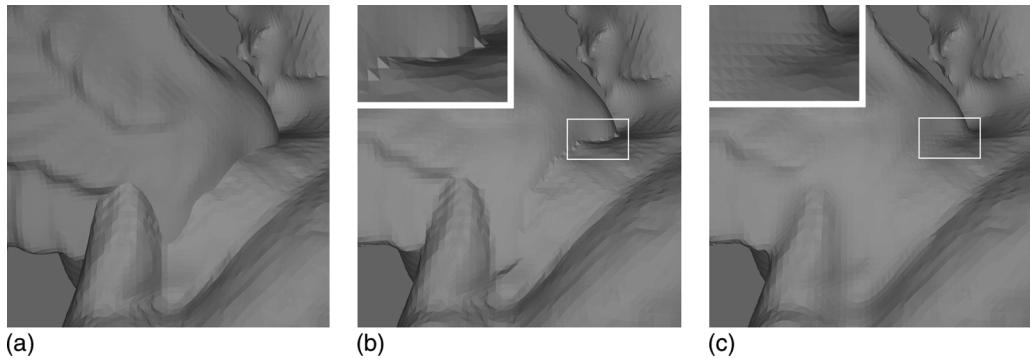


Figure 6.10 a) Positioning the (red) wing model on the dragon model. b) The models are pasted together (CSG union operation), producing sharp, undesirable creases, a portion of which is expanded in the box. c) Same region after automatic blending based on mean curvature. The blending is constrained to only move outwards. The models are rendered with flat-shading to highlight the details of the surface structure.

called *constrained surface nets* that embodies this strategy. The algorithm begins by extracting a surface mesh from the volume. This initial mesh consists of a vertex in each cell (an eight-voxel neighborhood arranged in a cube) whose corners indicate a transition from inside to out. This mesh undergoes an iterative process of deformation, where each vertex moves to the mean of its neighbors but is prohibited from moving outside of its original cell. The resulting surface can be converted back into a volume by computing a discrete sampling of the distance transform to the set of triangles associated with the final mesh.

Constrained surface nets have some very useful properties. They are essentially the solution of a constrained minimization of surface area. Constrained surface nets are capable of creating flat surfaces through sequences of distant *terraces* or *jaggies*, which are not easily spanned by a filter or interpolating function. The final solution is guaranteed to lie within a fixed distance of the original mesh, thus preserving small details, and even discontinuities, that can be lost through other anti-aliasing techniques such as those that rely on filtering [48,49] or surface approximation [50,51,52].

Using level sets, one can implement the basic philosophy of constrained surface nets while eliminating the need for an intermediate surface mesh—i.e., operate directly on the volume. The result is a transformation of a binary volume to a grey-scale volume, and thus it is a kind of nonlinear filtering process. The zero set of the volume that results from the algorithm has the desirable properties of the constrained surface net. However, the algorithm makes no explicit assumptions about the topology of the surface, but instead allows the topology of the surface to develop from the constrained minimization process.

Binary volumes are often visualized through treatment as implicit functions and rendering of the surfaces that correspond to the zero sets of interpolated version of $B : \mathcal{D} \mapsto \{-1, 1\}$, the binary volume. Alternatively, one could treat a binary volume, regardless of its origins, as a threshold (or binarization) of a discrete sampling of an embedding. That is,

$$\phi(x, y, z) \xrightarrow{\text{discretization}} f_{i,j,k} \xrightarrow{\text{threshold}} B_{i,j,k} \quad (6.38)$$

From this point of view, the problem of *extracting* surfaces from a binary volume is really the problem of *estimating* either f or ϕ , and extracting surfaces from one of those func-

tions. However, the loss of information (i.e., projection) associated with the binary sampling leaves the inverse problem ill posed—that is, for a given binary volume, there are infinitely many embeddings from which it could have been derived.

The strategy in this section is to construct a discrete sampling of a ϕ that *could* have given rise to B . An estimate of the embedding, $\hat{\phi}$, is *feasible* if

$$\hat{\phi}(x)B_x \geq 0 \forall x \in \mathcal{D} \quad (6.39)$$

i.e., B_x and $\hat{\phi}(x)$ must have the same sign at the grid-points of \mathcal{D} . This is the same as saying that the zero set of $\hat{\phi}$ must enclose all of those points indicated by the binary volume as *inside* and none of the points that are *outside*.

