Algorithms for Implicit Deformable Models

Ross T. Whitaker
European Computer-Industry Research Centre GmbH
81925 Munich, Germany

Abstract

This paper presents a framework for implicit deformable models and a pair of new algorithms for solving the nonlinear partial differential equations that result from this framework. Implicit models offer a useful alternative to parametric models, particularly when dealing with the deformation of higher-dimensional objects. The basic expressions for the evolution of implicit models are relatively straightforward; they follow as a direct consequence of the chain rule for differentiation. More challenging, however, is the development of algorithms that are stable and efficient.

The first algorithm is a viscosity approximation which gives solutions over a dense set in the range, providing a means of calculating the solutions of embedded families of contours simultaneously. The second algorithm incorporates sparse solutions for a discrete set of contours. This sparse-field method requires a fraction of the computation compared to the first but offers solutions only for a finite number of contours. Results from 3d medical data as well as video images are shown.

1 Introduction

Paradigms that rely on curve and surface evolutions play an important role in computer vision. In particular, deformable models combine input data with internal forces or constraints. Such models typically employ a minimization process to find a compromise between the internal forces of the model and the shapes indicated by the input data. The minimization algorithms often take the form of iterative processes that implement a type of hill-climbing strategy. The literature indicates a proliferation of algorithms which use such models for a wide array of problems ranging from segmentation to tracking.

Parameterized models have several drawbacks which make them inadequate for certain kinds of applications. The dependency on parameterization is critical; it limits the kinds of shapes a model can take. Models typically do not deform far from their initial conditions without some reparameterization. Such reparameterizations are often inefficient and developed ad hoc according to heuristics. The parameterization often makes it difficult to measure the intrinsic geometry of the model, as with polygonal meshes.

The models presented here, called active blobs [1, 2], incorporate an implicit, rather than explicit, representation of shape. These models are characterized by level sets of dense scalar fields [3]. Parameterizations of such dense scalar fields are easy to manipulate. When these scalar fields are sampled on discrete, regular, rectilinear grids, the implicit models can be represented by greyscale digital images. This paper presents the basic framework for implicit models and the differential equations that result from this framework. Later sections describe several algorithms for computing solutions to these equations and show some results for planar curves and 3d surface models.

2 Parameterized deformable models

2.1 Curve codels

Kass, Witkin, and Terzopoulis [4] propose the active contours or "snakes" paradigm, a general framework for deformable models. These models are parameterized curves that incorporate a set of internal forces that drive models toward solutions that are rigid and smooth. This paradigm and its many variations have proven useful for certain kinds of 2d segmentation problems.

A parameterized curve that evolves over time is a function

$$\mathbf{C} : \underset{s}{U} \times \underset{t}{\mathrm{I\!R}^+} \mapsto \underset{x,y}{V} \tag{1}$$

where $U \subset \mathrm{I\!R}$ (typically $U = [0,1]$ or S^1, the unit circle), $s \in U$ is the parameterization along the length of the curve, x and y are positions in the plane, and $V \subset \mathrm{I\!R}^2$ is the image space (usually a rectangular patch). The parameter t indicates time, a convention that expresses a one parameter family of such curves.

The evolution equation for an active contour is the weighted sum of the influences from the various energy terms being minimized. In [4] they describe several different smoothing terms as well as an attraction to some input $f(x, y)$:

$$\frac{\partial \mathbf{C}}{\partial t} = \alpha \mathbf{C}_{ss} + \beta \mathbf{C}_{ssss} + \gamma \nabla f(\mathbf{C}), \tag{2}$$

where f might include feature maps from images as well as user input. This partial differential equation can be solved using any one of a number of numerical methods.

2.2 Surface models

The same ideas have been applied with some success to surface models [5, 6]. A surface is a two-parameter object in a three-dimensional space, i.e., a surface $\mathbf{S}$ is

$$\mathbf{S} : \underset{r}{U} \times \underset{s}{U} \mapsto \underset{x,y,z}{V} \tag{3}$$

where $V \subset \mathrm{I\!R}^3$, and $U \subset \mathrm{I\!R}$.

