LEVEL-SET MODELS FOR THE DEFORMATION OF SOLID OBJECTS

Ross T. Whitaker

Department of Electrical Engineering
University of Tennessee
Knoxville, TN 37996-2100

David E. Breen

Computer Graphics Laboratory
California Institute of Technology
Pasadena, CA 91125

ABSTRACT

This paper presents a general approach to modeling surface deformations of solid objects using *level sets*. Level-set models encode the shapes of 3D objects as the iso-surfaces of dense scalar fields. The embedding is stored as a discretely-sampled, rectilinear grid, i.e. a volume. Thus, level-set models can be described as a class of implicit models with basis functions that are fixed in number and location (grid points) and have a finite extent. The advantage of this representation is its ability to model incremental deformations in shape. We have shown that level-set models can mimic parametric surface models that seek to minimize various kinds of energy functionals. At the same time level-set models offer a number of advantages, including flexible topology, no need for reparameterization, concise descriptions of differential structure, and a natural scale space for hierarchical representations. These methods have a wide range of applications in 3D surface modeling.

This paper presents the mathematics for embedding deformable surfaces as level sets within volumes. It also describes our work in developing numerical algorithms for computing the resulting evolution equations. These algorithms allow level-set models to be manipulated and positioned to sub-voxel accuracy. This paper also presents some of our work in the application of level-set models to several different problems in 3D graphics: 3D surface morphing, filleting and blending, and 3D reconstruction from multiple range maps.

1. INTRODUCTION

Surface deformations are an important part of 3D graphics. The ability to perturb the shape of a surface in order to meet certain criteria, whether it be the discerning eye of a human designer or a set of input data from some measuring device, is essential to the creation of useful 3D models. Surface deformations have played a role in a variety of application areas, including shape design [28], data fitting [20, 22], and physically-based animation [27].

The mathematical tools for manipulating 3D shapes are intimately tied to the mechanisms used for representing those shapes. Conventionally, deformable models are parameterized. That is, they are mappings from one space, which coincides with the dimensionality of the model, to another, which is the range. Typically, these parameterizations rely on a set of basis functions that span some finite subspace of all possible shapes.

In the context of deformations, parametric models have several drawbacks which make them inadequate for certain kinds of applications. The dependency on parameterization and the associated basis is critical; it limits the kinds of shapes a model can represent. Models typically do not deform far from their initial conditions without some kind of reparameterization [22]. Such reparameterizations are often inefficient and developed ad hoc according to heuristics. The parameterization can also make it difficult to measure the differential geometry of the model, as with polygonal meshes.

An alternative to a parametric model is an implicit model, i.e., specifying a model as a level set of a scalar function, Φ. Such a 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 thousands of such primitives. This is the strategy behind the "blobby" models proposed by Blinn [2].

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 $\mathbf{x} \in \mathcal{S}$ 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 $\mathbf{x}$ in a way that maintains continuity with

RTW support provided by the Volume Visualization program of the Office of Naval Research grant N00014-97-0227. DEB support provided by the National Science Foundation (ASC-89-20219), as part of the STC for Computer Graphics and Scientific Visualization.

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 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.

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. The total number of basis functions is fixed, as are their positions (grid points) and extent. One can change the magnitude of each basis function, but 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.

The advantage of using discretely-sampled volumes to represent implicit surfaces is that they are well suited to modeling local surface deformations. Sethian [26] has proposed the use of level sets to model certain kinds of physical problems. Several authors have proposed similar paradigms for computer vision [18, 31, 19, 5] — mostly for segmenting images and volumes. Level-set models have been shown to mimic the conventional parametric models that seek to minimize certain kinds of energy functionals. A well-founded mathematical structure leads to a set of rules that describe how voxel values can be manipulated to create deformations in the level sets. However, level-set models as proposed in the literature have some drawbacks. In particular, they are computationally burdensome and they cannot be positioned to sub-voxel accuracy. These drawbacks are critical in a number of computer-graphics applications. Our work has focused on two areas. First is overcoming these shortcomings in level-set modeling through improved numerical algorithms. Second is the development of local deformation processes to solve particular problems in computer graphics.

This paper proceeds as follows. In the next section we present the mathematics for embedding a deformable surface as a level set of a volume. Deformations of a surface are re-expressed in terms of the shape of the surface and then described by the differential structure of the embedding, or volume. The result is an evolution equation on a volume which governs the behavior of the level surfaces that lie within it. That is, a set of rules for changing the values of voxels in order to achieve movements of surfaces that have a precise geometric description. We also briefly describe some of the numerical algorithms that make such deformations computable.

We then show how the level-set paradigm offers, within a single, elegant mathematical framework, new solutions for three different application areas in 3D graphics. The first, described in Section 3, is 3D surface morphing. Surfaces can move on the distance transform of other surfaces, creating a continuous family of intermediate surfaces that are not limited by topology and require little or no user input. The second application, described in Section 4, is filleting and blending. The movement of surfaces according to selected components of their curvature can produce a "filling in" at sharp concave or hyperbolic regions. The third application, in Section 5, is 3D reconstruction from multiple range maps. Multiple laser range maps are combined using an objective function that accounts for discrepancies and varying confidence levels in the measurements. An object-based smoothing function is incorporated in a way that does not introduce view dependent artifacts. This smoothing can control the influence of noise and other high-frequency scanning artifacts.

2. LEVEL-SET MODELS

This section begins with mathematics for describing surface deformations on parametric models. The result is an evolution equation for a surface. Each of the terms in this evolution equation can be re-expressed 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.

2.1. Surface Evolutions

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

$$\mathbf{S} : \underset{r}{U} \times \underset{s}{U} \mapsto \underset{x,y,z}{\mathbb{R}^3}, \qquad (1)$$

where $U \subset \mathbb{R}$, 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 $\mathbf{N} = \mathbf{N}(r, s, t)$. The set $\mathcal{S}_t$ is the range of **S** evaluated at time t with a domain $U \times U$.

The deformations of **S** are described by an evolution equation, which is typically a differential equation on $\mathbf{S}$ that incorporates certain physical properties of the surface as well as responses to other forcing functions. That is,

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{G}(\mathbf{S}_r, \mathbf{S}_s, \mathbf{S}_{rr}, \mathbf{S}_{rs}, \mathbf{S}_{ss}, \ldots), \tag{2}$$

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 [14, 27], $\mathbf{f} : \mathrm{I\!R}^3 \mapsto \mathrm{I\!R}^3$, that is

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{f}(\mathbf{S}). \tag{3}$$

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

$$\frac{\partial \mathbf{S}}{\partial t} = w(\mathbf{S})\mathbf{N} \tag{4}$$

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

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{S}_{rr} + \mathbf{S}_{ss} \tag{5}$$

describes a surface that moves in way that is becomes more *smooth* with respect to its own parameterization. This motion can be combined with the motion of Equation 3 to produce a model that is pushed by a forcing function but maintains a certain smoothness in its shape and parameterization. There are a myriad of terms that depend on both the differential geometry of the surface and outside forces or functions to control the evolution of a surface.

2.2. Embedding Deformable Models

An alternative to a parametric model, is a level-set model [26], i.e., a model that treats a 3D surface as the level-set of a discretely-sampled volume. Previous work has shown promising results with this modeling technology for 3D reconstruction, primarily in the context of 3D medical data [19, 33, 31]. This section gives the formulation for this modeling strategy, starting with an equation of motion for a deformable surface [33].

The strategy relies on the properties of regular surfaces, which have local parameterizations that can be expressed in terms of differential quantities such as arc length, curvature, etc. If one starts with a parametric surface and the associated equations of motion, and then removes the parameterization from the deformable model, all that remains is the geometry of the surface itself, which can expressed in terms of the embedding, Φ. The strategy for embedding active surfaces consists of four steps:

1. Express the equations of motion for a deformable model in terms of some unspecified parameterization.
2. Describe the parameterization in terms of the differential structure of the model.
3. Assume the model is the level set of a function Φ.
4. Express the geometry of the level set in terms of the differential structure of Φ and create an evolution equation for Φ.

