Fast approximate surface evolution in arbitrary dimension

James Malcolm[1] Yogesh Rathi[2] Anthony Yezzi[1] Allen Tannenbaum[1]

[1]Georgia Institute of Technology, Atlanta, GA
[2]Brigham and Women's Hospital, Boston, MA

ABSTRACT

The level set method is a popular technique used in medical image segmentation; however, the numerics involved make its use cumbersome. This paper proposes an approximate level set scheme that removes much of the computational burden while maintaining accuracy.

Abandoning a floating point representation for the signed distance function, we use integral values to represent the signed distance function. For the cases of 2D and 3D, we detail rules governing the evolution and maintenance of these three regions. Arbitrary energies can be implemented in the framework.

This scheme has several desirable properties: computations are only performed along the zero level set; the approximate distance function requires only a few simple integer comparisons for maintenance; smoothness regularization involves only a few integer calculations and may be handled apart from the energy itself; the zero level set is represented exactly removing the need for interpolation off the interface; and evolutions proceed on the order of milliseconds per iteration on conventional uniprocessor workstations.

To highlight its accuracy, flexibility and speed, we demonstrate the technique on intensity-based segmentations under various statistical metrics. Results for 3D imagery show the technique is fast even for image volumes.

Keywords: Image segmentation, level set methods, fast numerical methods

1. INTRODUCTION

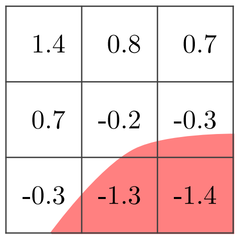
The level set method is a popular technique used in image processing applications. Casting the problem in an energy-based formulation, a local solution is found via gradient descent. However, the numerical considerations in this approach add a significant burden to stable and accurate implementations. For example, care must be taken in the choice of differencing scheme, maintaining the signed distance function, and interpolating values off the underlying grid. Such considerations introduce significant computational overhead into the technique.

Several techniques have been proposed to address these drawbacks. Since only the zero level set is of interest, significant performance improvement was initially achieved by reducing the domain of computation to the area immediately surrounding this interface. However, the same numerical considerations must still be applied to this reduced domain.[1, 2] Alternatively, changing the underlying distance function representation from floating point to binary removes the need for maintaining the distance function. Solutions in this case may contain inaccuracies and computation must be performed on the entire domain.[3] The closest method to the technique presented here maintains a lists of points just inside and just outside the interface and moves points between lists according to the sign of the force calculations; however, this necessitates interpolation of the interface and computation of the force twice along the interface.[4]

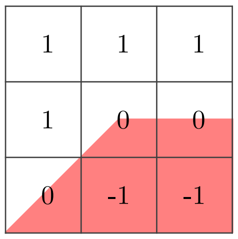
This paper introduces an approximate level set scheme that removes much of the computational burden while maintaining accuracy of the solution. Abandoning a floating point representation for the signed distance function, we use the integral values {-1, 0, 1} to represent the interior, zero level set, and exterior respectively. For the cases of 2D and 3D, we detail rules governing the evolution and maintenance of these three regions. Also, we show that arbitrary energies can be implemented with the definition of three operations: initialize iteration, move points in, move points out.

This scheme has several nice properties. First, computations are only performed at the zero level set instead of on the entire domain or a smaller narrow band. Second, the approximate distance function representation

Corresponding author: James Malcolm (`malcolm@gatech.edu`, `www.ece.gatech.edu/∼malcolm`)



(a) Floating point



(b) Discrete

Figure 1. An example of an implicit surface on a 3x3 grid using both floating point and approximate discrete representations. Interior regions are shaded. The floating point representation resolves to sub-pixel accuracy while the discrete version approximates the interface as crossing the center of each pixel.