The image energy term, the one that forces the surface toward regions of interest, is essentially the same for surfaces as for curves, but the smoothing of surfaces is not a trivial generalization of the smoothing for curves. In previous work [1, 2], I have proposed an invariant, second-order flow that produces desirable behavior for smoothing surfaces in a wide range of examples. This work is beyond the scope of this paper, and for now I use the average second-order flow with respect to the two parameters in the domain, realizing that this flow has some severe limitations. The results in later sections rely on a weighted-curvature flow as described in [2]. The internal constraints of an active surface incorporate both free parameters, r and s:

$$\begin{aligned} \mathrm{d}E_{\mathrm{smooth}} &= \mathbf{S}_{ss} + \mathbf{S}_{rr} \quad \text{and} \qquad (4) \\ \mathrm{d}E_{\mathrm{image}} &= -\nabla f(\mathbf{S}), \qquad (5) \end{aligned}$$

where $\mathrm{d}E$ indicates that these terms are derived from the first variation of the corresponding energy terms.

A gradient descent algorithm yields an evolution equation for the surface:

$$\frac{\partial \mathbf{S}}{\partial t} = \alpha \mathrm{d}E_{\mathrm{smooth}} + \beta \mathrm{d}E_{\mathrm{image}}, \qquad (6)$$

where α and β are user-defined parameters that control the relative influence of the internal constraints and the input data.

Equations 2 and 4–6 describe the evolution of a model in terms of its parameterization. The precise behavior of the model depends on the particular parameterization. Often surface parameterizations are limited by topology. For instance, 3d parameterizations of a sphere are generally not compatible with a torus. In addition, as models evolve and undergo large deviations from their original shapes, surface parameterizations often introduce de facto constraints. For instance the expansion of polygonal models creates a kind of coarseness which prevents the model from capturing smaller structures; thus the evolution of polygonal models requires the creation and deletion of polygons [6].

The continuous mapping described by Eq. 3 represents only a limited class of surfaces; *global parameterizations of surfaces are very restrictive and are difficult to manipulate as the surface deforms.* A more general definition is that of a regular surface which is a subset of 3-space: $S \subset \mathbb{R}^3$, such that for each point $\mathbf{p} \in \mathbf{S}$ a neighborhood V of $\mathbf{p}$ there exists a map $\mathbf{x} : U \mapsto V \cap \mathbf{S}$, where $U \subset \mathbb{R}^2$ is an open set and $\mathbf{x}$ is a differentiable homeomorphism that is one-to-one on U. Thus, the surface $\mathbf{S}$ can be represented locally as a function, $\mathbf{x}(r, s) = x(r, s),\ y(r, s)\ z(r, s)$. Regular surfaces provide a means of looking only at the local structure of surfaces without a need for a global parameterization. In the following section a local parameterization is expressed in terms of the intrinsic geometry of the surface in order to do away with the parameters r and s entirely.

3 Embedding deformable models

There is an intrinsic geometry to such objects which depends only on their shape in the range, V, rather than a particular parameterization. Thus, the intrinsic geometry of a surface can be captured implicitly by embedding the surface as level sets of scalar functions. This strategy removes the parameterization from the model, leaving only the intrinsic geometry. Embedding active contours consists of four steps.

1 Express the equations of motion for a deformable model in terms of some unspecified parameterization (as was done in Sect. 2).

2 Describe the parameterization in terms of the differential structure of the model.

3 Assume the model is the level set of a function F (which is an image or volume for planar curves or surfaces respectively).

4 Express the geometry of the level set in terms of the differential structure of F, and create an evolution equation for F.

To apply this strategy to deformable manifold $\mathbf{M} \subset \mathbb{R}^n$, represent $\mathbf{M}$ as a level set of an n dimensional scalar function, $F : \mathbb{R}^n \times \mathbb{R}^+ \mapsto \mathbb{R}$, which evolves over time. The evolution equation of the individual level surfaces specifies a corresponding evolution equation for the scalar function $F(\mathbf{x}, t)$, which is a dense set of data in the domain of the model.

A subclass of all parameterized deformable models, $\mathbf{M}(\mathbf{s}, t)$ where $\mathbf{s} \in U \subset \mathbb{R}^{n-1}$, can be represented by

$$\mathbf{M}(t) = \{\mathbf{x} | F(\mathbf{x}, t) = k\}. \qquad (7)$$

The manifold $\mathbf{M}$ remains a level set of F over time, so the time derivative is zero:

$$0 = \frac{\partial F(\mathbf{M}, t)}{\partial t} + \nabla F(\mathbf{M}, t) \cdot \frac{\partial \mathbf{M}}{\partial t}, \qquad (8)$$

where

$$\nabla F(\mathbf{x}) \stackrel{\mathrm{def}}{=} \frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_n}. \qquad (9)$$

Thus,

$$\frac{\partial F}{\partial t} = -\nabla F \cdot \frac{\partial \mathbf{M}}{\partial t} = -|\nabla F| \frac{\partial \mathbf{M}}{\partial t} \cdot \mathbf{N}, \qquad (10)$$

where $\mathbf{N}$ is the normal to the level set.