The ill-posed nature of the problem is addressed by imposing some criterion, a regularization, to which $\hat{\phi}$ must conform. In the case of surface estimation, a natural criterion is to choose the surface with minimal area. Often, but not always, surfaces with less area are qualitatively *smoother*. In the level-set formulation, the combined surface area of all of the level-sets of ϕ is the integral of the level-set density (which is the gradient magnitude of ϕ) over the domain D . Thus, using level-sets, the constrained minimization problem for reconstructing surfaces from binary data is

$$\hat{\phi} = \arg \min_{\phi} \left[\int_D |\nabla \phi(x)| dx \right] \quad (6.40)$$

such that $\phi(x)B_x \geq 0 \forall x \in \mathcal{D}$

Using the method of undetermined multipliers, construct the Lagrangian:

$$F(\phi, \lambda) = \int_D |\nabla \phi(x)| dx + \sum_{x_i \in \mathcal{D}} \lambda_i \phi(x_i) B_{x_i} \quad (6.41)$$

where $\lambda_i \leq 0$.

The Kuhn-Tucker [53] conditions describe the behavior of the solution. For points in the domain that are not on the grid, the level-sets of the solution are flat or hyperbolic (saddle points) with the principle curvatures offsetting

one another. For points in the domain that fall on one of the grid-points, there are two cases: the level-sets of ϕ are convex at places with $B(x_i) > 0$ and concave where $B(x_i) < 0$, which is consistent with a solution that is stretched around the positive and negative constraints. Also, when the curvature is non zero at a grid-point,

$$B(x_i)\phi(x_i) = \pm \phi(x_i) = 0 \quad (6.42)$$

which means that the zero set falls through grid-points of D except in those areas where the solution is flat.

This analysis leads to a gradient-descent strategy with an evolution parameter t . Starting with an initial estimate that is feasible, one can update ϕ in such a way that it minimizes the surface area but does not violate the constraints:

$$\frac{\partial \phi}{\partial t} = \begin{cases} 0 & \text{For } x = x_i \in D, \phi(x) = 0 \\ \text{and} & H(x)B_x > 0 \\ H(x) & \text{otherwise} \end{cases} \quad (6.43)$$

Because the solution must remain near the constraints, the full sparse-field solution is unnecessary, and one can instead use a static, narrow band. The appropriate narrow-band algorithm is as follows:

1. Construct an initial solution $u_{i,j,k}^0 = B_{i,j,k}$.
2. Find all of the grid-points in $u_{i,j,k}^0$ that lie adjacent to one or more grid-points of opposite sign. Call this set \mathcal{A}_0 .
3. Find the set all of the grid-points that are adjacent to \mathcal{A}_0 and denote it \mathcal{A}_1 . Repeat this for $\mathcal{A}_2, \mathcal{A}_3, \dots, \mathcal{A}_M$, to create a band that is $2M+1$ wide. The union of these sets, $\mathcal{A} = \cup_{i=0}^M \mathcal{A}_i$, is the *active set*.
4. For each $(i,j,k) \in \mathcal{A}$ calculate $\Delta u_{i,j,k}^n$ using a central difference approximation to the mean curvature.
5. For each $(i,j,k) \in \mathcal{A}$ update the value of $u_{i,j,k}^{n+1}$ according to Equation 6.43.
6. Find the average change for points in the active set:

