An $O(N \log N)$ algorithm for shape modeling*†

R. Malladi and J. A. Sethian

Lawrence Berkeley National Laboratory
and
Department of Mathematics
University of California, Berkeley, CA 94720

Abstract

We present a shape recovery technique in $2D$ and $3D$ with specific applications in modeling anatomical shapes from medical images. This algorithm models extremely corrugated structures like the brain, is topologically adaptable, and runs in $O(N \log N)$ time where N is the total number of points in the domain. Our technique is based on the level set shape recovery scheme introduced in [1, 2] and the fast marching method in [3] for computing solutions to static Hamilton-Jacobi equations.

Index Terms: Hamilton-Jacobi Equation, Eikonal Equation, Shape Recovery, Shape Modeling, Medical Image Analysis, Level Sets

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.

*Correction citation: Malladi, R., and Sethian, J.A., Proceedings of the National Academy of Sciences, Vol. 93, pp. 9389-9392, September 1996.

†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.

One of the challenges in shape recovery is to account for changes in topology as the shapes evolve. Malladi, Sethian, and Vemuri [1, 2] have done this by modeling anatomical shapes as propagating fronts moving under a curvature dependent speed function [4]. They adopted the level set formulation of interface motion due to Osher and Sethian [5]. 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 [5], 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 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 [1, 2] for details. Subsequently this work has been extended to $3D$ in [6]; see [7, 8, 9] for related efforts. In this paper, we solve the shape modeling problem by using the fast marching methods devised recently by Sethian [3, 10] and extended to a wide class of Hamilton-Jacobi equations in [11]. The marching method is a numerical technique for solving the Eikonal equation, and results from combining upwind schemes for viscosity solutions of Hamilton-Jacobi equations, narrow-band level set methods, and a *min*-heap data structure.

2 The Fast Marching Method

We now briefly discuss the marching algorithm introduced in [3]. Let Γ be the initial position of a hypersurface and let F be its speed in the normal direction. 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

[5], namely

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

Consider the special case of a surface moving with speed $F(x,y,z) > 0$. 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

$$\mid \nabla T \mid F = 1; \tag{2}$$

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 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 [4, 5] approximation to the gradient, we are therefore looking for a solution in the domain to the following equation:

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

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, our algorithm rests on building a solution to Eqn. 3 outwards from the smallest T value. Motivated by the methods in [12, 2], we confine the "building zone" 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, we imagine that we want to solve the Eikonal equation 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.

1. Initialize

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

   (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 $T_{i,M-1} = dy/F_{ij}$.

   (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 $Alive$; 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 $Alive$; 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.

For more general initial conditions, and for a proof that the above algorithm produces a viable solution, see [11].

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 a heap algorithm (see Segdewick [13]) 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 $Alive$. 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 u 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 $3D$ Shape Modeling

Given a noisy image function $I(x, y, z)$, the objective in shape modeling is to extract mathematical descriptions of certain anatomical shapes contained in it. This can be done by applying a special image-based speed function $k_I > 0$ which controls the outward propagation of an initial surface (an interior point in this case) such that the interface 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

$$\mid \nabla T \mid = \frac{1}{k_I}. \tag{4}$$

The speed function defined as

$$F(x, y, z) = k_I(x, y, z) = e^{-\alpha |\nabla G_\sigma * I(x,y,z)|}, \alpha > 0, \tag{5}$$

has values very close to zero near high image gradients, i.e., possible edges. False gradients due to noise can be avoided by applying a Gaussian smoothing filter or more sophisticated edge-preserving smoothing schemes (see [14, 15, 16, 6]). With this model, the surface does not stop at the shape boundary unless the speed values are very close to zero. The definition in Eqn. 5 ensures that the speed F goes to zero rapidly and minimizes the "over shoot" effect. To be further accurate, we now follow the ideas in [1, 2, 6] and outline how additional constraints can be imposed on the surface motion.

First, note that the shape model is represented implicitly as the zero level set of a function $\psi(x, y, z)$ defined in the image domain. Let the surface move under a simple speed law $F = 1 - \epsilon H$, $\epsilon > 0$, where H is the mean curvature given by the following expression:

$$H = \frac{\psi_{xx}(\psi_y^2 + \psi_z^2) + \psi_{yy}(\psi_x^2 + \psi_z^2) + \psi_{zz}(\psi_x^2 + \psi_y^2) - 2\psi_{xy}\psi_x\psi_y - 2\psi_{yz}\psi_y\psi_z - 2\psi_{zx}\psi_z\psi_x}{(\psi_x^2 + \psi_y^2 + \psi_z^2)^{3/2}}. \tag{6}$$

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 gradient value. Specifically, the equation of motion is

$$\psi_t + k_I(1 - \epsilon H)|\nabla\psi| - \beta\nabla P \cdot \nabla\psi = 0. \tag{7}$$

where,

$$k_I = \frac{1}{1 + |\ \nabla G_\sigma * I(x, y, z)\ |}. \tag{8}$$

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

$$P(x, y, z) = -|\nabla G_\sigma * I(x, y, z)|, \tag{9}$$

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. If a more accurate reconstruction is desired, we treat the final $T(x, y, z)$ 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 $\psi(x, y, z; t = 0) = T(x, y, z)$. This too can be done very efficiently in the narrow band framework. 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 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 Figure 1. The corrugated structure shown in Figure 1(f) is our final shape. In the Figures 2(a)-(d), we render the same $\{T(x, y, z) = 0.75\}$ surface from different perspectives. Finally, in Figure 2(e), we slice the surface parallel to the xy plane and superimpose the resulting contours on the corresponding image (see Figure 2(f)). The timings for various stages of calculation on a Sun SPARC 1000 machine are shown in Figure 3.