For surfaces, $\mathbf{x} = x, y, z$, and the evolution of $\mathbf{S}(r, s)$ is expressed in terms of derivatives of F under the assumption of an orthonormal parameterization, i.e., $|\mathbf{S}_r| = |\mathbf{S}_s| = 1$ and $\mathbf{S}_r \cdot \mathbf{S}_s = 0$. The terms of the evolution equation can be written as differential expressions on F:

$$\mathbf{S}_{rr} \cdot \mathbf{N} = -\frac{F_{uu}}{|\nabla F|},\ \mathbf{S}_{ss} \cdot \mathbf{N} = -\frac{F_{vv}}{|\nabla F|}, \qquad (11)$$
$$\text{and } \nabla f \cdot \mathbf{N} = \nabla f \cdot \frac{\nabla F}{|\nabla F|},$$

where the subscripts in u and v represent directional derivatives with respect to the directions S_r and S_s respectively.

Multiplying by $|\nabla F|$ and adding the energy terms gives

$$\frac{\partial F}{\partial t} = \alpha \left(F_{uu} + F_{vv}\right) - \beta \left(\nabla F \cdot \nabla f\right). \qquad (12)$$

Note that $F_{uu} + F_{vv}$ is twice the mean curvature of the level set which is invariant to rotations (within the tangent plane) of the u-v coordinate system. This means that the particular choice of r and s does not affect the result of the evolution. Indeed, this invariance should be a requirement of any energy term used in such models.

The mean curvature (as well as the Gaussian curvature and bending energy [2]) can be computed directly from the first- and second-order structure of F, i.e.,

$$\frac{\partial F}{\partial t} = \alpha \left(F_{xx} + F_{yy} + F_{zz} - F_{ww}\right) - \beta \nabla F \cdot \nabla f, \quad (13)$$

where the subscript w indicates a derivative in the normal direction; e.g., $F_w = |\nabla F|$. The directional derivatives F_w and F_{ww} can be computed directly from first- and second-order finite difference schemes.

Equation 13 is (at any particular instant in time) equivalent to an active surface with an explicit parameterization r, s, in which r and s are given in units of arc length and have perpendicular tangents. Thus the evolution of the implicit surfaces in Eq. 13 is equivalent to a parametric model if one performs a reparameterization locally at each time step.

4 Numerical algorithms

The previous section describes the continuous mathematics for embedding active surfaces, but it does not address the practical problems of solving these equations in the discrete domain. In general the scalar function F is not a continuous mapping from the image space, $V \subset \mathbb{R}^3$, to the real numbers, but rather it has some discrete representation in a digital computer. There are many options for representing $F : V \mapsto \mathbb{R}$; for this work I represent F as a discretely-sampled image that is sampled on a finite, Cartesian grid. For curves, F is 2d digital image, and for surfaces F is a set of voxels that form a 3d volume. The strategy of embedding active contours does not depend on this particular discrete representation; I choose this representation because it provides a mechanism for performing numerical differentiation at multiple scales. Derivatives of F are measured using finite differences with the "tightest fitting" kernels of a particular order [7]. All derivatives are measured in the directions x, y (and z) that are associated with the discrete grid. Directional derivatives (in local coordinates) are calculated from these discrete Cartesian derivatives, using a dot product with a unit vector in the desired direction.

4.1 Viscosity solutions

The evolution equations for the embedded contours have the form of a Hamilton-Jacobi equation, i.e.,

$$\frac{\partial F(\mathbf{x}, t)}{\partial t} = G(\mathbf{x}, t) |\nabla F(\mathbf{x}, t)|, \qquad (14)$$

where $\mathbf{x} = x, y, z$. Discrete solutions using finite forward differences in time are generally not stable. A finite forward differences scheme on a discrete grid gives

$$F(\mathbf{x}, t + \Delta t) = F(\mathbf{x}, t) + \frac{\partial F(\mathbf{x}, t)}{\partial t} \Delta t. \qquad (15)$$

Such equations can overshoot [3] near sharp edges causing ringing and instability.

