A Real-Time Algorithm for Medical Shape Recovery *

R. Malladi and J. A. Sethian
Lawrence Berkeley National Laboratory
and
Department of Mathematics
University of California, Berkeley, CA 94720

Abstract

In this paper, we present a shape recovery technique in 2D and 3D with specific applications in visualizing and measuring anatomical shapes from medical images. This algorithm models extremely corrugated structures like the brain, is topologically adaptable, is robust, and runs in $O(N \log N)$ time where N is the total number of points in the domain. Our two-stage technique is based on the level set shape recovery scheme introduced in [11, 12, 4] and the fast marching method in [19] for computing solutions to static Hamilton-Jacobi equations.

1 Introduction

In many medical applications such as cardiac boundary detection and tracking, tumor volume quantification etc., accurately extracting shapes in $2D$ and $3D$ from medical images becomes an important task. These shapes are implicitly present in noisy images and the idea is to construct their boundary descriptions. Visualization and further processing like volume computation is then possible. In this paper, we present a fast shape modeling technique with specific applications in medical image analysis.

Active contour models [7] and surface models [23] have been used by many researchers to segment objects from medical image data. These models are based on deforming a trial shape towards the boundary of the desired object. The deformation is achieved via solving Euler-Lagrange equations which attempt to minimize an energy functional. As an alternative, implicit surface evolution models have been introduced in Malladi *et al.* [12] and Caselles *et al.* [4]. In these models, the curve and surface models evolve under an implicit speed law containing two terms, one that attracts it to the object boundary and the other that is closely related to the regularity of the shape.

One of the challenges in shape recovery is to account for changes in topology as the shapes evolve. In [4, 12], the authors have modeled anatomical shapes as propagating fronts moving under a curvature dependent speed function [18]. They adopted the level set formulation of interface motion due to Osher and Sethian [16]. The central idea here is to represent a curve as the zero level set of a higher dimensional function; the motion of the curve is then embedded within the motion of the higher dimensional surface. As shown in [16], this approach offers several advantages. First, although the higher dimensional function remains a function, its zero level set can change topology, and form sharp corners. Second, a discrete grid can be used together with finite differences to devise a numerical scheme to approximate the solution. Third, intrinsic geometric quantities like normal and curvature of the curve can be easily extracted from the higher dimensional function. Finally, everything extends directly to moving surfaces in three dimensions. In the Lagrangian perspective, similar behaviour can be achieved by reparameterizing the curve once every few time steps and by monitoring the merge/split of various curves and surfaces based on some criteria; see [15]. However, the issue of time complexity still remains a concern in most existing segmentation techniques. In this paper, we address that issue by designing a segmentation algorithm that runs in real-time while retaining other advantages like topological adaptability and accuracy.

In order to isolate shapes from images, an artificial speed term has been synthesized and applied to the front which causes it to stop near object boundaries; see [4, 12] for details. In [5, 8], this work has been improved by adding an additional term to the governing equation. That work views the object detection problem as computation of curves of minimal (weighted) distance. The extra term is a projection of an attrac-

*Supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Dept. of Energy under Contract DE-AC03-76SD00098 and by the NSF ARPA under grant DMS-8919074. The authors can be reached at {malladi,sethian}@math.lbl.gov

tive force vector on the curve normal. Subsequently the level set based schemes for shape recovery have been extended to $3D$ in [6, 9] and geometric measurements from detected anatomical shapes were made in [13]. Finally, interested readers are referred to many closely related efforts [8, 20, 25, 21, 13].