References

[1] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Evolutionary fronts for topology-independent shape modeling and recovery," in *Proceedings of Third European Conference on Computer Vision*, LNCS Vol. 800, pp. 3–13, Stockholm, Sweden, May 1994.

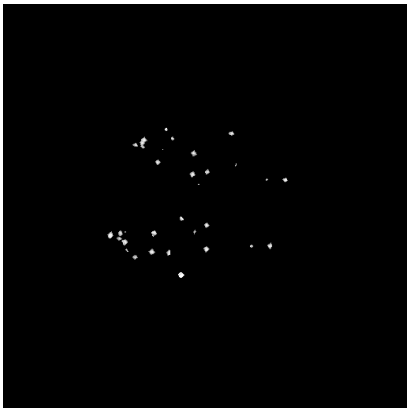
[2] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: A level set approach," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, pp. 158–175, Feb. 1995.

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

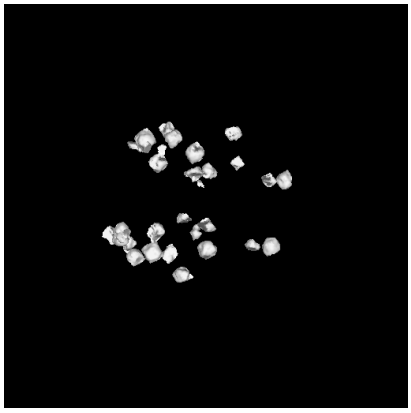
[4] J. A. Sethian, "Curvature and the evolution of fronts," *Commun. in Mathematical Physics*, Vol. 101, pp. 487–499, 1985.

[5] 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.

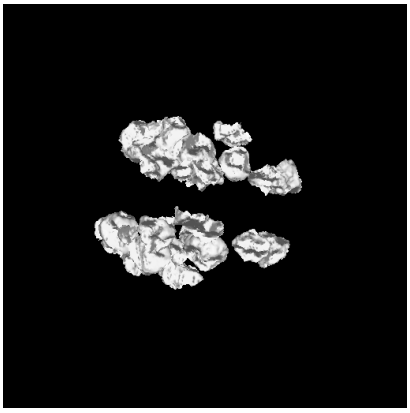
[6] R. Malladi and J. A. Sethian, "Level set methods for curvature flow, image enhancement, and shape recovery in medical images," *Proc. of International Conference on*