Stable solutions of Eq. 14 can be computed by using a *viscosity* approximation. Osher and Sethian [3] propose an up-wind scheme which incorporates piecewise continuous approximations to F and utilizes one-sided (or up-wind) derivatives in approximations to $|\nabla F|$. Instead of an up-wind scheme I use a second-order diffusion term for computing stable solutions to Eq. 14.

This viscosity solution can be shown to be stable by considering a Fourier analysis on a class of one-dimensional signals. I first show that the forward finite difference scheme is unstable and then use the same analysis to show that a second-order term controls this instability. Finally, this one-dimensional algorithm is generalized to higher dimensions by considering a Gauge coordinate representation of F (aligned with the level sets) and introducing a nonlinear diffusion which is a natural extension of the one dimensional case (which happens to be linear). The insight that drives this analysis is the fact that the flow lines of F can be treated as one-dimensional functions when solving Hamilton-Jacobi systems.

Consider a one-dimensional, monotonic function Φ; i.e., $\Phi : U \mapsto \mathbb{R}$, where $U \subset \mathbb{R}$ and $\Phi_x(x) > 0 \ \forall x \in U$. A discrete forward time difference gives:

$$\Phi(x, t + \Delta t) = \Phi + G \Phi_x \Delta t. \qquad (16)$$

Assume for the time being a constant $G(x) = \alpha$ where $-1 < \alpha < 1$. Then the Fourier transform associated with an incremental time step is,

$$\mathcal{F}[\Phi_{(t+\Delta t)}] = T(\omega) \mathcal{F}[\Phi_{(t)}], \qquad (17)$$

where $T(\omega)$ is the transfer function. For the forward time scheme given in Eq. 16 this transfer function is

$$\begin{aligned} T(\omega) &= 1 + \alpha \Delta t \left(e^{-\imath\omega} - e^{\imath\omega}\right) \qquad (18) \\ &= 1 - 2\alpha\imath \sin(\omega) \Delta t. \end{aligned}$$

The "overshooting" associated with the numerical scheme of Eq. 16 is understood by examining the absolute value of the transfer function, $|T(\omega)|^2 = 1 + (\alpha\omega \sin(\omega)\delta t)^2$, which is greater than one for all $\omega \neq \pi/2$. Thus, the forward difference scheme consistently increases certain frequencies over time creating unstable behavior.

The numerical scheme is made stable by the introduction of second-order diffusion term, i.e.,

$$\begin{aligned} &\Phi(x, t + \Delta t) = \qquad (19) \\ &\quad \Phi(x, t) + \alpha \left[|\Phi_x(x, t)| + \beta \Phi_{xx}(x, t)\right] \Delta t. \end{aligned}$$

In order to choose β consider the resulting transfer function,

$$T(\omega) = 1 - 2\left[\alpha\beta - \alpha\beta\cos(\omega) + \alpha\imath \sin(\omega)\right] \Delta t, \quad (20)$$

and note that on the complex plane this function is an ellipse which is offset from the origin by an amount $1 - 2\alpha\beta\Delta t$. This ellipse can be made to lie entirely inside the unit circle by setting

$$\beta = (1/2) \text{ Sign } (\alpha) = (1/2)(\alpha/|\alpha|) \text{ and } \Delta t < 1.$$

Putting this back into the numerical scheme gives

$$\Phi(x, t + \Delta t) = \Phi(x, t) + \left[\alpha|\Phi_x(x,t)| + \frac{1}{2}|\alpha|\Phi_{xx}(x,t)\right]\Delta t. \tag{21}$$

The result can be generalized from constant α to space-varying $G(x)$ by incorporating the assumption $|G(x)|\Delta t < 1$ and using the results from $\alpha = \pm 1$ to bound the results for $G(\boldsymbol{x})$. This gives the one dimensional numerical scheme

$$\Phi(x, t + \Delta t) = \Phi(x, t) + \left[G(x)|\Phi_x(x,t)| + \frac{1}{2}|G(x)|\Phi_{xx}(x,t)\right]\Delta t. \tag{22}$$

In one dimension, this formulation is identical (except at extrema, which must be handled as special cases) to the first-order up-wind scheme proposed by [3]. However the formulation I use here generalizes to higher dimensions in a rotationally invariant manner (to within the coarseness of the underlying grid), whereas the up-wind approaches typically assume that the velocity of a moving wave front is in one of the cardinal directions, i.e., aligned with the grid on which F is sampled.

