COMPETITIVE MEDICAL IMAGE SEGMENTATION WITH THE FAST MARCHING METHOD

by

JONATHAN HEARN

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

Thesis Advisor: Dr. M. Cenk Cavusoglu

Department of Electrical Engineering and Computer Science

Case Western Reserve University

May, 2008

Contents

List of Figures

List of Tables

Competitive Medical Image Segmentation with the Fast Marching Method

Abstract

by

JONATHAN HEARN

Extensions to the fast marching method are introduced with the aim of further automating the segmentation of medical image data. A competitive algorithm is used to minimize the occurrence of bleeding across boundaries, and techniques including automatic starting point selection, statistical region combination, and user-influenced region expansion are introduced to support the competitive paradigm. Results show good performance in a variety of scenarios, with poorer performance in the presence of high-intensity noise and unusually obscure boundaries.

1 Introduction

With continuous advances in medical imaging technology, useful image segmentation algorithms are becoming progressively more important to efficient patient diagnosis and treatment. Tools such as magnetic resonance imaging (MRI) make it possible for physicians to make non-invasive diagnoses in patients with various ailments, but this ability is hindered by limited capability to visualize and present the gathered data. Current graphical display technologies are highly developed to meet these needs, but shortcomings in automating the interpretation of captured data limit the usefulness of these technologies, particularly in segmenting data to identify different anatomical entities. While there are tools to assist in doing this by hand [30], it is still very expensive; it requires significant time and effort from knowledgeable people. Additionally, consistent increases in the resolution and speed of devices that capture medical data have made the problem a continually worsening bottleneck in effective treatment, making the processing and registration of medical image data an active area of research. In this thesis, we will present tools that aim to provide robost results in such image segmentation with minimum user interaction.

There are a number of popular front propagation models used in image segmentation, some more complex than others. Marching cubes [13] is a relatively simple algorithm by which a front is moved forward based on each pixel's intensity relative to some threshold value. Marching cubes, and other similar methods, have shown significant results using thresholding. One such example is in [26], where the brain section of an MRI head image is identified via propagation of a round mesh outward from the intensity-weighted center of the head. Additional work on an extended version of marching cubes is discussed in [11]; higher quality results are produced by increasing the resolution of the grid of cubes where edges occur. A major weakness of thresholding methods is that they may generate, depending on the threshold, too few or too many edges for the given application. While having too few results obviously

cannot be remedied, some methods exist for the removal of unwanted portions of edges [20].

Where detailed segmentation results are necessary, more complex front propagation methods such as the level set or fast marching methods, are required. [25] discusses use of the fast marching method in a probabilistic multi-label framework with a priori knowledge used to influence the results. In [2], an extension of the fast marching method is used to track moving objects. These methods are often preferred because of the smooth edges that can be obtained, but in some cases this is not considered significant; [10] discusses modifications to the level set method specifically for preserving rough edges. The mechanics of the levelset and fast marching methods are central to this study, so they will be discussed in greater detail later.

Region growth, merging, and competition methods discussed in [31] provide a statistical analysis of image data. In these algorithms, a region starts as a single point or small group of points, then expands outward adding bordering points if they meet a certain statistical criteria dependent on the intensity of points already in the region. Merging and competitive angles on this method provide flexibility in scenarios where two regions meet. A fundamental weakness of these methods is that poor initial region locations, such as the edge of what would otherwise be a single unsegmented structure, could significantly limit the output quality. A suggested enhancement to get around this problem is to use image-global energy functions to determine the best separation of regions, though these functions can be hard to formulate when attempting to come up with a general solution. While region growing alone is probably insufficient for very complex medical images, it may offer a significant gain when combined with other more promising methods. For example, [3] discusses the combination of region based and gradient based approaches using game theory.

Another popular method in medical image segmentation is the use of deformable models, as discussed in [17]. This algorithm starts with some geometric model that

approximates the shape of the structure whose boundary is being sought. The model is then expanded outward to meet the structure's actual boundaries. In [9, 6], this method is advanced further by using a model that is tailored to a specific anatomical structure; [5] provides methods for the generation of such models. These methods can produce very good results, but by nature require a significant amount of user input as well as some a priori knowledge.

Despite the loss of generality inherent in such methods, the use of significant a priori knowledge or pre-existing edge data is quite popular. [4] discusses a method for the reconstruction of incomplete edges from sets of loosely connected points by locating the saddle points between them. A similar effect for 3 dimensions is discussed in [7, 8], where a surface is reconstructed from points on that surface by optimizing and then fitting to a mesh model. In a more dramatic customization, [29] introduces a method of segmenting MRI data that is dependent on specific properties of the tissue being segmented.

Medical images are segmented using the KMeans clustering algorithm in [19] with reasonable results. Because of the disconnected regions that are generally created by such a method, [12] introduces KMeans with Connectivity Constraint (KMCC) as a modification of the algorithm that takes spatial separation of the voxels into account when doing the clustering. This method lends itself to less rigorous uses than medical imaging, such as video and photo processing [18], where the parameter of primary interest is intensity, not arrangement, of pixels. Consequently, when used on medical data, these methods would be better suited to categorizing tissues than finding boundaries between structures.