One drawback of the level set methodology stems from computational expense. By embedding a curve as the zero-level set of a higher dimensional function, one has turned a two-dimensional problem into a three-dimensional one. Reducing the added computational expense without sacrificing the other advantages of level set schemes has been the focus of some recent work. Tube or narrow-band methods both in $2D$ and $3D$ have been developed and used in [1, 12, 9]. The main idea of the tube method is to modify the level set method so that it only affects points close to the region of interest, namely the cells where the front is located. Another way to reduce the complexity of level set method is adaptive mesh refinement. This is precisely what Milne [14] has done in his thesis. The basic idea here is to start with a relatively coarse grid and adaptively refine the grid based on proximity to the zero level set or at places with high curvature. In both these cases it is possible to reduce computational expense from $O(M^3)$ to $O(M^2)$ per time step in the case of a moving surface, where M is the number of points in each coordinate direction. Multi-scale implementation has also been considered as a possibility for fast solution of the level set equation [25].

In this paper, we solve the shape modeling problem by using the fast marching methods devised recently in [19] and extended to a wider class of Hamilton-Jacobi equations in [2]. The marching method is a numerical technique for solving the Eikonal equation, and results from combining upwind schemes for viscosity solutions of Hamiltor-Jacobi equations, narrow-band level set methods, and a *min*-heap data structure. It results in a time complexity of $O(N \log N)$, where N is the total number of points in the domain.

2 The Fast Marching Method

We now briefly discuss the fast marching algorithm, which we shall need for shape recovery. Let Γ be the initial position of a hypersurface and let F be its speed in the normal direction. The speed function F in general can depend on the surface geometry.

In the level set perspective, one views Γ as the zero level set of a higher dimensional function $\psi(x, y, z)$. Then, by chain rule, an evolution equation for the moving hypersurface can be produced [16], namely

$$\psi_t + F(x, y, z) \mid \nabla\psi \mid = 0, \qquad (1)$$

Consider the special case of a monotonically advancing surface, i.e. a surface moving with speed $F(x, y, z)$ that is always positive (or negative). Now, let $T(x, y, z)$ be the time at which the surface crosses a given point (x, y, z). The function $T(x, y, z)$ then satisfies the equation

$$\mid \nabla T \mid F = 1; \qquad (2)$$

this simply says that the gradient of arrival time is inversely proportional to the speed of the surface. Broadly speaking, there are two ways of approximating the position of the moving surface; iteration towards the solution by numerically approximating the derivatives in Eqn. 1 on a fixed Cartesian grid, or explicit construction of the solution function $T(x, y, z)$ from Eqn. 2. Our marching algorithm relies on the latter approach.

For the following discussion, we limit ourselves to a two-dimensional problem. Using the correct "entropy" satisfying [18, 16] (upwind) approximation to the gradient, we are therefore looking for a solution in the domain to the following equation:

$$[\max(D^{-x}_{i,j}T, 0)^2 + \min(D^{+x}_{i,j}T, 0)^2 + \max(D^{-y}_{i,j}T, 0)^2 + \min(D^{+y}_{i,j}T, 0)^2]^{1/2} = 1/F_{i,j}, \qquad (3)$$

where D^- and D^+ are backward and forward difference operators. Since the above equation is in essence a quadratic equation for the value at each grid point, we can iterate until convergence by solving the equation at each grid point and selecting the largest possible value as the solution. This is in accordance with obtaining the correct viscosity solution.

2.1 The Algorithm

The key to constructing the fast algorithm is the observation that the upwind difference structure of Eqn. 3 means that information propagates from smaller values of T to larger values. Hence, the algorithm rests on building a solution to Eqn. 3 outwards from the smallest T value. Motivated by the methods in [1, 12], the "building zone" is confined to a narrow band around the front. The idea is to sweep the front ahead in an upwind fashion by considering a set of points in the narrow band around the current front, and to march this narrow band forward. We explain this algorithmically:

To illustrate, imagine that one wants to solve the equation 2 on an M by M grid on the unit box $[0, 1] \times [0, 1]$ with right-hand-side $F_{i,j} > 0$; furthermore, we are given an initial set $T = 0$ along the top of the box.