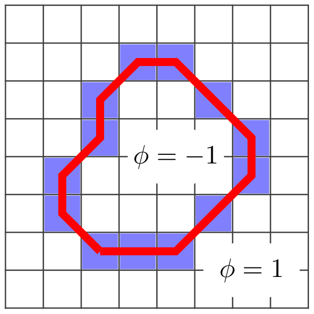


Figure 2. Discrete approximation of 2D signed distance function. Inside ($\phi = -1$) and outside ($\phi = 1$) regions are separated by interface points ($\phi = 0$, *blue*). Implicit curve is indicated *(red)*.

requires only a few simple integer comparisons for maintenance. Third, smoothness regularization involves only a few integer calculations and may be handled apart from the energy itself. Fourth, the zero level set is represented exactly thus removing the need for interpolation off the interface. Lastly, evolution proceeds on the order of milliseconds per iteration using conventional uniprocessor workstations.

2. PROPOSED ALGORITHM

This section describes the discrete algorithm for evolution and extensions to incorporate smoothing and improve speed. Each iteration of evolution involves the core routines for propagation and cleanup and the user-provided callback routines for force computation and bookkeeping as points cross the interface. Pseudo code listings are provided: the core loop in Procedure 1, propagation in Procedure 2, and cleanup in Procedure 3. We now give a brief discussion of the overall algorithm before covering those core and user-provided routines. For illustration, we first treat 2D curve evolution and then describe how the process generalizes to surfaces 3D. Section 3 provides definitions of user-provided routines for various energies.

To approximate the signed distance function ϕ, this paper uses a uniform grid with integer values {-1, 0, 1} to represent the interior, zero interface, and exterior regions, respectively. Additionally, we assume the curve or surface to be closed, the points of which are maintained in a set. Note that we use the terms "points" and "pixels" interchangeably. Figure 2 illustrates such a setup for a 2D grid.

2.1 Mechanics of evolution

The core algorithm is listed in Procedure 1. Each iteration of evolution proceeds in two phases: contraction and dilation. This two phased evolution is used to address a problem that develops upon convergence due to the approximation of the interface crossing at the pixel center. Suppose the curve is to truly settle slightly off the center of a particular grid pixel, hence the force will be slightly nonzero to correct for the approximated midpoint crossing. However, due to checking only the sign of the speed when propagating (see Procedure 2), the curve is still pushed a full pixel off. In the next iteration, the force compensates sending the pixel back one full pixel and the process repeats. Depending on the arrival of the interface along that line of convergence, this can lead

Procedure 1 Main algorithm

```
for each iteration do
    {Contraction}
    Callback: initialize iteration
    Restrict to contraction (only allow positive forces)
    Propagate (Procedure 2)
    Cleanup (Procedure 3)
    Callback: move points in and out

    {Dilation}
    Callback: initialize iteration
    Restrict to contraction (only allow negative forces)
    Propagate (Procedure 2)
    Cleanup (Procedure 3)
    Callback: move points in and out
end for
```

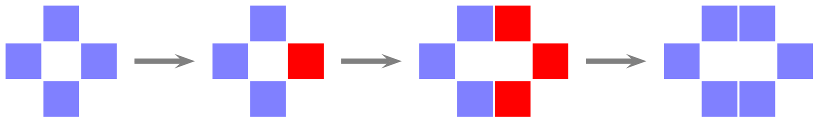


Figure 3. Curve dilating outward: initial, point to move, outward neighbors, final *(left to right)*.

to a fractal interface; with each iteration, points oscillate between sides of the true interface. Figure 5 illustrates such an interface oscillating. In a fully numerical scheme, the curve can settle with subpixel accuracy. To solve this, the phased approach restricts one direction of movement with each phase. This gives the interface a chance to catch up, preventing the jagged oscillation. With the two phased approach, whole smooth sections oscillate which gives a more appropriate result. Section 4.2 describes a faster, stripped down version of the algorithm which involves fewer speed computations but often results in a somewhat jagged interface.

The algorithm for curve propagation is listed in Procedure 2. For each point with nonnegative force it pushes the curve to its immediate neighbors in the appropriate direction. Figures 3 and 4 illustrate the process of determining neighbors and propagating in each phase. Note that after each phase of evolution, it is important to drop unnecessary points to maintain a minimal interface so that artifacts do not develop. With a minimal interface, we need only look to the four neighbors of any point when making decisions during evolution.