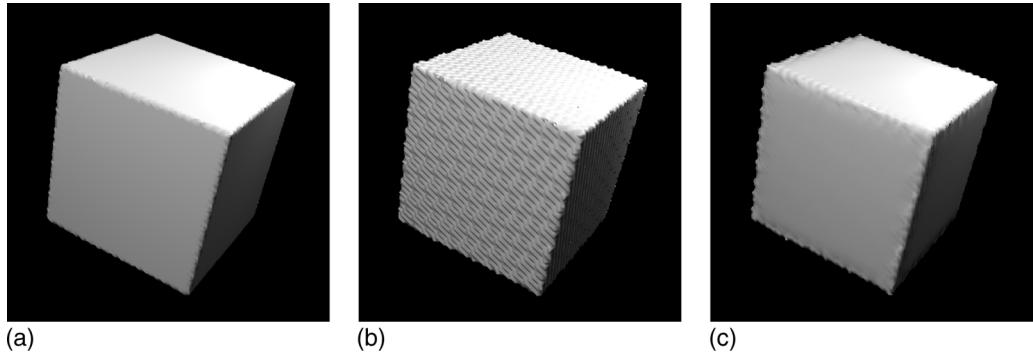


Figure 6.11 a) An ideal grey-scale embedding of a cube (i.e., distance transform) results in a smooth, accurate isosurface. b) A binary volume yields significant aliasing artifacts. c) The surface estimation from the binary data with $M = 4$ and a stopping threshold of 0.002 shows a quality that is comparable to the ideal.

$$c^n = \left(\frac{1}{|\mathcal{A}|} \sum_{\mathcal{A}} |u_{i,j,k}^{n+1} - u_{i,j,k}^n|^2 \right)^{\frac{1}{2}}. \quad (6.44)$$

7. If c^n is below some predefined threshold, then the algorithm is complete; otherwise, increment n and go to step 4.

Figures 6.11a and 6.11b show the zero sets of grey-scale and binary volumes of a cube. The grey-scale volume is the distance transform. The cubes shown are rotated 22.5° around each axis (in order to create significant aliasing artifacts in the binary version), and each cube edge has a length of 50 grid units. Figure 11c shows the zero set of the solution to the constrained minimization problem with a stopping threshold of 0.002 using a narrow band of $M = 4$. These results show significant improvements in the aliasing, especially along the flat faces where the minimum-surface-area approach is most appropriate. On the corners and edges, artifacts remain, because the algorithm is trying to stretch the minimal surface across the constraints, which contain jaggies. The algorithm converges rapidly, in about 20 iterations. Experiments show that the choices of band width and stopping threshold do not affect the results in any significant way, provided that the width is sufficiently large and the stopping threshold is sufficiently small.

Figure 12 shows a series of before (left) and after (right) isosurface renderings. Generally the algorithm succeeds in reducing aliasing artifacts with a minimal distortion of the shapes. For some shapes, such as the low-resolution torus, the aliasing is reduced, but only marginally so, which demonstrates a fundamental limitation of the proposed algorithm; *the minimal surface criterion does not always get the solutions close to the ideal*. Instead, the solution is stretching across the rather coarse features formed by the binary volumes. This is especially bad in cases such as a torus, which includes points for which one principal curvature is significantly greater than the other, causing the surface to *pucker* inward, leaving pronounced aliasing artifacts. On flat surfaces or those with higher resolution, the aliasing effects are virtually eliminated.

6.7.4 Surface Reconstruction and Processing

The ability to compute free-form surface deformations independent of topology or complexity opens up new possibilities in reconstructing and processing surfaces. For instance, in building 3D models from multiple laser range (ladar) images, one can express the likelihood of a closed surface as a function of an integral over