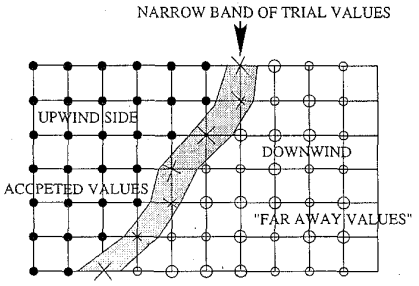

Figure 1: A snap shot of a $2D$ grid after several steps of the fast marching method.

1. **Initialize**

   (a) (Accepted Points): Let *Accepted* be the set of all grid points at which the value of T is fixed. In our example, $Accepted = \{(i,j) : i \in \{1,..,M\}, j = M\}$.

   (b) (Narrow Band Points): Let $NarrowBand$ be the set of all grid points in the narrow band. For our example $NarrowBand = \{(i,j) : i \in \{1,..,M\}, j = M-1\}$, also set $T_{i,M-1} = h/F_{ij}$, where h refers to the grid spacing.

   (c) (Far Away Points): Let $FarAway$ be the set of all the rest of the grid points $\{(i,j) : j < M-1\}$, set $T_{i,j} = \infty$ for all points in $FarAway$.

2. **Marching Forwards**

   (a) Begin Loop: Let $(i_{\min}, j_{\min})$ be the point in $NarrowBand$ with the smallest value for T.

   (b) Add the point $(i_{\min}, j_{\min})$ to *Accepted*; remove it from $NarrowBand$.

   (c) Tag as neighbors any points $(i_{\min}-1, j_{\min})$, $(i_{\min}+1, j_{\min})$, $(i_{\min}, j_{\min}-1)$, $(i_{\min}, j_{\min}+1)$ that are not *Accepted*; if the neighbor is in $FarAway$, remove it from that set and add it to the $NarrowBand$ set.

   (d) Recompute the values of T at all neighbors according to Equation (3), solving the quadratic equation given by our scheme.

   (e) Return to top of Loop.

Figure 1 shows a snap shot of the $2D$ grid after some steps of the above algorithm. For more general initial conditions, and for a proof that the above algorithm produces a viable solution, see [2].

2.2 The Min-Heap Data Structure

An efficient version of the above technique relies on a fast way of locating the grid point in the narrow band with the smallest value for T. We use a variation on the heap data structure (see Segdewick [17]) with back pointers to store the T values.

Specifically, we use a min-heap data structure. In an abstract sense, a min-heap is a "complete binary tree" with a property that the value at any given node is less than or equal to the values at its children. In practice, it is more efficient to represent a heap sequentially as an array by storing a node at location k and its children at locations $2k$ and $2k+1$. From this definition, the parent of a given node at k is located at $\lfloor \frac{k}{2} \rfloor$. Therefore, the root which contains the smallest element is stored at location $k = 1$ in the array. Finding the parent or children of a given element are simple array accesses which take $O(1)$ time.

We store the values of T together with the indices which give their location in the grid structure. Our marching algorithm works by first looking for the smallest element in the $NarrowBand$; this `FindSmallest` operation involves deleting the root and one sweep of `DownHeap` to ensure that the remaining elements satisfy the heap property. The algorithm proceeds by tagging the neighboring points that are not in *Accepted*. The $FarAway$ neighbors are added to the heap using an `Insert` operation and values at the remaining points are updated using Equation 3. `Insert` works by increasing the heap size by one and trickling the new element upward to its correct location using an `UpHeap` operation. Lastly, to ensure that the updated T values do not violate the heap property, we need to perform a `UpHeap` operation starting at that location and proceeding up the tree.

The `DownHeap` and `UpHeap` operations (in the worst case) carry an element all the way from root to bottom or vice versa. Therefore, this takes $O(\log M)$ time assuming there are M elements in the heap. It is important to note that the heap which is a complete binary tree is always guaranteed to remain balanced. This leaves us with the operation of searching for the $NarrowBand$ neighbors of the smallest element in the heap. We make this $O(1)$ in time by maintaining back pointers from the grid to the heap array. Without the back pointers, the above search takes $O(M)$ in the worst case.