The final core routine for maintaining a minimal interface is listed in Procedure 3. Points along the interface are considered unnecessary if, in addition to touching other interface points, they only touch interior or only touch exterior points. Conversely, points are considered necessary if they touch both interior and exterior points. Figure 6 illustrates cleanup of the contracted interface in Figure 4.

2.2 Energy specific callbacks

Having described the core evolution mechanics, it remains to discuss the user-provided callbacks for force computation and bookkeeping as points cross the interface. This is the energy specific aspect of the curve evolution. We have found the above evolution algorithm to be generic and applicable to a wide range of curve evolution energies.

At every iteration, the user must compute the force, only the sign of which matters. The user is provided with the underlying ϕ and underlying points comprising the surface. Notice that since ϕ only describes the zero level set, it only makes sense to compute (approximate) derivatives $\nabla\phi$ and the force only along the interface. Since the zero level set is indicated explicitly, there is no need for interpolation onto the interface.

Additionally at each iteration, the user is provided an opportunity to adjust regional statistics, etc., in response to points crossing the interface. More specifically, we *move in* points that go from $\phi = 1$ to become $\phi = 0$ and we *move out* points that go from $\phi = 0$ to become $\phi = 1$. Since we consider the zero interface to

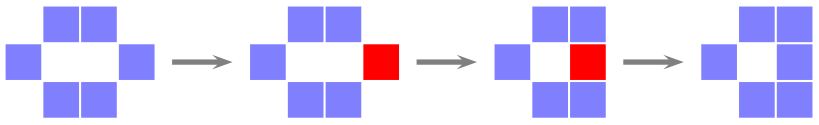


Figure 4. Curve contracting inward: initial, point to move, inward neighbors, final *(left to right)*. The result includes unnecessary points violating minimal interface principle; these are removed during cleanup (see Figure 6).

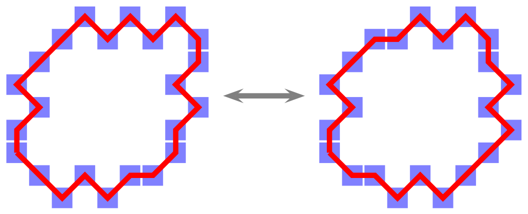


Figure 5. In an unfazed approach, jagged interfaces may develop when the interface oscillates upon convergence. Notice that the shape and position of the interface remain relatively stable despite oscillation.

.

be "inside", points that go from $\phi = 0$ to $\phi = -1$ do not change bookkeeping status; they are still considered "inside".

2.3 Extension to higher dimensions

Extension to higher dimensions is trivial provided a minimal interface is maintained according to Procedure 3. Neighborhoods now include two additional points to account for the added dimension.

3. EXAMPLE ENERGIES

In order to implement an energy, we must define callback routines for force computation and bookkeeping. To illustrate formulating these callbacks, we briefly examine three types of intensity-based segmentation energies.

It is important to note that, since these energies are computed directly on the interface ($\phi = 0$), we can discard support descriptors such as the Dirac delta $\delta(\phi)$.

3.1 Mean intensity separation

One of the most common segmentation strategies involves separating the mean intensity of two regions.[5,6] Assuming $\phi < 0$ to be the interior and using the Heaviside function $H(\cdot)$ to describe regional support, a typical continuous representation of this energy is:

$$E = \int (I(x) - v)^2 H(\phi(x)) + (I(x) - u)^2 H(-\phi(x)) dx, \tag{1}$$

where u and v are the mean intensities of inside and outside, respectively. A simple form of its gradient is

$$\nabla E(x) = (I(x) - v)^2 - (I(x) - u)^2. \tag{2}$$

For the proposed technique, force computation simply involves computing this expression at each point along the interface. Bookkeeping simply involves adjusting the regional means upon the movement of pixels, a computation that can be done recursively. For example, for a point moving from inside to outside at time t use the following:

$$u_{t+1} = \frac{u_t A_u - I(x)}{A_u - 1} \qquad \text{and} \qquad v_{t+1} = \frac{v_t A_v + I(x)}{A_v + 1}, \tag{3}$$

where A_u and A_v are the areas of inside and outside, respectively.