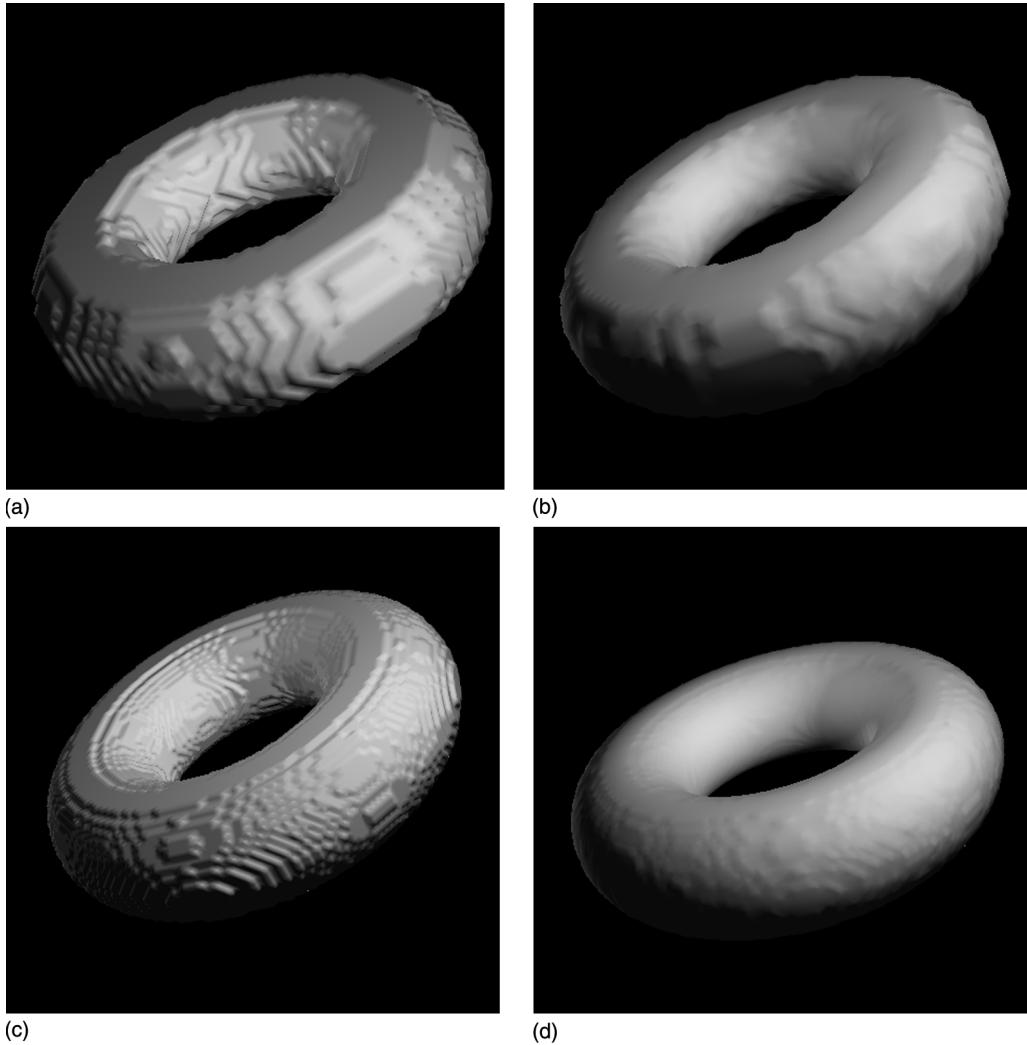


Figure 6.12 Left) Binary input volumes. Right) Results of surface estimation. Top) low-resolution torus. Bottom) higher-resolution torus.

the enclosed solid [29,54]. Using a gradient descent to minimize the likelihood gives rise to a data-driven deformation that fits a surface model to a collection of noisy ladar data. Combining the likelihood with an area or curvature-based prior and embedding the motion in the level-set framework generates a PDE for 3D surface reconstruction:

Q12

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| G(\mathbf{x}, \mathbf{n}_\phi) + \beta |\phi| \mathcal{P} \quad (6.45)$$

where G , the fitting term, depends on the set of input data and a sensor model, while \mathcal{P} , which depends on derivatives of ϕ , is the first variation of the log prior. For examples, \mathcal{P} is the mean curvature in the case of a surface area prior. Fig. 6.13 shows a surface rendering of noisy range