Equation 23 generalizes to functions $F(\mathbf{x})$ with n-dimensional domains by considering local parameterizations of F along flow lines. Flow lines are the integral paths of the gradient, ∇F. These lines are perpendicular to the level sets of F which represent implicit curves, surfaces, or hypersurfaces (depending on n). These flow lines are the one dimensional sub-space to which the previous one-dimensional analysis applies. Thus, the multi-dimensional numerical scheme is:

$$F(\mathbf{x}, t + \Delta t) = F(\mathbf{x}, t) + \left[G(\mathbf{x})|\nabla F(\mathbf{x},t)| + \frac{1}{2}|G(\mathbf{x})|F_{ww}(\mathbf{x},t)\right]\Delta t, \tag{23}$$

where F_{ww} is the second derivative in the direction of the gradient of F. This value is computed from the first and second derivatives of F, i.e., $F_{ww} = (\nabla F)(\mathrm{D}^2 F)(\nabla F)$, and $\mathrm{D}^2 F$ is the matrix of second derivatives, or the Hessian, of F.

Solutions to Eq. 24 require some specification of the boundary conditions on the image space V. For the results in this paper I use $F_{\mathbf{NN}}(\mathbf{x}) = 0 \; \forall \mathbf{x} \in \partial V$, where ∂V is the boundary of the image space, and $\mathbf{N}$ is the direction normal to the boundary. These boundary conditions enable level sets to move across the boundaries unimpeded.

This second-order viscosity scheme requires time steps Δt that are inversely proportional to the velocity of the fastest moving wave front;

$$\Delta t \leq \frac{1}{\sup_{\mathbf{x} \in V}\{|G(\mathbf{x}, t)|\}}, \tag{24}$$

which is required in order to maintain the assumption that $|G(\mathbf{x}, t)|\Delta t < 1$. The mean and weighted curvature flows are stable when using forward differences in time and centralized differences in space. Therefore, the $|G(\mathbf{x}, t)|$ term used for the viscosity solution includes only the influence of the image energy $\mathrm{d}E_{\mathrm{image}}$; the second-order smoothing term is included without the viscosity approximation.

Figure 1: One slice from an MRI data set containing 22 "thick" slices.

4.2 Results of viscosity solutions

Figure 1 shows an MRI data set of a human head, which consists of 22 slices. The sampling in the z (transversal) direction is less dense than the x and y directions. The slices are thick in the z direction and so attempts to reconstruct the head with a linear interpolation (Fig. 2(a)) show the "wedding cake" effect. In this example the active blob models are used to construct models of both the head and the tumor. These models are produced with a 3 to 1 supersampling in the z direction and a 2 to 1 subsampling in the x and y directions. This sampling is used because it produces models that resemble more closely the proportions of the actual anatomy. Figure 2(b) shows the initial models that were used to do the segmentation of the head and tumor. Figure 2(c) shows the resulting models.

4.3 Sparse-field solutions

The viscosity solutions described in the previous sections give solutions for implicit models over the entire range of the models. These solutions describe the behavior of an embedded family of contours, effectively representing a contour density at each point. In this section I describe an alternative numerical algorithm 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.

When solving for only a single level set, $F(\mathbf{x}, t) = k$, the evolution of F is important only in the vicinity of that level set. The evolution of the implicit models is such that the level sets evolve independently (to within the error introduced by the discrete grid). Thus, one should perform calculations for the evolution of F only in some neighborhood of the set $\{\mathbf{x}|F(\mathbf{x}) = k\}$. The difficulty is keeping track of this neighborhood as the k-level sets of F move in the range.