Procedure 2 Perform one iteration of curve propagation

```
for each point $x$ on curve do
  if nonzero force then
    if negative force then
      $\phi(x) \leftarrow 1$
    else
      $\phi(x) \leftarrow -1$
    end if
    Drop $x$ from interface

    for each neighbor $y$ of $x$ having opposite sign as $\phi(x)$ do
      $\phi(y) \leftarrow 0$
      Insert along interface
    end for
  end if
end for
```

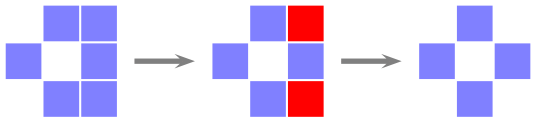


Figure 6. During cleanup, unnecessary interface points *(red)* are detected and removed to ensure minimal interface.

3.2 Mean and variance separation

Separation of mean intensity alone assumes the distributions to be of unit variance, an assumption that often leads to incorrect segmentation. Incorporating the log-likelihood of the normal distributions $\mathcal{N}(\mu, \sigma)$ directly into the energy allows separation by mean and variance,[7]

$$E = \int \log \mathcal{N}(u, \sigma_u) H(-\phi(x)) + \log \mathcal{N}(v, \sigma_v) H(\phi(x)) dx, \tag{4}$$

which has corresponding gradient:

$$\nabla E(x) = \log \frac{\mathcal{N}(v, \sigma_v)}{\mathcal{N}(u, \sigma_u)} \tag{5}$$

$$= \left(\frac{I(x) - v}{\sigma_v}\right)^2 - \left(\frac{I(x) - u}{\sigma_u}\right)^2 + \log \left(\frac{\sigma_v}{\sigma_u}\right)^2. \tag{6}$$

With the introduction of a temporary variable, the variances can be updated recursively, similar to how the mean values were updated in (3). We first un-center the variance, compute the expectation $\mathcal{E}[I^2]$, and finally recenter it. For example, for a point moving from inside to outside at time t use the following:

$$\sigma_u^{(t+1)} = \frac{(\sigma_u^{(t)} + u_t^2)A_u - I(x)^2}{A_u - 1} - u_{t+1}^2 \qquad \text{and} \qquad \sigma_v^{(t+1)} = \frac{(\sigma_v^{(t)} + v_t^2)A_v + I(x)^2}{A_v + 1} - v_{t+1}^2, \tag{7}$$

which make use of the already updated means u_{t+1} and v_{t+1}.

3.3 Distribution separation

Another common energy for separating regions involves maximizing the Bhattacharyya distance between two intensity distributions.[8,9] A typical continuous representation of this energy is:

$$E = d^2(\mathbf{p}, \mathbf{q}) = \int_{\mathcal{Z}} \sqrt{\mathbf{p}(z)\mathbf{q}(z)} dz, \tag{8}$$

Procedure 3 Drop unnecessary points to ensure minimal interface

```
for each point $x$ on curve do
  if neighbors' include only interior and interface then
    $\phi(x) \leftarrow 1$
    Drop $x$ from interface
  end if
  if neighbors' include only exterior and interface then
    $\phi(x) \leftarrow -1$
    Drop $x$ from interface
  end if
end for
```

where $\mathcal{Z}$ is the set of possible intensities and $\mathbf{p}, \mathbf{q}$ are the distributions for inside and outside, respectively. The gradient of this expression is found to include nested integrals:

$$\nabla E(x) = \underbrace{\frac{d^2(\mathbf{p},\mathbf{q})}{2}\left(\frac{1}{A_u} - \frac{1}{A_v}\right)}_{\text{global}} + \underbrace{\frac{1}{2}\int_{\mathcal{Z}} K(z - I(x))\left(\frac{1}{A_v}\sqrt{\frac{\mathbf{p}(z)}{\mathbf{q}(z)}} - \frac{1}{A_u}\sqrt{\frac{\mathbf{q}(z)}{\mathbf{p}(z)}}\right) dz}_{\text{local}}, \tag{9}$$