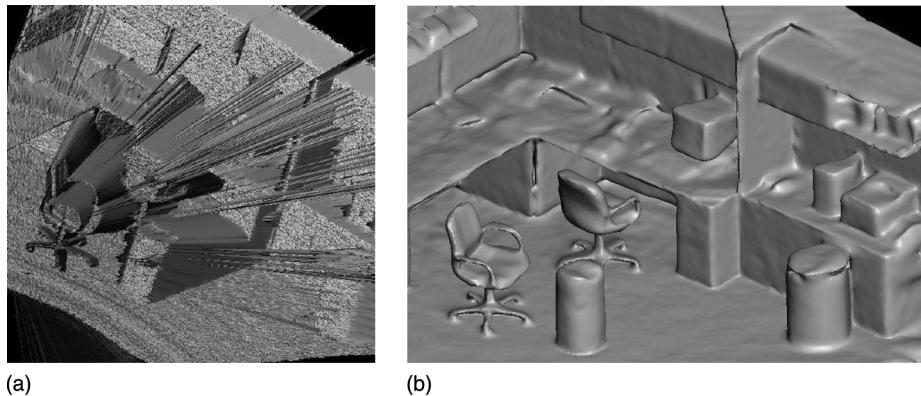


Figure 6.13 a) A surface rendering of a noisy range image. b) A 3D reconstruction obtained by fitting a level-set model to 12 noisy range images from different points of view.

data and a 3D reconstruction using a level-set model and the curvature-based prior described Tasdizen and Whitaker [55]. The ability to systematically combine data from different points of view and incorporate a smoothing prior that preserves creases results in 3D reconstructions that exceed the accuracy of the laser range finder.

This same strategy applies to other imaging modalities. For instance, the problem of recon-

structing 3D interfaces from tomographic projections leads to a formulation very similar to Equation 6.45, in which the data term depends on the set of input data and the shape of the surface estimate [9]. This is an important problem in situations where one is given limited or incomplete tomographic data, such as in transmission electron microscopy. Fig. 6.14 shows a TEM surface reconstruction of a spiny dendrite

Q13

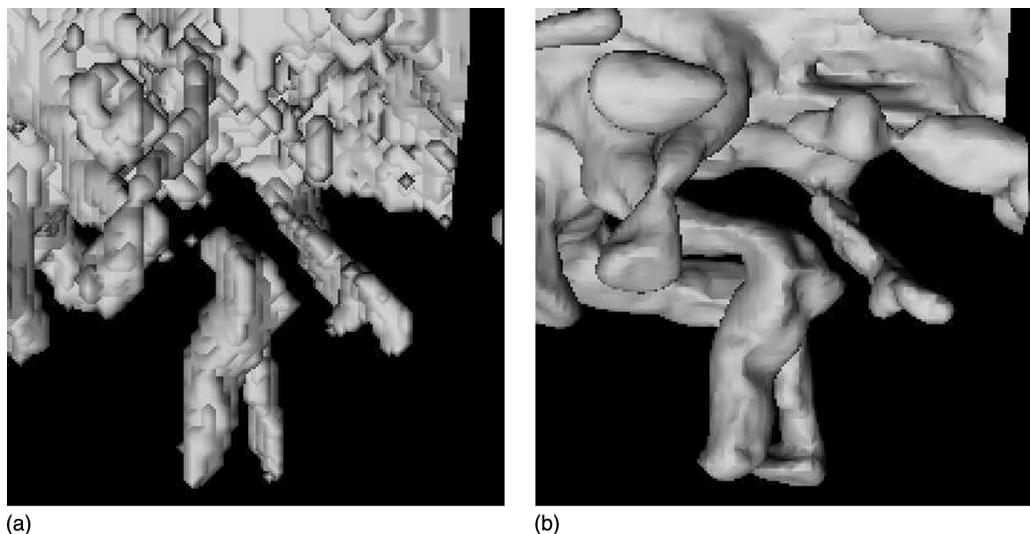


Figure 6.14 a) An initial model constructed from a back projection. b) The model deforms to minimize the discrepancy with the projected data, and forms new connections in the process.

using this strategy. The application of level-sets for this problem is important, because the complex topology of this model changes during the fitting process.

A problem with relating the reconstruction is that of surface processing, which has gained importance as the number of 3D models grows. One would like to have the same set of tools for manipulating surfaces as exists for images. This would include cutting and pasting, blending, enhancement, and smoothing. This is especially important in visualization, where the 3D models are often derived from measured data and are therefore noisy or incomplete or somehow imperfect. Level-set models provide some mechanisms for filtering surfaces in a way that does not depend on a particular parameterization or topology. Tasdizen et al. [7] describe a strategy for filtering level-set surfaces that relies on processing a *fourth-order* geometric flow. Using this strategy one can generalize a wide range of image-processing algorithms to surfaces. Fig. 6.15 shows a generalization of anisotropic diffusion

[56] to surfaces in a way that enhances sharp creases. Fig. 6.16 shows a generalization of unsharp masking (a form of high-boost filtering), which brings out surface detail.