3 Shape Recovery from Medical Images

Given a noisy image function $I(\mathbf{x})$, $\mathbf{x} \in \mathcal{R}^2$ for a $2D$ image and $\mathbf{x} \in \mathcal{R}^3$ in $3D$, the objective in shape modeling is to extract mathematical descriptions of certain anatomical shapes contained in it. We are interested in rapidly recovering boundary representation of these shapes with minimal user interaction. Real-time processing of medical images is then possible. In general, our approach consists of starting from user-defined "seed points" in the image domain and to grow

trial shape models from these points. These surfaces are made to propagate in the normal direction with a speed $F(\mathbf{x})$.

Shape recovery is possible if we synthesize a special image-based speed function which is defined as a decreasing function of image gradient $|\nabla I(\mathbf{x})|$. The image-based speed function, say k_I, controls the outward propagation of an initial surface (an interior point or a set of interior points) such that the shape model stops in the vicinity of shape boundaries. Mathematically this procedure corresponds to solving a static Hamilton-Jacobi equation (see Eqn. 1) which, when recast in the arrival time framework, is

$$| \nabla T(\mathbf{x}) | = \frac{1}{k_I}. \tag{4}$$

The speed function defined as

$$F(\mathbf{x}) = k_I(\mathbf{x}) = e^{-\alpha |\nabla I^{EP}(\mathbf{x})|}, \ \alpha > 0, \tag{5}$$

has values very close to zero near high image gradients, i.e., possible edges. False gradients due to noise can be avoided by applying an edge-preserving geometric smoothing scheme on the image (see [3, 10, 22]); in Eqn. 5, $I^{EP}(\mathbf{x})$ is the image processed using an edge-preserving smoothing scheme.

As an example, we consider the problem of reconstructing the entire cortical structure of the human brain from a dense MRI data set. The data is given as intensity values on a $256 \times 256 \times 124$ grid. We start by defining "seed" points in the domain. The value of T at these points is set to zero and the initial heap is constructed from their neighbors. The fast marching algorithm in $3D$ is then employed to march ahead to fill the grid with time values according to Eqn. 2. We visualize various stages of our reconstruction by rendering particular level surfaces of the final time function $T(x, y, z)$. These stages are shown in Fig. 2. On a 167 MHz Ultra Sparc machine, this simulation executes in 56 seconds.

Looking at the shapes in Fig. 2, we first note that they are very noisy. This is because we have not used any regularization or smoothing component in our speed function. Second, since the speed function falls to zero very rapidly, the surface could stop few grid points away from the "real" edge. In addition, usually there are variations in the gradient along the object boundary which can cause the shape to "overshoot". In large part, the definition of Eqn. 5 ensures that the speed F goes to zero rapidly and minimizes the over-shoot effect. However, to be further accurate, we now follow the ideas in [5, 12, 9] and outline how additional constraints can be imposed on the surface

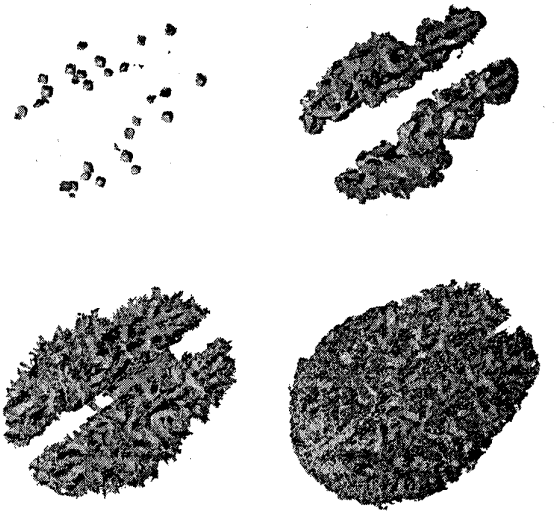

Figure 2: Evolutionary sequence showing the brain reconstruction. The surfaces shown are level surfaces at time values 0.01, 0.125, 0.25, and 0.75.

motion; see Fig. 5 for an accurate and regularized version of the brain shape.