In order to apply this strategy to a deformable surface $\mathcal{S}_t$, represent $\mathcal{S}_t$ as a level set of a 3D scalar function, $\Phi : \mathrm{I\!R}^3 \times \mathrm{I\!R}^+ \mapsto \mathrm{I\!R}$, which evolves over time. The evolution equations of the individual level surfaces imply corresponding evolution equations for the scalar function $\Phi(\mathbf{x}, t)$, where $\mathbf{x} \in \mathrm{I\!R}^3$.

A parametric deformable model, $\mathbf{S}$ where $r, s \in U \subset \mathrm{I\!R}$, can be represented

$$\mathcal{S}_t = \{\mathbf{x} | \Phi(\mathbf{x}, t) = k\}. \tag{6}$$

The function $\mathbf{S}$ is a parameterized version of $\mathcal{S}_t$, and the surface $\mathcal{S}_t$ remains a level set of Φ over time. Therefore the time derivative is zero:

$$\frac{\partial \Phi(\mathbf{S}, t)}{\partial t} + \nabla\Phi(\mathbf{S}, t) \cdot \frac{\partial \mathbf{S}}{\partial t} = 0, \tag{7}$$

where

$$\nabla\Phi(\mathbf{x}) \stackrel{\text{def}}{=} \left(\frac{\partial \Phi}{\partial x}, \frac{\partial \Phi}{\partial y}, \frac{\partial \Phi}{\partial z}\right). \tag{8}$$

Thus,

$$\frac{\partial \Phi}{\partial t} = -\nabla\Phi \cdot \frac{\partial \mathbf{S}}{\partial t} = |\nabla\Phi| \frac{\partial \mathbf{S}}{\partial t} \cdot \mathbf{N}, \tag{9}$$

where $\mathbf{N} = -\nabla\Phi/|\nabla\Phi|$ is the surface normal. Equations 7–9 describe the evolution of Φ in terms of the evolution of its level surfaces, but it assumes a parametric representation of the level surfaces. The next step is to re-express certain kinds of motion equations on Φ in terms of the *shape* of the surfaces (i.e., geometry that is independent of the parameterization), and thereby remove the parametric surface altogether.

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 on levels sets, there can be no reference to the underlying surface parameterization (terms depending on r and s in Equations 1 through 5). This has two important implications: 1) only those surface movements that are normal to the surface are represented (any other movement is the equivalent of a reparameterization) 2) 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 $\mathbf{S}_{rr}+\mathbf{S}_{ss}$ from equation 5. If r, s is an orthonormal parameterization, the effect of that term is purely on surface shape, not on the parameterization, and the expression $\mathbf{S}_{rr}+\mathbf{S}_{ss}$ is twice the *mean curvature*, H, of the surface. The corresponding level-set formulation is given by Sethian [26]:

$$\begin{aligned} H \;=\; & \frac{1}{2}\left(\Phi_x^2+\Phi_y^2+\Phi_z^2\right)^{-\frac{1}{2}}\left[\left(\Phi_y^2+\Phi_z^2\right)\Phi_{xx}\right. \\ & +\left(\Phi_z^2+\Phi_x^2\right)\Phi_{yy}+\left(\Phi_x^2+\Phi_y^2\right)\Phi_{zz} \\ & \left.-2\Phi_x\Phi_y\Phi_{xy}-2\Phi_y\Phi_z\Phi_{yz}-2\Phi_z\Phi_x\Phi_{zx}\right] \end{aligned} \quad (10)$$

Table 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 the parameterization that give rise to the level-set expressions. The term K is the Gaussian curvature and is given in [26] to be

$$\begin{aligned} K \;=\; & \left(\Phi_x^2+\Phi_z^2+\Phi_z^2\right)^{-2}\left[\Phi_x^2\left(\Phi_{yy}\Phi_{zz}-\Phi_{yz}^2\right)\right. \\ & +\Phi_y^2\left(\Phi_{xx}\Phi_{zz}-\Phi_{xz}^2\right)+\Phi_z^2\left(\Phi_{xx}\Phi_{yy}-\Phi_{xy}^2\right) \\ & +2\left[\Phi_x\Phi_y\left(\Phi_{xz}\Phi_{yz}-\Phi_{xy}\Phi_{zz}\right)\right. \\ & +\Phi_y\Phi_z\left(\Phi_{xy}\Phi_{xz}-\Phi_{yz}\Phi_{xx}\right) \\ & \left.\left.+\Phi_x\Phi_z\left(\Phi_{xy}\Phi_{yz}-\Phi_{xz}\Phi_{yy}\right)\right]\right] \end{aligned} \quad (11)$$

2.3. Numerical Algorithms

The level-set terms in Table 1 are combined, based on the needs of the application, to create a partial differential equation on $\Phi(\mathbf{x}, t)$. The next step is to solve this partial differential equation on u, a discrete approximation to Φ. The terms in Table 1 fall into two basic categories: the first-order terms (items 1 and 2 in Table 1) 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. Osher and Sethian [24] have proposed an *up-wind* scheme. The up-wind method relies on a one-sided derivative that looks in the up-wind direction of the moving wavefront, and thereby avoids the over-shooting associated with finite forward differences.

The second-order terms in Table 1 generally do not require the up-wind methods and can be solved with a finite forward difference schemes using centralized differences for the derivatives. For example,

$$u_x(x,y,z) \;=\; \frac{u(x+h,y,z)-u(x-h,y,z)}{2h} \quad (12)$$

$$\begin{aligned} u_{yy}(x,y,z) \;=\; & \frac{1}{h^2}\left[u(x,y+h,z)+\right. \\ & \left.u(x,y-h,z)-2u(x,y,z)\right], \end{aligned} \quad (13)$$

$$\begin{aligned} u_{xy}(x,y,z) \;=\; & \frac{1}{4h^2}\left[u(x+h,y+h,z)\right. \\ & +u(x-h,y-h,z) \\ & -u(x+h,y-h,z) \\ & \left.-u(x-h,y+h,z)\right]. \end{aligned} \quad (14)$$

2.4. Sparse-Field Solutions

Up-wind solutions to the equations described in the previous section give equations of motion for level-set models over the entire range of the embedding, i.e., for all values of k in Equation 6. These solutions describe the behavior of an embedded family of contours. This family of contours can be explored by looking at the surfaces that result from the various levels of k. This dense family of solutions might be advantageous for certain applications, such as volume rendering [33]. However, some applications, such as those described in Sections 4–5 of this paper, require only a single model. In such applications the calculation of solutions over the entire range of iso-values 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 of interest.

Our work has focused on the development of an alternative numerical algorithm, called the sparse-field method, that computes the geometry of only a small subset of points in the range and requires a fraction of the computation time required by the previous algorithm. We have shown two advantages to this method [30]. First is a significant improvement in computation times. Second is increased accuracy when fitting models to forcing functions that are defined to sub-voxel accuracy.

The sparse-field method relies on the fact that when solving for only a single level set, $\Phi(\mathbf{x}, t) = k$, the evolution of Φ is important only in the vicinity of that level set. The evolution equations for Φ are such that the level sets of Φ evolve independently (to within the error introduced by the discrete grid). Thus, one should perform calculations for the evolution of Φ only in a neighborhood of the set $\mathcal{S} = \{\mathbf{x} | \Phi(\mathbf{x}) = k\}$. For u this means that there is a small subset of grid locations that are essential to any particular step in the evolution; the rest can be ignored. The challenge is keeping track of this neighborhood as the k-level set of Φ moves from one set of voxels to another.