6.8 Summary

Volumes provide a powerful tool for modeling deformable surfaces, especially when one is dealing with measured data. With measured data, the shape, topology, and complexity of the surface are dictated by the application rather than the user. Implicit deformable surfaces, implemented as level-sets, provide a natural mechanism for processing such data in a manner that relieves the user of having to decide on an underlying parameterization. This technology easily handles the many degrees of freedom that are important to capturing the fine detail of measured data. Furthermore, the level-set approach provides a powerful mechanism for constructing *geometric flows*, which results in output that depends only on the shape of input (and the

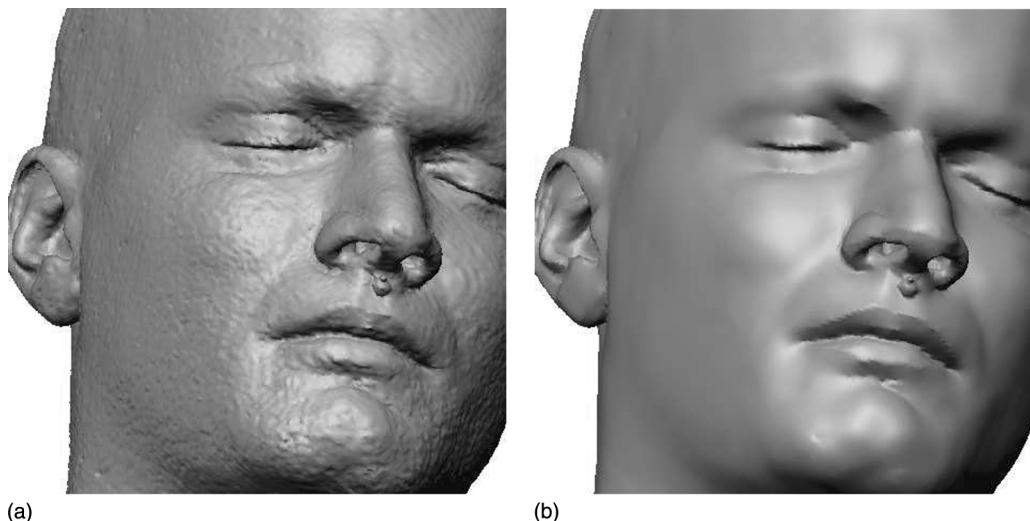


Figure 6.15 a) A noisy isosurface obtained from an MRI volume. b) Processing with feature-preserving smoothing alleviates noise while enhancing sharp features.

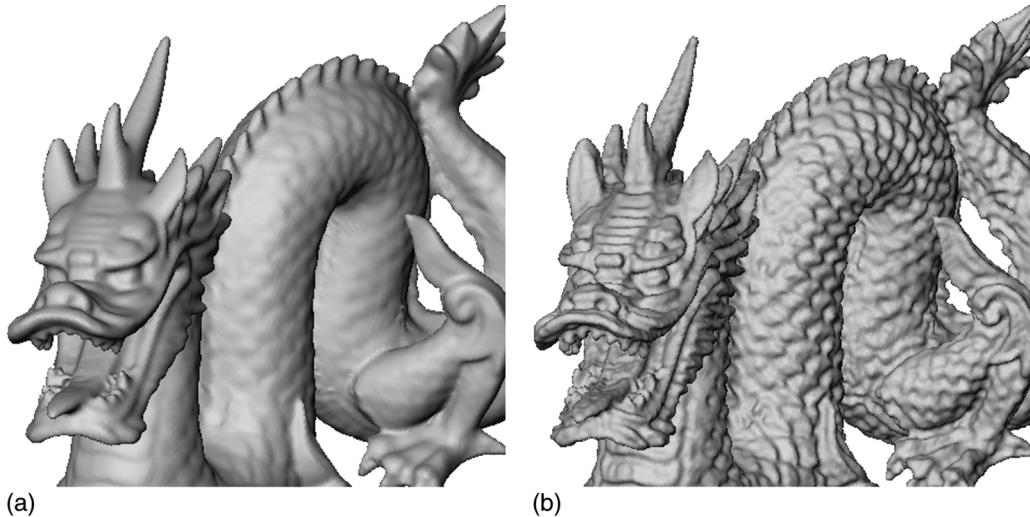


Figure 6.16 A volumetric surface model (a) is enhanced via unsharp masking (b).

resolution) and does not produce artifacts that are tied to an arbitrary, intermediate parameterization.