3.1 Level set method

We are still interested in approximating the motion of a hypersurface but in order to do that we switch to the level set framework. First, note that the shape model is represented implicitly as a particular level set of a function $\psi(\mathbf{x})$ defined in the image domain. As shown in section 2, an evolution equation can be written for the function ψ such that it contains the motion of the surface embedded in it. Let the surface move under a simple speed law $F = 1 - \epsilon K$, where $K(\mathbf{x})$ is the curvature and $\epsilon > 0$. The constant component of F causes the model to seek object boundaries and the curvature component controls the regularity of the deforming shape. Geometric quantities like surface normal and curvature can be extracted from the higher dimensional function ψ; for example the mean curvature is given by

$$K = \nabla \cdot \left(\frac{\nabla \psi}{| \nabla \psi |} \right). \tag{6}$$

The driving force that molds the initial surface into desired anatomical shapes comes from two image-based terms. The first one is similar to Eqn. 5 and the second term attracts the surface towards the object boundaries; the latter term has a stabilizing effect especially when there is a large variation in the image

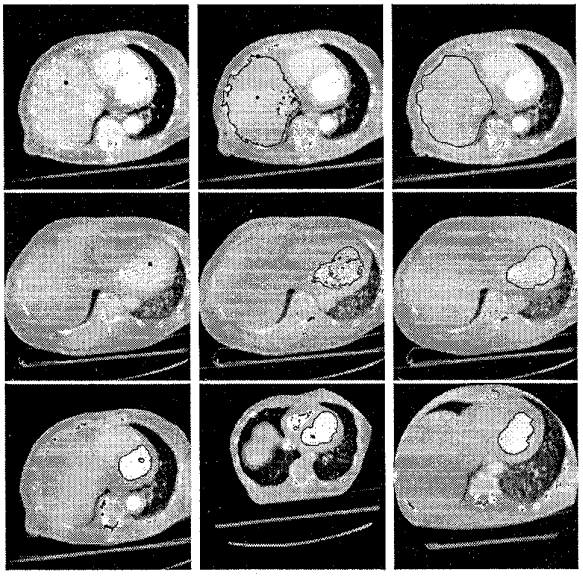

Figure 3: $2D$ examples of our two-stage shape recovery scheme; see text for description.

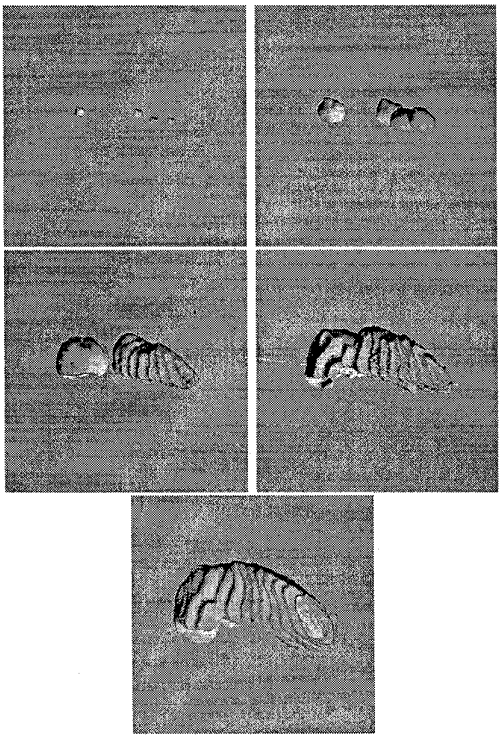

Figure 4: The two-stage shape recovery in $3D$: right figure in row 2 marks the end of marching or stage # 1 and the figure in row 3 depicts the final reconstruction after solving the level set shape recovery equation for a few steps.