The sparse-field algorithm does precisely the number of calculations needed to compute the next position of the level surface. It is analogous to a locomotive engine that lays down tracks before it and picks them up from behind (readers might recall such an engine in the once popular animation series, "Gumby"). These grid points can be accessed through a linked-list structure, and in this way the number of computations increases with the size of the model rather than the

	Effect	Parametric Evolution	Level-Set Evolution	Parameter Assumptions
1	External force	$\mathbf{f}$	$\mathbf{f} \cdot \nabla\Phi$	None
2	Expansion/ contraction	$\mathbf{N}w(\mathbf{S})$	$\lvert\nabla\Phi(\mathbf{x},t)\rvert w(\mathbf{x})$	None
3	Mean curvature	$S_{rr} + S_{ss}$	$H\lvert\nabla\Phi\rvert$	Orthonormal
4	Gauss curvature	$S_{rr} \times S_{ss}$	$K\lvert\nabla\Phi\rvert$	Orthonormal
5	Second order	S_{rr} or S_{ss}	$\left(H \pm \sqrt{H^2 - K}\right) \lvert\nabla\Phi\rvert$	Principle curvatures

Table 1: A list of evolution terms for parameteric models has a corresponding expression on the embedding, Φ, associated with the level-set models.

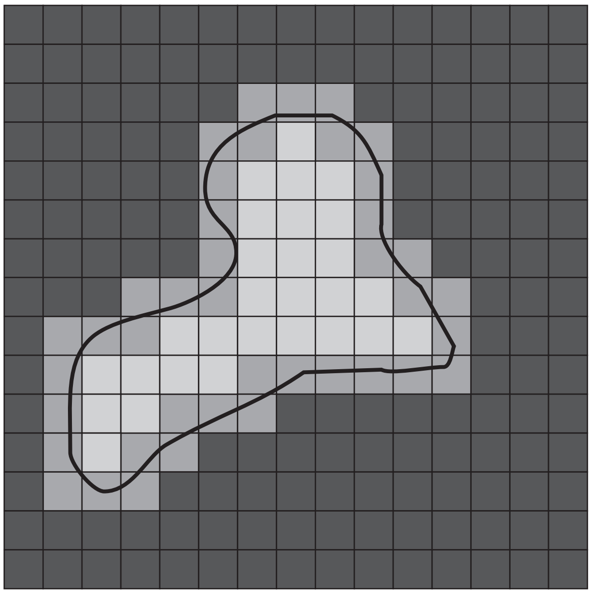


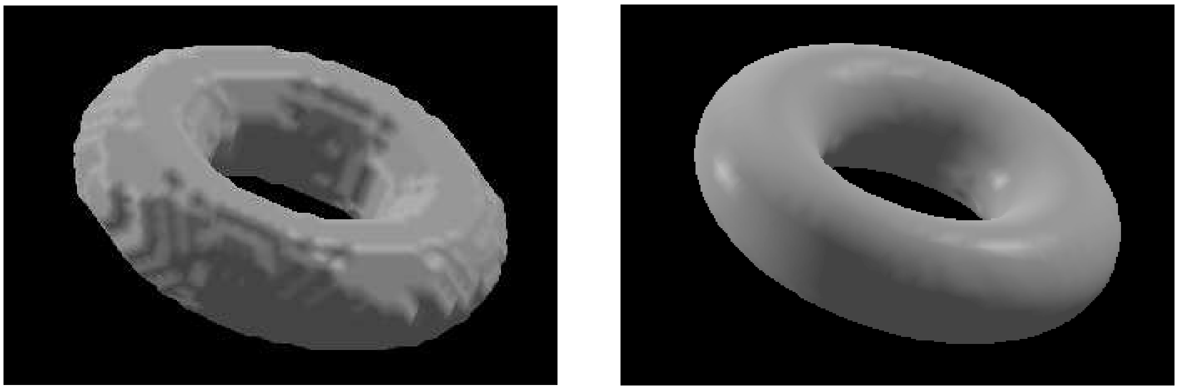
Figure 1: A level curve of a 2D scalar field passes through a finite set of grid points. Only those grid points and their nearest neighbors are relevant to the evolution of that curve.

resolution of the embedding.

The set of grid points that lie adjacent to the kth level set are called the *active set*. The distance of the surface from the center of any active point is proportional (to within a first-order approximation) to the intensity of that grid point and the first derivative of u at that point. Because all of the derivatives (up to second order) in this approach are computed using nearest neighbor differences, only the active set and its immediate neighbors are relevant to the evolution of the level surface (at any one particular point in time). The evolution of the surface $\mathcal{S}$ is independent of the embedding Φ and its discrete approximation, u. Thus we choose a particular u outside of the active set and its immediate neighborhood that allows efficient computation. The resulting alogrithm is very efficient; the sparse-field algorithm has a computational complexity that grows linearly with the surface area of the model as with parametric models. Details are given in [30].

A second advantage of the sparse-field approach is resolution. The first-order terms from Table 1 push the level sets (all of them) toward places where the values of the forcing functions cross zero. The result is a *shock*, a discontinuity in Φ. In volumes these shocks take the form of high-contrast areas causing problems with aliasing in the resulting models. The results of this are often too crude for many computer-graphics applications.

When using the sparse-field method, the active points serves as a set of control points on the level set. Changing the values of these voxels changes the position of the level set. The forcing function is sampled not at the grid point, but at the location of the nearest level set, which generally lies between grid points. Using a first-order approximation to Φ produces results that avoid the aliasing problem associated with the shocks that typically occur with level-set models. Figure 2 shows the result of fitting a level-set model to a torus. In this case, we use an evolution equation of the form of Equation 4, with the signed distance



(a) (b)

Figure 2: The deformation of a cube moves downhill on the distance transform of a torus. In (a) the steady state of the dense-field approach shows that the torus suffers from aliasing due to the shocks which form near the boundary of the torus. In (b) the modified sparse-field algorithm uses a first-order approximation to position the zero-level set to sub-voxel accuracy.

transform of a torus as forcing function. Figure 2(a) demonstrates the aliasing caused by the formation of shocks on the discrete grid, while figure 2(b) shows the high degree of accuracy with which we can fit the level-set model using the sparse-field method with the first-order modification of surface position. Both the sparse-field method and the first-order modification also maintain the inherent ability of level-set models to deal with changes in surface topology and the singularities that accompany those changes. These computational developments for level-set models open up a number of interesting applications in 3D graphics, several of which are discussed in the sections that follow.

3. SURFACE MORPHING

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. There is not yet a single, general method for generating such transitional shapes. However, we have identified several desirable aspects of morphing algorithms that allow us to compare the adequacy of different approaches to surface morphing. Several desirable properties of 3D surface morphing are:

1. The transition process should begin with an *initial* surface and end with a specified *target* surface.
2. The morphing algorithm should apply to a wide range of shapes and topologies.
3. Intermediate surfaces should undergo continuous 3D transitions (rather than continuity only in the image space).
4. A 3D morphing algorithm should incorporate user input easily but should degrade gracefully without it.
5. Transitional shapes should depend only on the surface geometry of the two input shapes and user input.

These requirements are not exhaustive, but they capture many of the practical aspects of 3D morphing.

In this section we show how level-set models provide an algorithm for 3D morphing which meets most of these criteria and compare favorably with existing algorithms. Furthermore, this algorithm 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 target surfaces. The coordinate transformation can be computed automatically or using user input (as in [16]).

Much of the previous 3D morphing work has focused on morphing parametric models [15, 25] and applies to only very limited classes of shapes and topologies. Hughes [13] demonstrates how volumes can provide topological flexibility in surface morphing. However, that method suffers from several shortcomings. It introduces artifacts that must be smoothed away by yet another process, the intermediate shapes depend on the surface embedding (which violates property 5), and there is no provision for user input to control the intermediate shapes (property 4). Also, surface morphing in this way can introduce the spontaneous generation of new surfaces (violating property 3). Lerios et al. [16] have developed a volume-based scheme which incorporates user input via underlying coordinate transformations (a known generalization of image warping). However, results from this method are severely limited by the quality and quantity of user input. In the extreme case (no user input) their method produces a transparent alpha blending of surfaces with no deformation (defies property 3). Neither of the previous volume-based approaches [13, 16] have dealt with the deeper issue of deforming the level sets of a volume. A recent, related development is the use of "discrete potentials" and user-defined velocity fields to perform morphing and "coating" of implicit particle models [9].