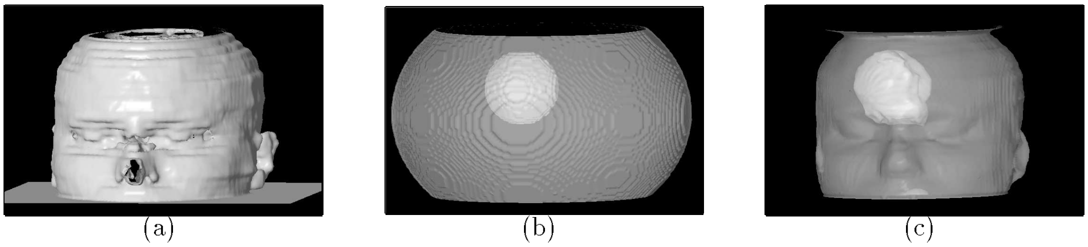

Figure 2: (a) An isosurface rendering of a 3 to 1 linear interpolation (to account for the thickness of the slices) from the data in Fig. 1, (b) the ellipsoidal models that were used as the initial conditions for segmenting the head and tumor, (c) the pair of models after 40 iterations, produced with identical parameters α and β but different initial conditions.

Malladi, Sethian, and Vemuri [8] construct an embedding of the evolving curve (via a signed distance transform) that has a finite width of m pixels. The evolution of the curve is calculated only on this set of pixels that are within the embedding. The algorithm presented here is even more sparse and does precisely the number of calculations needed to compute the next position of the level curve. It is very efficient and at each iteration visits only those pixels through which the k-level curve passes.

The sparse-field algorithm takes advantage of the fact that a k-level curve, $\mathbf{C}$, of a discrete image (of any dimension) has a set of pixels through which it passes (Fig. 3). I call this set of pixels the active set. The distance of the curve from the center of any active pixel is proportional (to within first order) to the intensity of that pixel and the gradient magnitude of F 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 their immediate neighbors are relevant to the evolution of this curve (at a particular point in time). The active set can be kept in a separate list (the active list) which gives their indices into the volume data. The list is constructed so that pixels can be added and removed from the active set in an efficient manner.

The evolution of the curve C is independent of the embedding F, so I choose a particular F outside of the active set that allows efficient computation. Assuming, without loss of generality, that k is zero; then F consists of values inside the active set, which are 1, values outside, which are -1, and the active set, which ranges between -1 and 1 (inclusive).

The sparse field algorithm involves visiting each active pixel once during each iteration and spreading the activity to other pixels (or "cells") as the level set moves from one pixel to another. The calculations for each active pixel at each time step involve the following algorithm.

1 Calculate the local geometry (first and second derivatives using centralized differences) for the current active pixel.

2 Compute the net change, based on the input data and model forces, to the value of F at that position.

3 Add this change to the pixel and decide if the new value $F^{(t+\Delta t)}$ falls outside the $[-1, 1]$ interval.

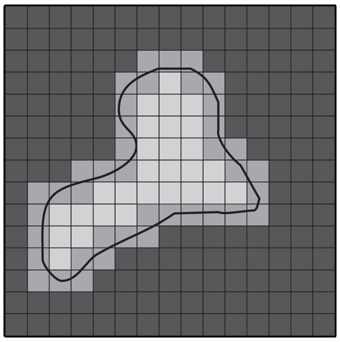

Figure 3: A level curve of 2d scalar field passed through a finite set of pixels. Only those pixels and their nearest neighbors are relevant to the evolution of that curve.

If $F^{(t+\Delta t)} \geq 1$ set it to 1, make it inactive, and set all neighboring, inactive, external pixels as active.

If $F^{(t+\Delta t)} \leq -1$ set it to -1, remove it from the active list, and set all neighboring, inactive, internal pixels as active.

Because solutions of F are constrained to lie in the $[-1, 1]$ interval, the overshooting associated with forward finite differences algorithms is not a problem, and there is no need for the second-order viscosity term described in the algorithm of Sect. 4.1.

Experiments in 2d (planar curve models) and 3d (surfaces of solid objects) show that the sparse field algorithm is stable and efficient. The active pixels can be kept in a linked list data structure so that at each time step *only the active pixels are visited*. This decrease in computation allows curve and surface evolutions to be computed in reasonable times (tens of iterations per second) on conventional workstations.