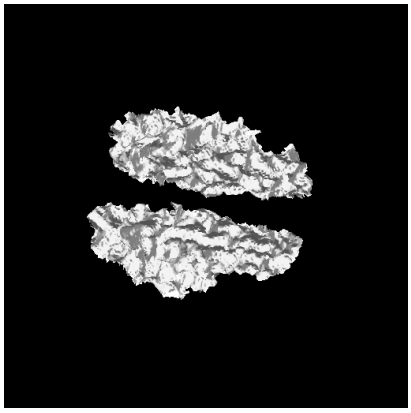
(a) $\{T(x, y, z) = 0.01\}$



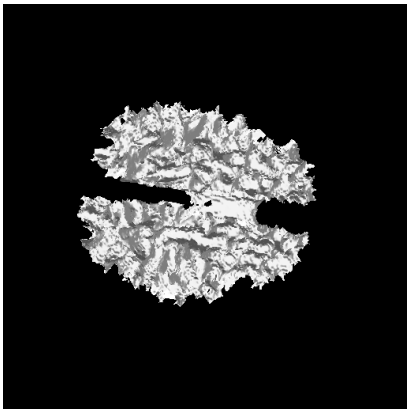
(b) $\{T(x, y, z) = 0.035\}$



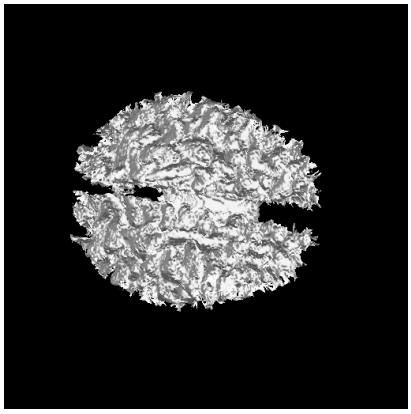
(c) $\{T(x, y, z) = 0.075\}$



(d) $\{T(x, y, z) = 0.125\}$

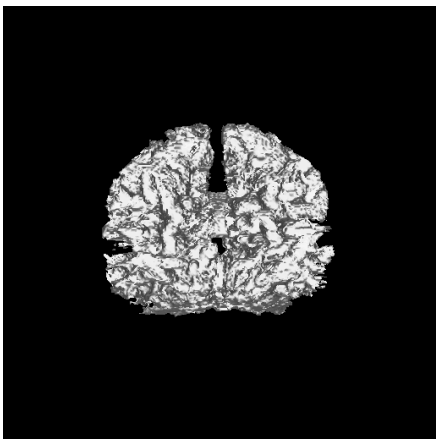


(e) $\{T(x, y, z) = 0.25\}$

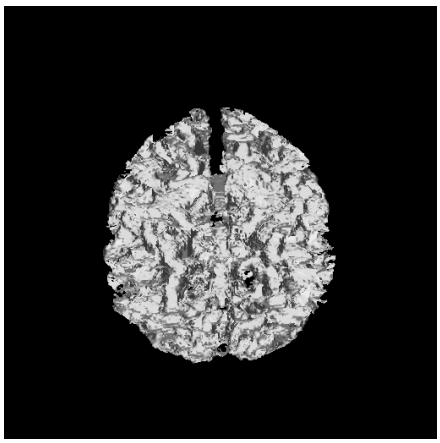


(f) $\{T(x, y, z) = 0.75\}$

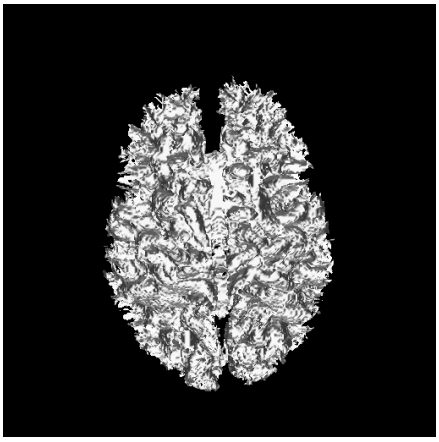
Figure 1: Evolutionary sequence showing the brain reconstruction.



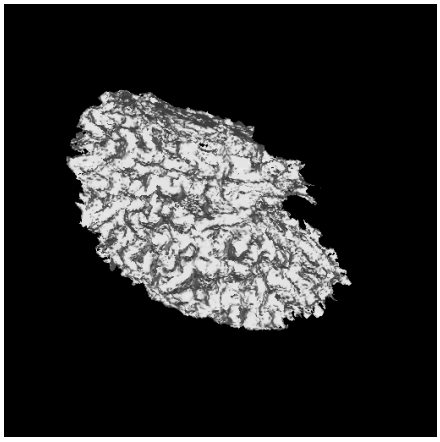
(a) View # 1



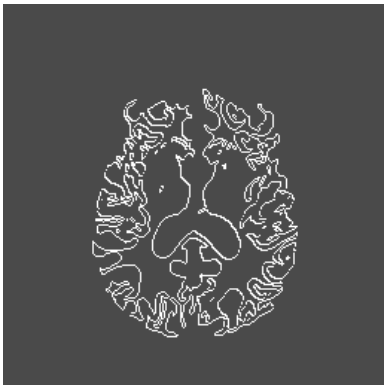
(b) View # 2



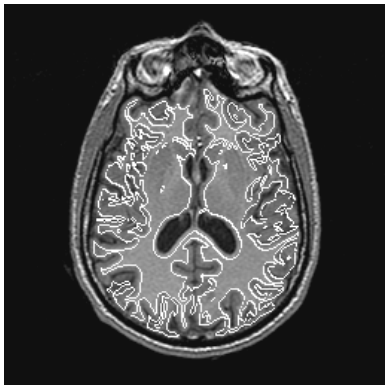
(c) View # 3



(d) View # 4



(e) Slice



(f) Superposition

Figure 2: Cortical structure rendered from different perspectives; part (e) & (f) depict a slice of the surface on the corresponding image.

Grid Size	Time to Load Speed File	Time to Propagate Surface	Total Time
$256 \times 256 \times 124$	8.61 secs	74.92 secs	83.53 secs

Figure 3: Timing for Marching to T=0.75: Sun SPARC 1000

Mathematics and Visualization, Ed. K. Polthier, Berlin, June 1995, Springer-Verlag, in press.

[7] 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.

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

[9] 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.

[10] J. A. Sethian, "A review of the theory, algorithms, and applications of level set methods for propagating interfaces," in press, *Acta Numerica*, 1995.

[11] 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.

[12] 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.

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

[14] R. Malladi and J. A. Sethian, "Image processing via level set curvature flow," *Proc. Natl. Acad. Sci., USA*, Vol. 92, pp. 7046–7050, July 1995.

[15] 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.

[16] R. Malladi and J. A. Sethian, "A unified approach to noise removal, image enhancement, and shape recovery," *IEEE Transactions on Image Processing*, Nov. 1996, in press.