3.1. Free-Form Deformations

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 our 3D morphing algorithm.

Let $D(\mathbf{x})$ 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

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{N}\ D(\mathbf{S}), \tag{15}$$

where $\mathbf{S}_{t=0} = A$. The free-form deformations can be combined with an underlying coordinate transformation. The strategy is to use a coordinate transformation (for instance 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

$$\mathbf{x}' = T(\mathbf{x}, \alpha), \tag{16}$$

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

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{N}\ D(T(\mathbf{S}, 1)), \tag{17}$$

and the corresponding level-set equation is

$$\frac{\partial \Phi(\mathbf{x}, t)}{\partial t} = |\nabla \Phi(\mathbf{x}, t)|\ D(T(\mathbf{x}, 1)). \tag{18}$$

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

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* as was used by [16]. 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. Our assertion, however, is that this underlying coordinate transformation can 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 surface. In [16] they 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.

Figure 3 shows a 3D model of a jet that was built using Clockworks [10], a CSG modeling system. Lerios et al. [16] demonstrate the transition of a jet to a dart, which was accomplished using 37 user-defined correspondences, roughly a hundred user-defined parameters. Figure 4 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 marching cubes [17]. These results are obtained without any user input. Distance transforms on the CSG models are computed near the level surface using an analytical description and extended into the volume using a level-set method [4]. User input, in the form of nonlinear coordinate transformations, is easily incorporated into this algorithm, and would allow users to control morphs based on a deeper understanding of the objects involved.

The application in this section shows how level-set models moving according to the first-order term given in expression 2 in Table 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 1, a second-order flow that depends on the principle curvatures of the surface itself.

4. FILLETING AND BLENDING SOLID OBJECTS

The construction of blending surfaces is an important tool in solid modeling. Geometric solid primitives and their intersections often produce sharp corners or creases that are often not consistent with the real-world objects that they are intended to represent. This section shows how blending can be described as a deformation process, where surfaces move under a geometric flow that can add or remove material based on local curvature information. The result is a method for solid object blending that does not depend on any particular model representation. Thus this method is not restricted to a specific class of shapes or topologies. Additionally, the results are invariant; they do not depend on arbitrary choices of coordinate systems or bases. The only requirement is that the blended objects must be closed surfaces with some known inside-outside function.

Surface blending techniques are typically tied very closely to the choice of geometric primitives. For instance, Middleditch and Sears [21] propose a set-theoretic method for blending solids which relies on low-order algebraic primitives. A fillet at the joint of two tori requires the solution of a degree 32 polynomial. Bloomenthal and Shoemake [3] propose a modeling system based on convolutions, which relies on a skeletonized representation of objects. In general the use of convolution to achieve deformations on implicit shapes results in shapes that reflect both the shape of the model and the embedding, Φ.

The blending method proposed in this section implements an interative smoothing scheme that smooths only along the level set; the final result is independent of the embedding. Consider the case of fillets. We propose that a fillet can be constructed from a process of "filling in" material in places of high curvature. The curvature of a level-set model can be calculated from the embedding, and the deformation of the level set is well defined by the curvature terms in Table 1.

The strategy is to construct a curvature term, k_p, that consists of only positive curvatures. [1] The principle curvatures of the level sets of Φ are functions of Φ and its derivatives. For a specific Φ the principle curvatures are functions of 3-space $k_1(\mathbf{x})$ and $k_2(\mathbf{x})$. For *adding* material the joint between two objects, we consider only the positive curvature components, i.e.,

$$\frac{\partial \Phi}{\partial t} = |\nabla \Phi| k_p = |\nabla \Phi| k_1^+ + |\nabla \Phi| k_2^+, \qquad (19)$$

where k^+ consists of only the positive parts of k and is defined as zero elsewhere. Because the use of separate curvature terms can cause over-shooting, the up-wind scheme (treating k_p as a space-varying velocity in the normal direction) is used for this evolution.

Figure 5 shows how the positive-curvature flow can be used to construct fillets. No knowledge of the underlying models is necessary. The fillets grow larger as more time passes. The physical extent or position of the fillet can be controlled by either specifying a region of action or by placing a small blob of deformable material in the joint that requires a fillet. Figure 6 shows how such a blending capability can be useful in animation. In this case a pair of superquadrics undergo a rigid transformation that controls their relative positions. Level-set models with a positive-curvature flow are used to create a smooth joint between these two primitives. Notice that the positive curvature method does not suffer from the growth or expansion artifacts that are often associated with distance-based blending methods [8].

Thus, a second-order flow can create smooth blends between objects in a way that does not require specific knowledge of the shapes or topologies of the object involved. The application in the next section, 3D scene reconstruction, shows how a combination of first-order and second-order terms from Table 1 are combined to create technique that fits models to data while maintaining certain smoothness constraints and thereby offsetting the effects of noise.

5. 3D RECONSTRUCTION FROM MULTIPLE RANGE MAPS

Level-set models are useful for problems related to 3D reconstruction. Previous work has presented level-set results derived from noisy 3D data such as MRI [32] and ultrasound [33]. In [30] we have shown how the reconstruction of objects from multiple range maps can be formulated as a problem of finding the surface that optimizes the posterior probability given a set of

[1] The sign of curvature is defined by the direction of the normals— in this work normals point into the volume enclosed by the object.