4.4 Results of sparse-field solutions

Figure 4(a) shows a greyscale 2d image of a model engine that must be segmented from the background. Figure 4(c) shows an initial model that consists of a hand-placed circle with a radius that appeared to be similar to the engine size. The circle is represented as an image with a coarse resolution; it is one quarter the size of the original image in Fig. 4(a). This model is allowed to deform (and the resolution increased as

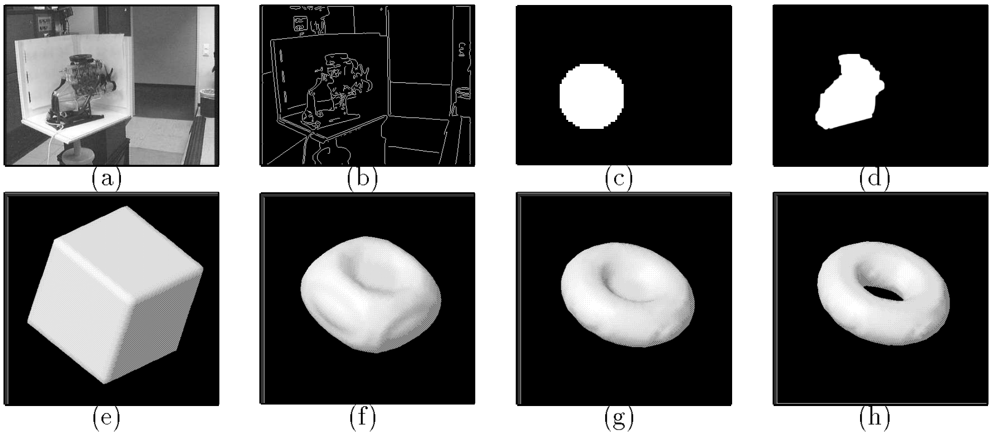

Figure 4: **Video image segmentation:** (a) An image of a model engine, (b) an initial, coarse-scale model, (c) a set of edges from the input image, (f) the steady-state model that results from moving downhill (with a smoothing term, through 3 distinct resolutions) on the distance transform of those edges. **Surface evolution**: A cube model moves downhill on the distance transform of a torus. From the top left: (e) 0 iterations, (f) 10 iterations, (g) 20 iterations, and (h) 30 iterations.

it settles) so that it moves downhill on the distance transform associated with edge maps computed from the original image. Figure 4(b) shows the edge map at the finest resolution and 4(d) shows the steady state of the model at the same resolution. While the model is capable of a reasonable figure/ground separation, it suffers from some of the problems typically associated with deformable models. In particular, it is attracted to local minima.

The sparse-field algorithm proved to be very efficient, with on the order of 200 active pixels in the course-scale calculations and 600 active pixels in the finer-scale calculations. The evolution for all three levels combined took several seconds on a SPARCstation 20, a shorter time than required for the edge calculations.

Figures 4(e)–(h) show how the sparse-field solutions can provide topological flexibility for surface models. The initial model is a cube (represented as a binary $32 \times 32 \times 32$ voxel image) which moves downhill on the distance transform of a torus. There were approximately 2500 active pixels and the entire evolution process took about a minute on a SPARCstation 20.

Acknowledgments

Thanks to the University of North Carolina Department of Computer Science for providing computing resources and data to assist this work. This work is supported by Bull SA, ICL Plc, and Siemens AG.

References

[1] R. T. Whitaker and D. T. Chen, "Embedded active surface for volume visualization," in *SPIE Medical Imaging 1994*, (Newport Beach, California), 1994.

[2] R. T. Whitaker, "Volumetric deformable models: Active blobs," in *Visualization In Biomedical Computing 1994* (R. A. Robb, ed.), (Mayo Clinic, Rochester, Minnesota), pp. 122–134, SPIE—The International Society for Optical Engineering, 1994.

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

[4] M. Kass, A. Witkin, and D. Terzopoulis, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, pp. 321–323, 1987.

[5] D. Terzopoulos and D. Metaxas, "Dynamic 3d models with local and global deformations: Deformable superquadrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, pp. 703–714, 1991.

[6] J. Miller, D. Breen, W. Lorensen, R. O'Bara, and M. Wozny, "Geometrically deformed models: A method for extracting closed geometric models from volume data," in *SIGGRAPH 1991*, pp. 217–226, ACM Press, 1991.

[7] T. Lindeberg, *Discrete Scale-space Theory and the Scale-space Primal Sketch.* PhD thesis, Royal Institute of Technology, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1991. TRITA-NA-P9108.

[8] R. Malladi, J. Sethian, and B. Vemuri, "Evolutionary fronts for topology-independent shape modeling and recovery," in *Third European Conference on Computer Vision* (J.-O. Eklundh, ed.), pp. 3–13, Springer-Verlag, 1994.