References

- Q14
1. S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, vol. 79, pages 12–49, 1988.
 2. J.A. Sethian. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.
 3. R. Fedkiw and S. Osher. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.
 4. R.T. Whitaker. Volumetric deformable models: active blobs. In *Visualization In Biomedical Computing 1994* (R.A. Robb, ed.), (Mayo Clinic, Rochester, Minnesota), pages 122–134, SPIE, 1994.
 5. R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape modeling with front propagation: a level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.
 6. R. Whitaker, D. Breen, K. Museth, and N. Soni. A framework for level set segmentation of volume datasets. In *Proceedings of ACM Int'l. Wkshp. on Volume Graphics*, pages 159–168, June 2001.
 7. T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. *ACM Trans. on Graphics* (to appear).
 8. D. Breen and R. Whitaker. A level-set approach to 3D shape metamorphosis. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.
 9. R. Whitaker and V. Elangovan. A direct approach to estimating surfaces in tomographic data. *Medical Image Analysis*, 6:235–249, 2002.
 10. K. Museth, D. Breen, L. Zhukov, and R. Whitaker. Level-set segmentation from multiple non-uniform volume datasets. In *IEEE Visualization 2002*, pages 179–186, October 2002.
 11. J. Blinn. A generalization of algebraic surface drawing. *ACM Trans. on Graphics*, vol. 1, pages 235–256, March 1982.
 12. S. Muraki. Volumetric shape description of range data using “blobby model.” *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol. 25, pages 227–235, July 1991.
 13. D.E. Breen, S. Mauch, and R.T. Whitaker. 3D scan conversion of CSG models into distance volumes. In *The 1998 Symp. on Volume Visualization*, pages 7–14, October 1998.
 14. W. Lorensen and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
 15. M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 9(3):245–261, 1990.

- Q15**
16. R. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.
 17. G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: methods and applications. In *Proc. IEEE Visualization 2003* (to appear), October 2003.
 18. J.J. Koenderink. *Solid Shape*. Cambridge, Mass, MIT press, 1990.
 19. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1:321–323, 1987.
 20. D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, vol. 4, pages 306–331, December 1988.
 21. S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
 22. J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4):1591–1595, 1996.
 23. J.A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Sciences*. Cambridge University Press, 1996.
 24. S. Osher and R. Fedkiw. Level set methods. an overview and some recent results. Tech. Rep. 00–08, UCLA Center for Applied Mathematics, Department of Mathematics, University of California, Los Angeles, 2000.
 25. L. Alvarez and J.-M. Morel. A morphological approach to multiscale analysis: from principles to equations. In *Geometry-Driven Diffusion in Computer Vision* (B.M. ter Haar Romeny, ed.), pages 4–21, Kluwer Academic Publishers, 1994.
 26. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *Fifth International Conference on Computer Vision*, pages 694–699, IEEE Computer Society Press, 1995.
 27. R. Kimmelman and A. Bruckstein. Shape offsets via level sets. *Computer Aided Design*, 25(5):154–162, 1993.
 28. R. Whitaker and D. Breen. Level-set models for the deformation of solid objects. In *The Third International Workshop on Implicit Surfaces*, pages 19–35, Eurographics, 1998.
 29. R.T. Whitaker. A level-set approach to 3D reconstruction from range data. *International Journal of Computer Vision*, pages 203–231, 1998.
 30. T. Chan and L. Vese. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3):271–293, 2000.
 31. R.T. Whitaker. Algorithms for implicit deformable models. In *Fifth International Conference on Computer Vision*, IEEE Computer Society Press, 1995.
 32. S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Gradient flows and geometric active contour models. In *Fifth Int. Conf. on Comp. Vision*, pages 810–815, IEEE Computer Society Press, 1995.
 33. A. Yezzi, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum. A geometric snake model for segmentation of medical imagery. *IEEE Transactions on Medical Imaging*, vol. 16, pages 199–209, April 1997.
 34. L. Lorigo, O. Faugeras, W. Grimson, R. Keriven, and R. Kikinis. Segmentation of bone in clinical knee MRI using texture-based geodesic active contours. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI '98)* (W. Wells, A. Colchester, and S. Delp, eds.), pages 1195–1204, October 1998.
 35. R. Whitaker and X. Xue. Variable-conductance, level-set curvature for image denoising. In *IEEE International Conference on Image Processing*, pages 142–145, October 2001.
 36. D. Adalsteinson and J.A. Sethian. A fast level set method for propagating interfaces, *Journal of Computational Physics*, pages 269–277, 1995.
 37. J. Kent, W. Carlson, and R. Parent. Shape transformation for polyhedral objects. *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, pages 47–54, July 1992.
 38. J. Rossignac and A. Kaul. AGRELS and BIPs: metamorphosis as a bezier curve in the space of polyhedra. *Computer Graphics Forum (Eurographics '94 Proceedings)*, 13(9):C-179–C-184, 1994.
 39. J. Hughes. Scheduled fourier volume morphing. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(7):43–46, 1992.
 40. A. Lerios, C. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *SIGGRAPH '95 Proceedings*, pages 449–456, August 1995.
 41. B. Payne and A. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, 1992.
 42. D. Cohen-Or, D. Levin, and A. Solomivici. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2): 117–140, 1998.
 43. P. Getto and D. Breen. An object-oriented architecture for a computer animation system. *The Visual Computer*, 6(3):79–92, 1990.
 44. K. Museth, D. Breen, R. Whitaker, and A. Barr. Level-set surface editing operators. In *ACM SIGGRAPH*, pages 330–338, 2002.