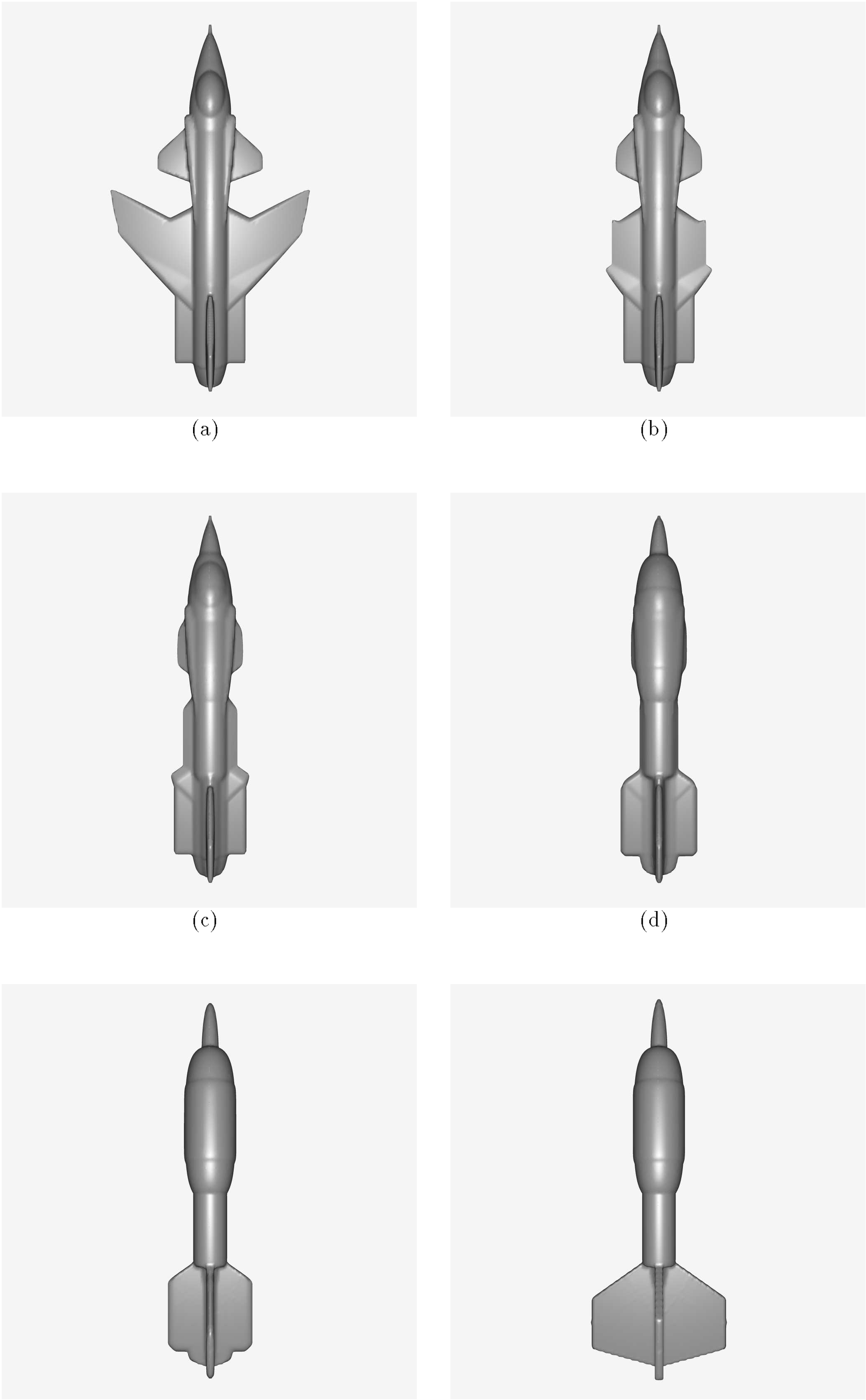
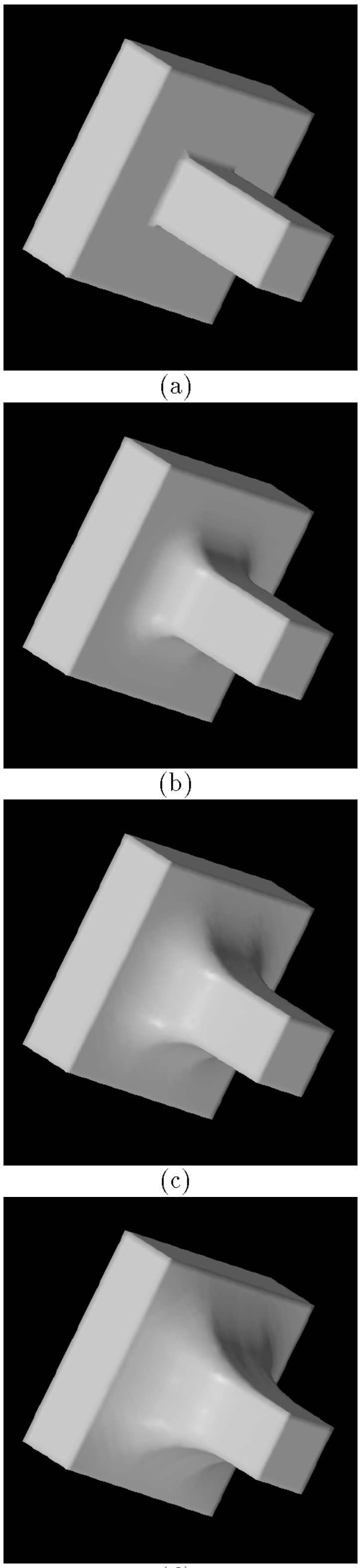

(a) (b) (c) (d) (e) (f)

Figure 4: 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.



(a)

(b)

(c)

(d)

Figure 5: Two rectangular solid models are joined by a volumetric fillet that is created from a positive curvature flow.

measurements (noisy range maps) and some information about the a-priori likelihood of different kinds of surfaces. That optimization problem can be expressed as a volume integral which can be solved with level-set models. This section presents the mathematical expressions that result from those formulations and presents some new results: the reconstruction of entire scenes by fitting level-set models to the data from a scanning LADAR (laser ranging and detection) system.

A *range map* is a collection of range measurements taken along different directions (lines of sight) but from a single point of view. Range maps could come from any number of different sources including laser scanners, structured light depth systems, shape from stereo, or shape from motion. We assume that such range maps are noisy and uncertain. The goal is to combine a number of range maps from different points of view to create a 3D structure that reflects the collective confidence and depth measures.

Several examples in the literature have applied parametric models to this task. Turk and Levoy [29], for instance, "zip" together triangle meshes in order to construct 3D objects from sequences of range maps from a laser range finder. They perform minor adjustments to the surface position in order account for ambiguity in the range maps. Their approach assumes very little noise in the input, which is reasonable given the high quality of their range maps. Chen and Medioni [6] use a parametric (triangle mesh) model which expands inside a sequence of range maps. Curless and Levoy [7] describe a volume-based technique for combining range data. They use the signed distance transform to encode volume elements with data that represent the averages (with some allowance for outliers) of multiple measurements. Surfaces of objects are the level sets of volumes. Related approaches are given in [12, 11]. Bajaj et. al. [1] use a Delaunay triangulation to impose a topology on a set of unordered 3D points and then fit trivariate Bernstein-Bezier patches—i.e. a higher-order implicit model—to the data. Muraki [23] uses implicit or blobby models to reconstruct objects from range data. The individual blobs are spherically symmetric 3D potentials that are combined linearly so that they blend together. The resulting models, with approximately 400 primitives are quite coarse.

This work differs from previous work in two ways. First, rather than heuristics, our reconstruction strategy is based on a strategy that solves for the optimal surface estimate. This optimal estimate includes information about one's expectations of the likelihood of different surfaces. The result is not a closed-form solution, but an iterative process that seeks to fit a level-set model to the data while enforcing a kind of smoothness on the data.