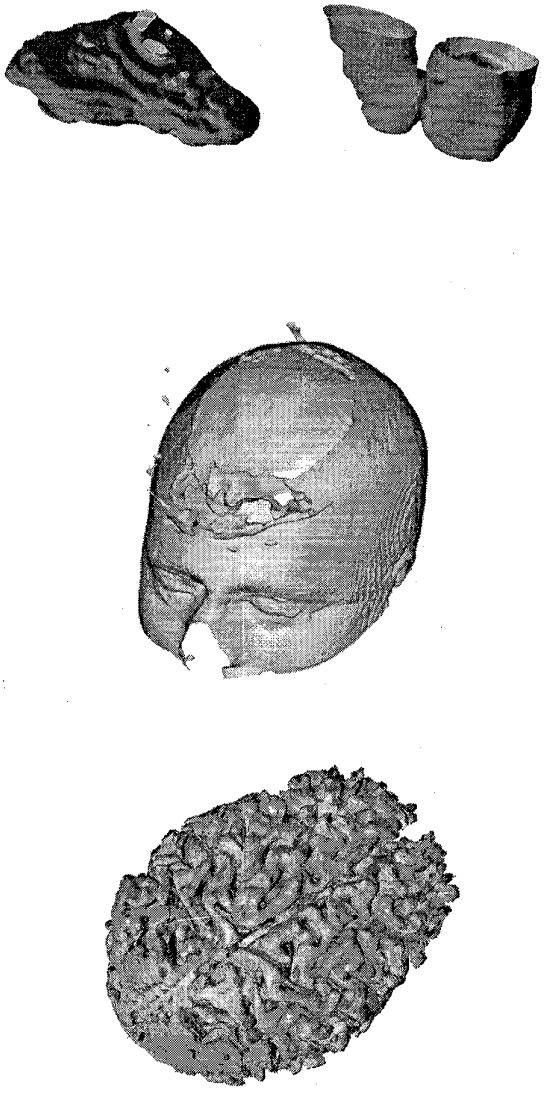

Figure 5: More examples of $3D$ shape recovery. First row shows the reconstructed shapes of the liver and heart chambers and the next two figures are the shapes of outer skin layer and the brain respectively that have been recovered from an MRI data set.

gradient value. Specifically, the equation of motion is

$$\psi_t + g_I(1 - \epsilon K)|\nabla\psi| - \beta\nabla P \cdot \nabla\psi = 0. \quad (7)$$

where,

$$g_I(\mathbf{x}) = \frac{1}{1 + |\nabla I^{EP}(\mathbf{x})|}. \quad (8)$$

Again, the function $I^{EP}(\mathbf{x})$ is the image smoothed using an edge-preserving, geometric smoothing scheme.

The second term $\nabla P \cdot \nabla\psi$ denotes the projection of an (attractive) force vector on the surface normal. This force which is realized as the gradient of a potential field (see [5])

$$P(\mathbf{x}) = -|\nabla I^{EP}(\mathbf{x})|, \quad (9)$$

attracts the surface to the edges in the image; coefficient β controls the strength of this attraction.

In this work, we adopt the following two stage approach when necessary. We first construct the arrival time function using our marching algorithm. This helps us to construct very quickly a very good initial guess on the final surface. Therefore, if a more accurate reconstruction is desired, we treat the final $T(\mathbf{x})$ function as an initial condition to our full model. In other words, we solve Eqn. 7 for a few time steps using explicit finite-differencing with the initial condition $\psi(\mathbf{x}; t = 0) = T(\mathbf{x})$. This too can be done very efficiently in the narrow band framework [12, 1]. Finally, the above initial condition is a valid one since the surface of interest is a particular level set of the final time function T.