- 45. P. Maragos. Differential morphology and image processing. *IEEE Transactions on Image Processing*, 5(6):922–937, 1996.
- 46. R. Whitaker. Reducing aliasing artifacts in isosurfaces of binary volumes. In *IEEE Symp. on Volume Visualization and Graphics*, pages 23–32, October 2000.
- 47. S. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *1998 Symposium on Volume Graphics*, pages 23–30, ACM SIGGRAPH, 1991.
- 48. S. Wang and A. Kaufman. Volume-sampled 3D modeling. *IEEE Computer Graphics and Applications*, 14(5):26–32, 1994.
- 49. S. Wang and A. Kaufman. Volume-sampled voxelization of geometric primitives. In *Proceedings of the 1993 Symposium on Volume Visualization*, pages 78–84, ACM SIGGRAPH, October 1993.
- 50. U. Tiede, T. Schiemann, and K. Höhne. High quality rendering of attributed volume data. In *IEEE Visualization 1998*, pages 255–262, October 1998.
- 51. M.I. Miller and B. Roysam. Bayesian image reconstruction for emission tomography incorporating Good’s roughness prior on massively parallel processors. *Proc. Natl. Acad. Sci.*, 88:3223–3227, 1991.
- 52. T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.
- 53. L. Cooper and D. Steinberg. *Introduction to Methods of Optimization*. New York, W.B. Saunders Company, 1970.
- 54. R. Whitaker and J. Gregor. A maximum likelihood surface estimator for dense range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1372–1387, 2002.
- 55. T. Tasdizen and R. Whitaker. Higher-order nonlinear priors for surface reconstruction. *IEEE Trans. on PAMI* (to appear).
- 56. P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 12, pages 629–639, 1990.

AUTHOR QUERIES

- Q1 Au: which sections?
- Q2 Au: ok?
- Q3 Au: ok?
- Q4 Au: as meant?
- Q5 Au: as meant?
- Q6 Au equation missing or delete number?
- Q7 Au ok?
- Q8 Au: ok?
- Q9 Au: ok?
- Q10 Au: ok?
- Q11 Au: source?
- Q12 Au: ok?
- Q13 Au: ok?
- Q14 Au: location?
- Q15 Au: location?
- Q16 Au: location?