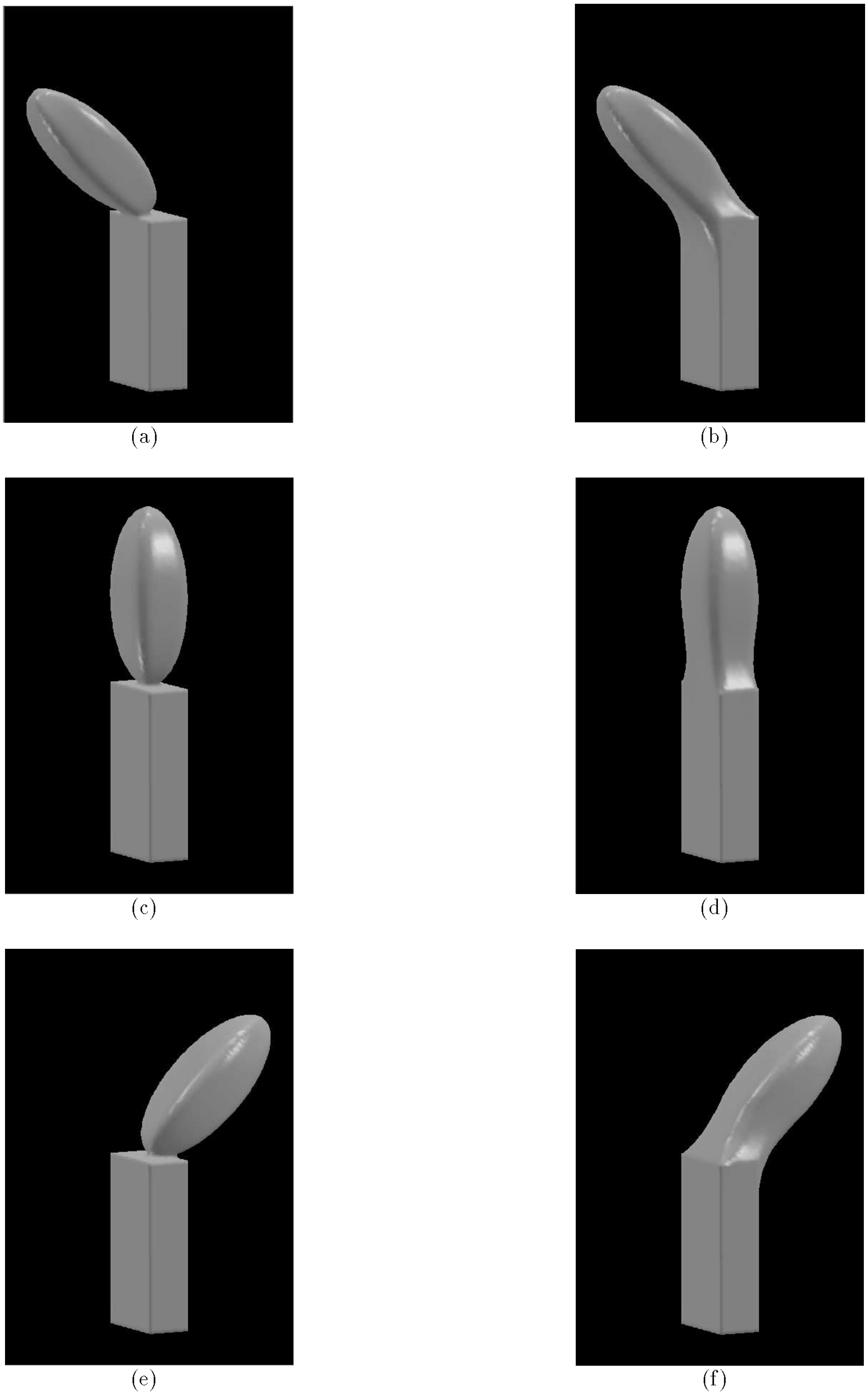


Figure 6: A short animation is created by specifying the relative motion between two superquadric components of an object. A positive-curvature flow (applied frame by frame to the joint between the two 3D models) creates a smooth, flexible object.

5.1. *Objective function for multiple range maps*

The evolution equation for the estimation of optimal surfaces is shown in [30] to consist of two parts:

$$\frac{\partial \mathbf{S}}{\partial t} = -g(\mathbf{x})\mathbf{N} + \rho(\mathbf{S}). \tag{20}$$

This first part, $-g(\mathbf{x})\mathbf{N}$, is a movement with variable speed (as in expression 2 from Table 1) that is the cumulative effect from all of the individual range maps:

$$g(\mathbf{x}) = \sum_j c^{(j)}(\mathbf{x})\ D^{(j)}(\mathbf{x})\ \omega\left(D^{(i)}(\mathbf{x})\right)\ \gamma^{(j)}(\mathbf{x}), \tag{21}$$

where D_j is the signed distance along the line of sight from a range measurement in range map j associated passing through $\mathbf{x}$. The function $\omega : \mathbb{R} \mapsto \mathbb{R}$ is a windowing function that limits the penalty of any one range measurement, and $c(\cdot)$ is a confidence function, which is inversely proportional to the level of noise in the range measurement associated with the same line of sight. The term $\gamma(\cdot)$ is an integration constant that takes into account the curvilinear coordinate system of the range scanner.

Thus, a set of range maps creates a scalar function of 3D, which describes the movement of a surface model as it seeks the optimal surface position. In the absence of a prior, $\rho = 0$, the zero set of this function is the final position (steady state) of that evolving surface. Thus, in the absence of a prior, one could sample $g(\mathbf{x})$ and obtain an approximation to the optimal surface estimate. This strategy results in an algorithm that is very much like that of [7].

There are several reasons for going to an iterative scheme for finding optimal solutions. First is the use of a prior. In surface reconstruction, even a very low level of noise can degrade the quality of the rendered surfaces in the final result, and in such cases better reconstructions can be obtained by introducing a prior. Second is aliasing. Discretizing $g(\mathbf{x})$ and finding the zero crossings will cause aliasing in those places where the transition from positive to negative is particularly steep. A deformable model can place the surface much more precisely. The third reason for going to an iterative scheme is that despite the windowing function $\omega(\mathbf{x})$ there is interference between different range maps at places of high curvature. This problem is addressed by introducing a nonlinearity which is solved in an iterative scheme given by equation 20. In this work, the solution of the linear problem, the zero set of $g(\mathbf{x})$, serves as the initial estimate for the nonlinear, iterative optimization strategy that results from the inclusion of a prior and a nonlinear term that compensates for lack of any explicit model of self occlusions.

Equation 20 includes a *prior*, which is a likelihood function on surface shape. A reasonable choice of prior is one that models objects with less surface area as more likely than objects with more surface area. Alternatively, one could say that given a set of surfaces that are near the data, the algorithm should choose a surface that has less area. Often, but not always, this will be the smoother surface. The $\rho(\mathcal{S})$ that results from this prior is the mean curvature. Therefore the evolution of the surface, using the level-set formulation, that seeks to maximize the posterior probability (given a set of range maps and a prior that penalizes surface area) is

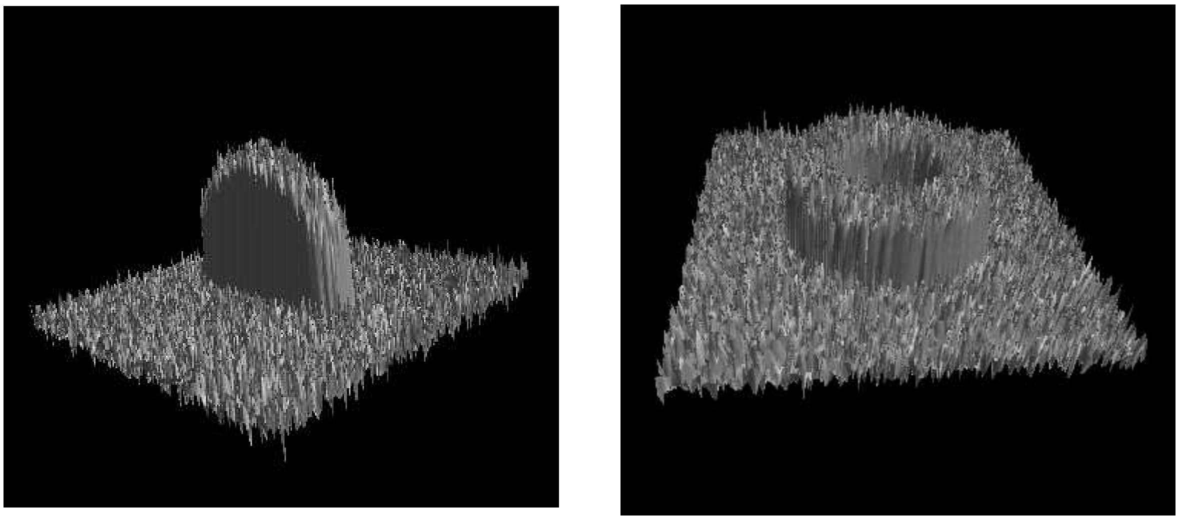
$$\begin{aligned} \frac{\partial \Phi(\mathbf{x},t)}{\partial t} \;&=\; |\nabla \Phi(\mathbf{x})| \sum_j \Bigg(D^{(j)}(\hat{\mathbf{x}})\ \omega\left(D^{(i)}(\hat{\mathbf{x}})\right) \\ &\quad \times \gamma^{(j)}(\hat{\mathbf{x}})\ c^{(j)}(\hat{\mathbf{x}}) \frac{\left(\nabla\Phi \cdot \mathbf{n}^{(j)}(\hat{\mathbf{x}})\right)^+}{\nabla\Phi \cdot \mathbf{n}^{(j)}(\hat{\mathbf{x}})} \Bigg) \\ &\quad + \beta H, \end{aligned} \tag{22}$$

where $\mathbf{n}^{(j)}(\mathbf{x})$ is the line of sight from a range finder to a 3D point, $\mathbf{x}$, β is a free parameter that controls the level of smoothing in the model, and H is the expression for the mean curvature given in equation 10.

Figure 7 shows a pair of simulated range maps constructed from an analytical description of a torus. These 200×200 pixel range maps are corrupted with additive Gaussian noise that has a standard deviation of 20% (as a function of the smaller of the two radii). Six synthetic noise-corrupted viewpoints of a torus are combined to create a level-set reconstruction of a torus. Figure 8(a) shows the initial model (80×80×40 voxels) used for fitting a level-set models to the range data. Figure 8(b) shows the result of the level-set models that uses 8(a) as an initial state and has a value of β equal to 0.5. The result is a reasonable reconstruction of the noiseless model (Figure 8(c)) which combines the six points of view and the smoothing function.