where $K(\cdot)$ is from the kernel density estimate of each distribution. Since we are already working in an approximate framework, we take the liberty of further approximations to implement this gradient. For example, we assume that the distributions are left unsmoothed and so this becomes:

$$\nabla E(x) = \frac{d^2(\mathbf{p},\mathbf{q})}{2}\left(\frac{1}{A_u} - \frac{1}{A_v}\right) + \frac{1}{2}\left(\frac{1}{A_v}\sqrt{\frac{\mathbf{p}(I(x))}{\mathbf{q}(I(x))}} - \frac{1}{A_u}\sqrt{\frac{\mathbf{q}(I(x))}{\mathbf{p}(I(x))}}\right). \tag{10}$$

As an implementation note, notice there is a global term and a local term that depends only on possible intensities encountered, both of which can be precomputed each iteration. Computing the speed along the interface merely entails looking up the local value for each intensity encountered and adding the global term.

4. EXTENSIONS

Here we demonstrate two simple extensions to the basic algorithm. The first extension is smoothing using integer Gaussian kernels, the second is an additional speed enhancement at the expense of a rougher interface.[4]

4.1 Smoothing

Typically, smoothing is incorporated as an additional term in the energy that penalizes curve length. This penalty takes the form of a Laplacian of ϕ the evolution of which equates to Gaussian filtering the signed distance function. It has been proposed to use the Gaussian filter response similar to the force thereby moving points if the sign of $\phi(x)$ differs from the filter response.[4] This can be decoupled from the energy into a subsequent phase of evolution shown in Procedure 4. Here, the main algorithm alternates between the two phased evolution and smoothing.

4.2 Further speed increase

As mentioned in Section 2.1, portions of the interface can become jagged as they oscillate upon convergence (see Figure 5). The two phased approach addressed the jagged interface. However, if the smoothness of the interface is not critical to the application, the algorithm can be further simplified to one phase with no restriction on direction (see Procedure 5). This approach can be significantly faster with monotonically advancing fronts, where the phased approach would discard the work of every other force computation.

Procedure 4 Main algorithm with smoothing.

```
for each iteration do
  for each iteration of evolution do
    Callback: initialize iteration
    Restrict to contraction (only processes positive forces)
    Propagate, cleanup (Procedures 2, 3)
    Callback: move points in and out

    Callback: initialize iteration
    Restrict to contraction (only processes negative forces)
    Propagate, cleanup (Procedures 2, 3)
    Callback: move points in and out
  end for

  for each iteration of smoothing do
    Compute integer kernel along interface (use as force)
    Propagate, cleanup (Procedures 2, 3)
    Callback: move points in and out
  end for
end for
```

Procedure 5 Faster unfazed version without smoothing.

```
for each iteration do
  for each iteration of evolution do
    Callback: initialize iteration
    Propagate, cleanup (Procedures 2, 3)
    Callback: move points in and out
  end for
end for
```

5. EXPERIMENTS

Intensity segmentation was performed using mean, variance, and full distribution separation. As in all iterative optimization problems, timing varied based on initialization and energy employed; however, all experiments took between 100ms and 2s for convergence.

Figure 7 shows that the discrete technique may miss some detail because it lacks subpixel accuracy. Here the discrete curve was unable to move in between the two ventricles. This experiment used the mean and variance separation energy (4).

Figure 8 shows segmentation of an image volume using the mean separation energy (1). Figure 9 shows segmentation of the white matter using the Bhattacharyya distance (8). Notice that the method is able to capture much of the sulci undulation.

In exploring the numerical properties of this formulation, we have found that artifacts may develop due to the neighborhood size used in the rules governing maintenance of the regions. Figure 10 shows a circle collapsing inward under unit speed. We should expect the circle to remain round; however, the discrete approximation develops into a square in the process of collapsing. Note that the final solution is still correct.

6. CONCLUSION

The proposed method is a simple form of curve evolution suitable for high performance surface propagation. The process of energy implementation is shown to be simple and straight forward. The resulting segmentations are accurate up to sub-pixel accuracy.