4 Results

In this section, we present some shape recovery results from $2D$ and $3D$ medical images using the two-stage procedure we described in the previous section. We begin by defining seed points inside the region of interest; in most cases one mouse click will suffice. The value of $T(\mathbf{x})$ at these points is set to zero and the initial heap in order to start the marching method is constructed from their neighbors. We then employ the marching method to march until a fixed time or until the size of heap doesn't change very much between two successive time increments. This ends stage #1 of our scheme. We pass the final $T(\mathbf{x})$ function as the initial state to Eqn. 7 which is then solved for a few time steps. In $2D$, this whole procedure takes less than a second on a Sun Ultra Sparc workstation and to recover a $3D$ shape, the method executes in less than 20 seconds.

First, we present some $2D$ results in Fig. 4. The images in the first two rows depict reconstruction of liver and heart cross-sections from 256×256 images of the thoracic region; in the left column we show the original image along with user-defined seed point inside the the region of interest. The marching method is then run until $T(x, y) = 0.90$. The middle column depicts the level set $\{T = 0.75\}$. This marks the end of stage # 1. In stage #2, we use the T function as the initial state to our full method, namely Eqn. 7. The right column shows the final shapes – the level set $\{\psi = 0.75\}$ – that are obtained after solving Eqn. 7 for a few steps. Finally, in the third row, we show reconstructed shapes of left ventricle cross-sections from three other images.

In the next set of figures, we present examples in $3D$. Figure 4 shows the reconstruction of spleen from a $3D$ CT image of size $256 \times 256 \times 64$. We begin by initializing stage # 1 with a set of mouse clicks in the image domain; see the first two rows of Fig. 4. We then run the marching algorithm until time $T = 0.1$. As we did before in Fig. 2, we render various isosurfaces of the final time function $T(x, y, z)$ (T =0.01, 0.035, 0.07, and 0.1). The time function T is passed as an initial state to the level set shape recovery equation which is then solved for a few steps in a narrow-band around the level surface $\{\psi = 0.1\}$. The result is shown in the third row of Fig. 4. The level surface $\{T = 0.1\}$ that marks the end of stage # 1, is noisy and is stopped a little further away from the object boundary compared to the final reconstruction. This is because the speed function in Eqn. 5 falls to zero rapidly. To check the fidelity of the surface, we slice it parallel to the xy plane and superimpose the resulting contour on the corresponding image slice; see Fig. 6. Finally, in Fig. 5 we show two views of reconstructed shapes of liver, heart chambers from $256 \times 256 \times 64$ medical images, outer skin surface and the brain from a $256 \times 256 \times 128$ MRI image. The brain structure shown here is the regularized version of the one shown in Fig. 2; in other words, we solved Eqn. 7 for a few time steps with the final T function from the marching algorithm as the initial state.

References

[1] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *J. Comp. Phys.*, Vol. 118(2), pp. 269–277, May 1995.

[2] D. Adalsteinsson, R. Kimmel, R. Malladi, and J. A. Sethian, " Fast marching methods for computing solutions to static Hamilton-Jacobi equations," submitted for publication, *SIAM Journal of Numerical Analysis*, January 1996.

[3] L. Alvarez, P. L. Lions, and J. M. Morel, "Image selective smoothing and edge detection by nonlinear diffusion. II," *SIAM Journal on Numerical Analysis*, Vol. 29(3), pp. 845–866, 1992.

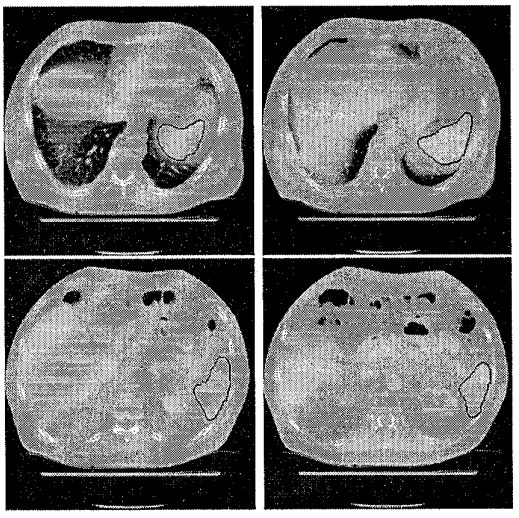

Figure 6: Various slices of a CT image of the thoracic region and superimposed cross-section of the reconstructed spleen surface; slice 10, 20, 40, and 50 are shown.