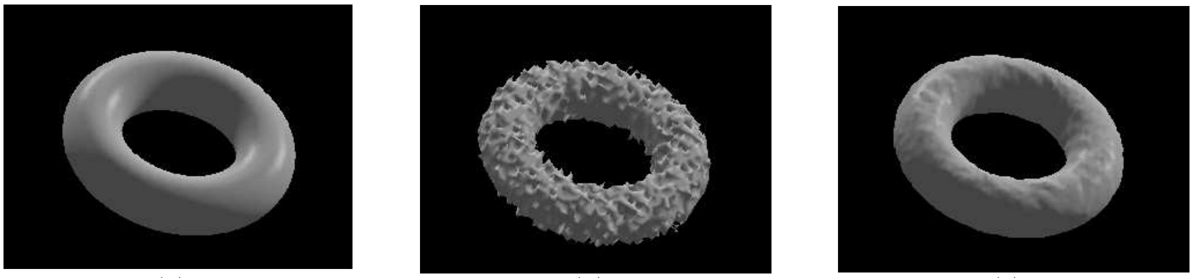
Figure 9(a) shows a range map taken with the Perceptron model P5000, an infra-red, time-of-flight laser range finder with a pan-tilt mechanism. Figure 9(b) shows the amplitudes associated with the return signal (an *intensity*), and 9(c) shows a surface plot of the range map to demonstrate the degree of noise (additive and outliers). Figure 9(d) shows the confidence values associated with those range measurements. These confidence values are derived from empirical data about the level of noise in the range finder (which depends on the return amplitude), and some analysis, from first principles, about the effects of uncertainty in the 3D positions of the scans and the model — which results in the lower confidence at edges as described in [30]. We combined twelve such views from different locations in the room to generate the results that follow.

Figure 10(a) shows the initial estimate based on the zero crossings of $g(\mathbf{x})$, and 10(b) shows the result of 32 iterations with the prior term and the correction for the surface normal direction. The size of the



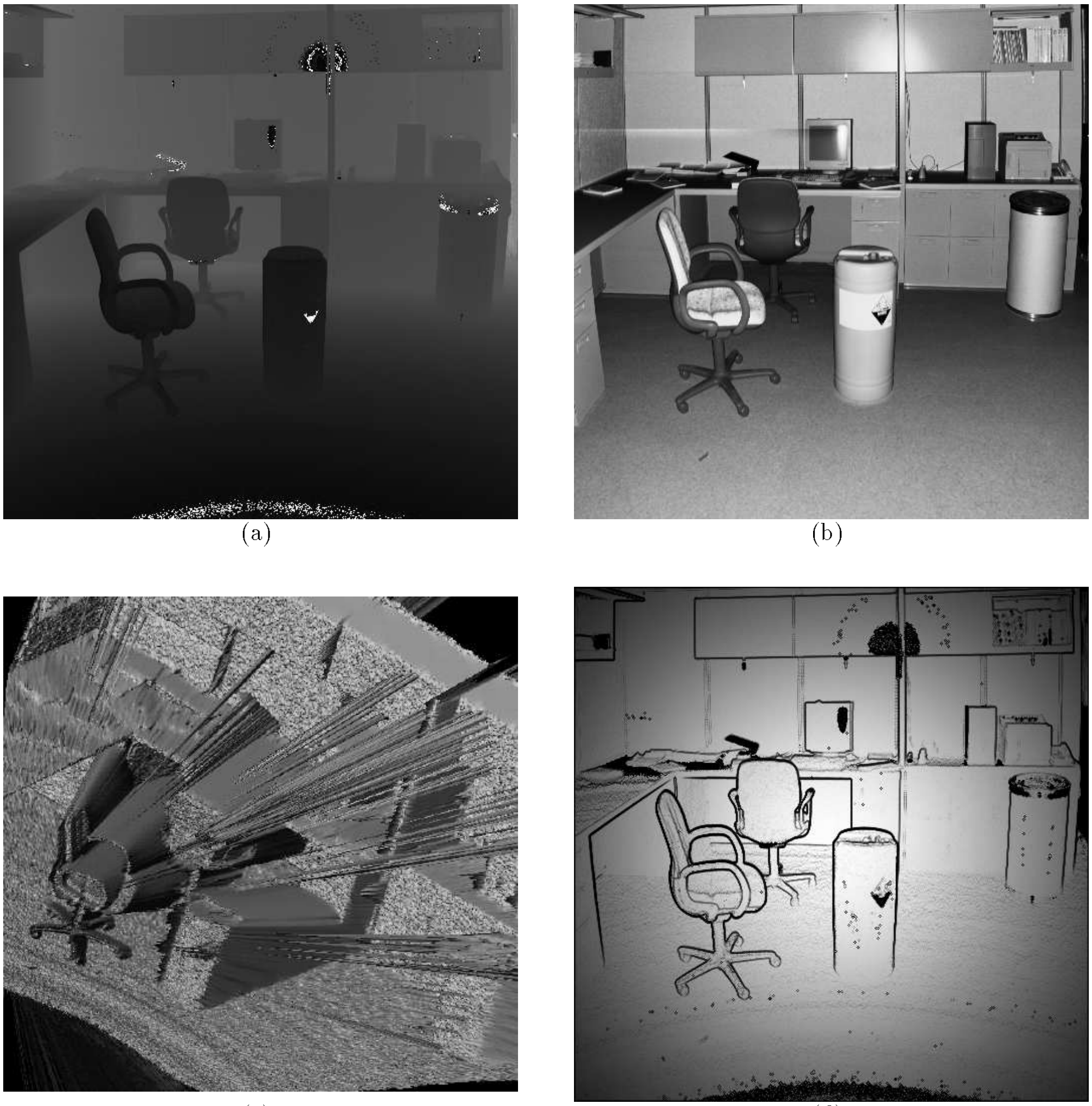
(a) (b)

Figure 7: Range maps: Synthetic range data 200×200 pixels with 20% Gaussian white noise of a torus end (a) and side (b).



(a) (b) (c)

Figure 8: (a) An analytically-defined model of a torus. (b) An initial model (80×80×40 voxels) is constructed by combining six points of view of a torus and solving for $g(\mathbf{x}) = 0$. (c) The model, which is attracted to the range data but subject to internal forces, evolves and settles into a smoother steady state.



(a) (b)

(c) (d)

Figure 9: (a) One of twelve range maps (b) The associated amplitude map (c) A surface plot of the range data to show the level of noise. (d) The confidence measures associated with those range values.