All experiments showed the standard method without smoothing (Procedure 1). In experiments with the faster version (Procedure 5), we noticed that this fast method works best if gradient computation detects for

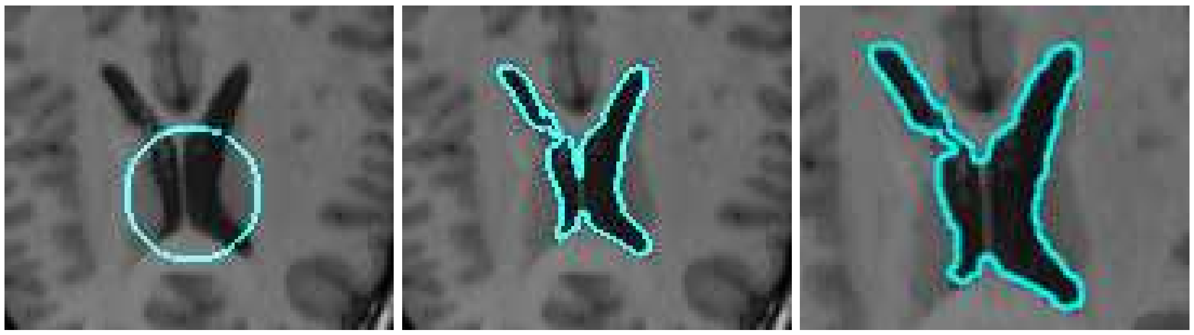


Figure 7. Intensity-based segmentation of the ventricles using mean and variance (time: <100ms): initial, full numerical implementation, proposed discrete implementation *(left to right)*. As expected, the discrete approximation captures all but the sub-pixel width tissue separating ventricles.

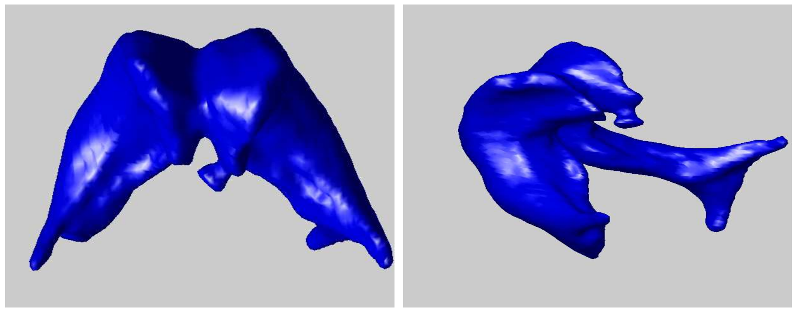


Figure 8. Ventricle segmentation using mean (time: ∼400 ms). Initialized from single bubble placed in each ventricle. Results are shown from two views: front, side *(left to right)*.

convergence and zeros out forces; this dampens or eliminates oscillation. In order to achieve computation speeds that are faster yet, we may reformulate the evolution rules in terms of cellular automata to capitalize on the highly parallelized nature of the evolution.

Both C and Matlab versions of this algorithm are provided online.*

ACKNOWLEDGMENTS

This work was supported in part by grants from NSF, AFOSR, ARO, MURI, as well as by a grant from NIH (NAC P41 RR-13218) through Brigham and Women's Hospital. This work is part of the National Alliance for Medical Image Computing (NAMIC), funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 EB005149. Information on the National Centers for Biomedical Computing can be obtained from `http://nihroadmap.nih.gov/bioinformatics`. The authors would also like to thank Caribou coffee for providing much of the motivation behind this paper's development.

REFERENCES

1. D. Adalsteinson and J. A. Sethian, "A fast level set method for propagating interfaces," *J. of Computational Physics* **118**(2), pp. 269–277, 1995.
2. R. Whitaker, "A level-set approach to 3D reconstruction from range data," *Int. J. of Computer Vision* **29**(3), pp. 203–231, 1998.
3. F. Gibou and R. Fedkiw, "A fast hybrid k-means level set algorithm for segmentation," in *Int. Conf. Statistics and Mathematics*, pp. 281–291, 2005.
4. Y. Shi and W. Karl, "A fast level set method without solving PDEs," in *Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 97–100, 2005.
5. T. Chan and L. Vese, "Active contours without edges," *Trans. on Image Processing* **10**(2), pp. 266–277, 2001.