[4] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numerische Mathematik*, Vol. 66(1), pp. 1–32, 1993.

[5] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic snakes," *Proc. of ICCV*, MIT, Cambridge MA, June 1995.

[6] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, "Three-dimensional object modeling via minimal surfaces," *Proc. ECCV*, pp. 97–106, Cambridge, UK, April 1996.

[7] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, Vol. 1, pp. 321–331, 1988.

[8] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contour models," *Proc. of ICCV*, MIT, Cambridge MA, June 1995.

[9] R. Malladi and J. A. Sethian, "Level set methods for curvature flow, image enhancement, and shape recovery in medical images," *Proc. of International Conference on Mathematics and Visualization*, H. C. Hege, K. Polthier (eds), pp. 255–267, Springer-Verlag, Berlin, Summer 1996.

[10] R. Malladi and J. A. Sethian, "Image processing: Flows under Min/Max curvature and mean curvature," *Graphics Models and Image Processing*, Vol. 58(2), pp. 127–141, March 1996.

[11] R. Malladi, J. A. Sethian, and B. C. Vemuri, "A topology-independent shape modeling scheme," in *Proc. of SPIE Conference on Geometric Methods in Computer Vision II*, Vol. 2031, San Diego, California, pp. 246–258, July 1993.

[12] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: A level set approach," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, pp. 158–175, Feb. 1995.

[13] R. Malladi, R. Kimmel, D. Adelsteinsson, G. Sapiro, V. Caselles, and J. A. Sethian, "A geometric approach to segmentation and analysis of $3D$ medical images," *Proc. of IEEE/SIAM workshop on Mathematical Methods in Biomedical Image Analysis*, pp. 244–252, San Francisco, CA, June 1996.

[14] R. B. Milne, "An adaptive level set method," *Ph. D. Thesis*, Report LBNL-39216, Lawrence Berkeley National Laboratory, University of California, December 1995.

[15] T. McInerney and D. Terzopoulos, "Medical image segmentation using topologically adaptable surfaces," in *Proc. First Joint Conference of CVRMed-MRCAS*, LNCS Vol. 1205, pp. 23–32, March 1997.

[16] S. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation," *Journal of Computational Physics*, Vol. 79, pp. 12-49, 1988.

[17] R. Sedgewick, *Algorithms*, Addison-Wesley Publ., 1988.

[18] J. A. Sethian, "Curvature and the evolution of fronts," *Commun. in Mathematical Physics*, Vol. 101, pp. 487–499, 1985.

[19] J. A. Sethian, "A marching level set method for monotonically advancing fronts," *Proc. Natl. Acad. Sci., USA*, Vol. 93(4), 1996.

[20] J. Shah, "Recovery of shapes by evolution of zero-crossings," Tech. Report, Math. Dept, Northwestern University, Boston, MA, 1995.

[21] J. Shah, "A common framework for curve evolution, segmentation, and anisotropic diffusion," in *Proc. IEEE CVPR'96*, pp. 136–142, 1996.

[22] N. Sochen, R. Kimmel, and R. Malladi, "A general framework for low level vision," to appear in *IEEE Trans. on Image Processing*, special issue on PDEs and geometry-driven diffusion in image processing and analysis, 1998.

[23] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3D shape and nonrigid motions," *Artificial Intelligence*, Vol. 36, pp. 91–123, 1988.

[24] J. Weickert, "Anisotropic diffusion in image processing," Ph.D. Thesis, Kaiserslautern University, Kaiserslautern, Germany, Nov. 1995.

[25] R. T. Whitaker, "Algorithms for implicit deformable models," *Proc. ICCV*, pp. 822–827, Cambridge, MA, June 1995.