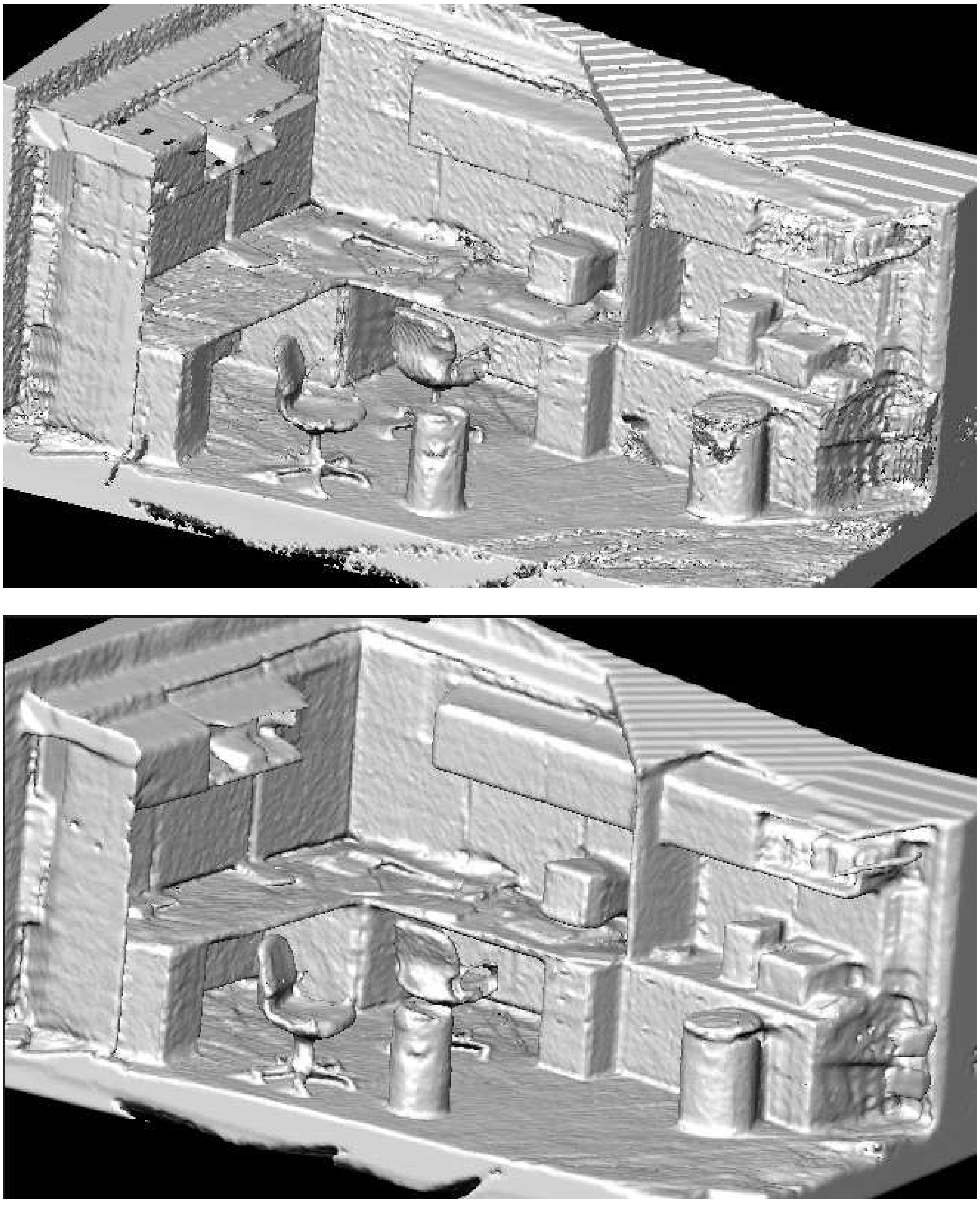


Figure 10: (top) The 3D reconstruction resulting from the zero crossings of $g(\mathbf{x})$ gives some averaging, but includes no prior. (bottom) The result of 32 iterations with the iterative scheme includes the prior and excludes influences of data on surfaces that face away from the scanner.

volume is $300 \times 150 \times 180$ voxels, and the resolution is 1.8 cm/voxel. These results show the ability of the statistically-based approach to overcome the noise in the scanner, and they show that the inclusion of iterative, model-fitting scheme helps create more accurate reconstructions. The resolution of the model falls below that of the scans, because it was limited by the random-access-memory available on our workstation. Some small features, such as the arm rests of the chairs, are lost because of the inaccuracies in the registration of the individual range maps.

6. CONCLUSIONS

Level-set models offer a powerful mechanism for deforming the surfaces of solid objects. The level-set formulation provides a set of rules that describe how to manipulate the values within a volume so that the level surfaces within that volume move in a way that minimizes various energy functionals. The rules for manipulating the level surfaces of volumes follow from a systematic reformulation of the mathematics that describe surface movements. This method borrows from many of the developments made in modeling surface deformations using parametric models, but it provides a number of useful advantages.

We have shown several applications of level-set model in 3D graphics. Each application follows from the same basic mathematical analysis, and in each application level-set models offer some advantages over previously proposed methods. For surface morphing, level-set models deal elegantly with changing topologies, and they can incorporate global coordinate transformations that are generated automatically or by user input. In filleting and blending, level-set models can generate blended surfaces with no assumptions about object shape or topology. Likewise, the results are independent of any particular parameterization; they are geometrically invariant. The blending is local and can be restricted to add material only to concave or hyperbolic regions. For 3D reconstruction, level-set models offer a mechanism for finding optimal surface reconstructions with viewpoint invariant smoothing as well as a broad flexibility in shape and topology.

7. REFERENCES

[1] Chadrajit Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *SIGGRAPH '95 Proceedings*, pages 109–118, August 1995.

[2] J.F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. on Graphics*, 1(3):235–256, March 1982.

[3] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 251–257, July 1991.

[4] David E. Breen, Sean Mauch, and Ross T. Whitaker. 3D scan conversion of CSG models into distance volumes. Submitted to 1998 Symposium on Volume Visualization.

[5] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In *Fifth International Conference on Computer Vision*, pages 694–699. IEEE, IEEE Computer Society Press, 1995.

[6] Y. Chen and G. Médioni. Fitting a surface to 3-D points using an inflating ballon model. In A. Kak and K. Ikeuchi, editors, *Second CAD-Based Vision Workshop*, volume 13, pages 266–273. IEEE, 1994.

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Computer Graphics (SIGGRAPH '96 Proceedings)*, July 1996.

[8] M. Desbrun and M.-P. Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH '95 Proceedings*, pages 287–290, August 1995.

[9] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for animation. In *Graphics Interface '98 Conference Proceedings*, June 1998.

[10] Phillip Getto and David Breen. An object-oriented architecture for a computer animation system. *The Visual Computer*, 6(2):79–92, March 1990.

[11] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *ECCV*. Springer-Verlag, 1996.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[13] J. Hughes. Scheduled fourier volume morphing. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):43–46, July 1992.

[14] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–323, 1987.

[15] J. Kent, W. Carlson, and R. Parent. Shape transformation for polyhedral objects. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):47–54, July 1992.

[16] A. Lerios, C. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *SIGGRAPH '95 Proceedings*, pages 449–456, August 1995.

[17] W.E. Lorenson and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1982.

[18] R. Malladi, J.A. Sethian, and B.C. Vemuri. Evolutionary fronts for topology-independent shape modeling and recovery. In J.-O. Eklundh, editor, *Third European Conference on Computer Vision*, pages 3–13. Springer-Verlag, 1994.

[19] Ravikanth Malladi, James A. Sethian, and

Baba C. Vemuri. Shape modeling with front propogation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[20] T. McInerney and D. Terzopoulos. A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4d image analysis. *Computerized Medical Imaging and Graphics*, 19(1):69–83, 1995.

[21] Alan Middleditch and Kenneth Sears. Blend surfaces for set theoretic volume modeling systems. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):161–170, July 1985.

[22] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, and M.J. Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):217–226, July 1991.

[23] S. Muraki. Volumetric shape description of range data using "blobby model". *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):227–235, July 1991.

[24] S. Osher and J. Sethian. Fronts propogating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[25] 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(3):C–179–C–184, September 1994.

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

[27] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

[28] D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

[29] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *SIGGRAPH '94 Proceedings*, pages 311–318, July 1994.

[30] Ross T. Whitaker. A level-set approach to 3D reconstruction from range data. to appear *International Journal of Computer Vision*.

[31] Ross T. Whitaker. Volumetric deformable models: Active blobs. In Richard A. Robb, editor, *Visualization In Biomedical Computing 1994*, pages 122–134, Mayo Clinic, Rochester, Minnesota, 1994. SPIE.

[32] Ross T. Whitaker. Algorithms for implicit deformable models. In *Fifth International Conference on Computer Vision*. IEEE, IEEE Computer Society Press, 1995.

[33] Ross T. Whitaker and David T. Chen. Embedded active surfaces for volume visualization. In *SPIE Medical Imaging 1994*, Newport Beach, California, 1994.