*`www.ece.gatech.edu/∼malcolm`

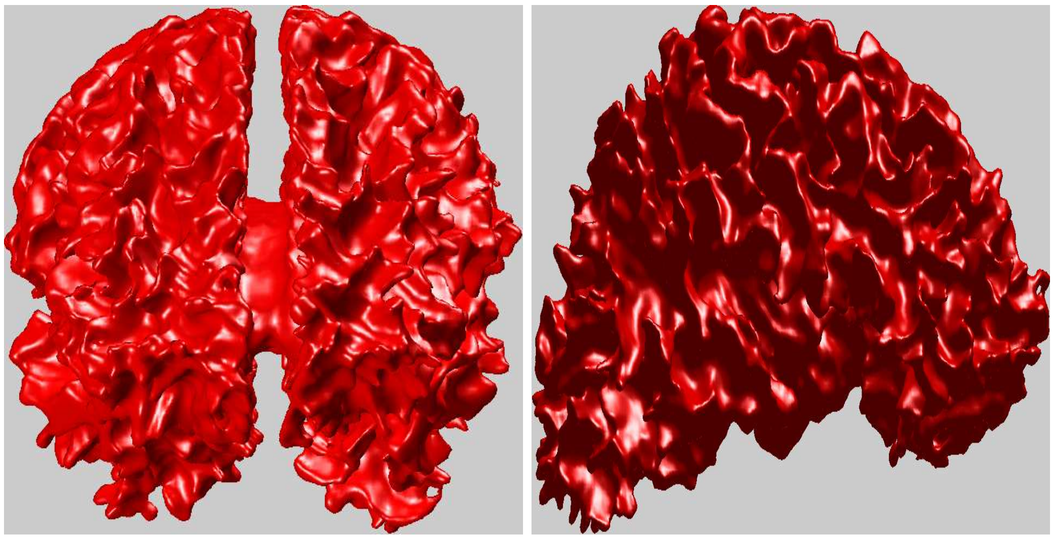


Figure 9. White matter segmentation using Bhattacharyya distance.[8] Initialized from three bubbles placed in each hemisphere. Results are shown from two views: front, side *(left to right)*. Notice that the method is able to segment the fine structure despite the discrete approximation.

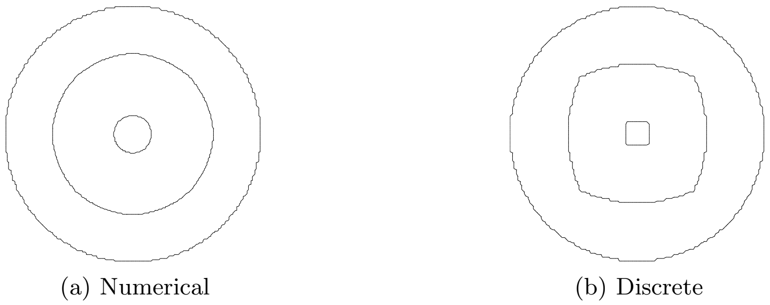

(a) Numerical (b) Discrete

Figure 10. Circle shown at various time steps while shrinking inward under unit speed: full numerical implementation and discrete approximation *(left to right)*. In both techniques the circle disappears to a point as expected.

6. A. Yezzi, A. Tsai, and A. Willsky, "A statistical approach to snakes for bimodal and trimodal imagery," in *Int. Conf. on Computer Vision*, pp. 898–903, 1999.
7. M. Rousson and R. Deriche, "A variational framework for active and adaptive segmentation of vector valued images," in *Workshop on Motion and Video Computing*, pp. 56–61, 2002.
8. Y. Rathi, O. Michaeilovich, J. Malcolm, and A. Tannenbaum, "Seeing the unseen: Segmenting with distributions," in *IASTED Conf. on Signal and Image Processing*, 2006.
9. T. Zhang and D. Freedman, "Tracking objects using density matching and shape priors," in *Int. Conf. on Computer Vision*, 2003.