Another interesting method is the watershed transformation discussed in [28]. The author cites examples that make it appear effective in MRI segmentation, but it requires an initialization process where connected sets of points must be provided. We will see that this method is not entirely unlike our own algorithm in that it attacks

the problem by searching for multiple regions simultaneously, a characteristic many other algorithms lack.

In this thesis, we will draw on several of these methods to come up with a set of tools that, taken together, introduce a new way of segmenting medical image data. The goals of this method will be minimizing user interaction and data-specific input while maximizing segmentation quality. We start by introducing the level set and fast marching methods, which serve as the core of our algorithm. Then a number of modifications to the conventional fast marching method will be introduced to increase the level of automation and generality in the algorithm. Several images, some contrived to test specific attributes and others from actual medical data, are used to test and evaluate the effectiveness of our new methods.

These discussions focus on segmenting two dimensional grayscale images only; the ultimate target for our methods, magnetic resonance data, is generally presented in grayscale, and working in two dimensions allows for better result visualization and evaluation. However, all algorithms are designed to make extension into 3 dimensions a straightforward process. Finally, we draw conclusions and look at potential extensions to our work.

Contributions

With the exception of statistical methods, image segmentation literature offers very little with respect to competitive algorithms. Furthermore, there are no such methods that do not require either the subjective selection of a threshold value or significant a priori information to generate starting points for growth. The scarcity of front propagation methods in competitive algorithms is significant since front propagation methods can produce higher quality edges than statistical methods in many scenarios. Results discussed in this thesis will evaluate the effectiveness of one such approach.

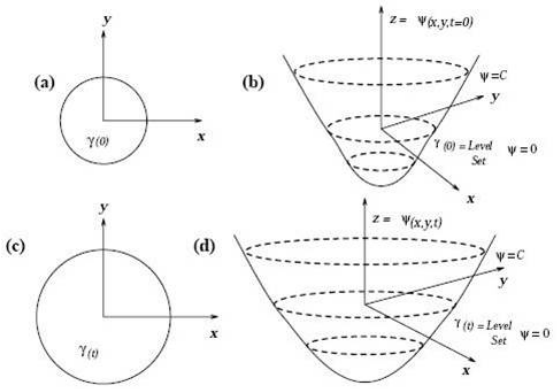
Some portions of the method we discuss can potentially find use in other image

processing algorithms. Specifically, our starting point initialization techniques could be used to initialize various algorithms, and our front coercion method might prove useful in assisting user-guided segmentation whenever the fast marching method is utilized.

2 Background - The Level Set and Fast Marching Methods

The strength of the level set method [24, 16, 21, 27, 23] is in the extension of an N-dimensional front into an N+1-dimensional space. This is shown in Figure 1; the 2 dimensional expanding circular front is extended into 3 dimensions (1a and 1b), such that the state of the original front at any given time can be viewed as a level set of one of a 3-dimensional family of surfaces. That is, γ(t) is the series of points where $\psi_t(x, y, t) = 0$.

Figure 1: Extension of a 2-dimensional front into a 3-dimensional space. [14].



We then go a step further with this by calculating the propagation of the N dimensional front in terms of our N+1 dimensional front. In general, the propagation through the higher dimensional family of fronts can be described by the equation

$$\psi_t + F\,|\nabla\psi| = 0, \tag{1}$$

where ψ_t is the difference between the current surface and the next surface in the N+1-dimensional family of surfaces, F is a function describing the speed at which the N-dimensional front should propagate, and $|\nabla\psi|$ is the gradient magnitude of the current N+1-dimensional surface[14]. Using this equation, we can determine

the position of γ by iteratively solving for ψ and then taking the level set of the appropriate curve[23].

At first glance, this method may seem unnecessarily indirect, but it comes with a number of advantages. We will be most concerned with two in particular. First, regardless of the topology of our front, the higher-dimensional representation will always be a function. This makes features such as cusps, which would otherwise be mathematical discontinuities, trivial to deal with. Second, the use of a generalized propagation function F introduces a high degree of flexibility into the algorithm. While the possible inputs for motion of the front are not discussed in this paper, the ability to manipulate the way the image boundaries are located is a strong argument for the extendability of the other techniques we introduce.

However, the level set method also has some disadvantages. First, it is computationally expensive to iteratively re-evaluate the position of ψ in a three dimensional space. While methods have been introduced to mitigate this problem [1], we are also left with the difficulty of choosing a suitable starting surface. Finally, while the continuous nature of these computations is very helpful in formulating detailed representations of real-world data, our data will already be discretized, so this is an unnecessary complication for us in scenarios where the output is to be used in a discretized form. In order to avoid these setbacks, we will instead concentrate our algorithm on an extension of the level set method more suitable to our purposes known as the fast marching method.

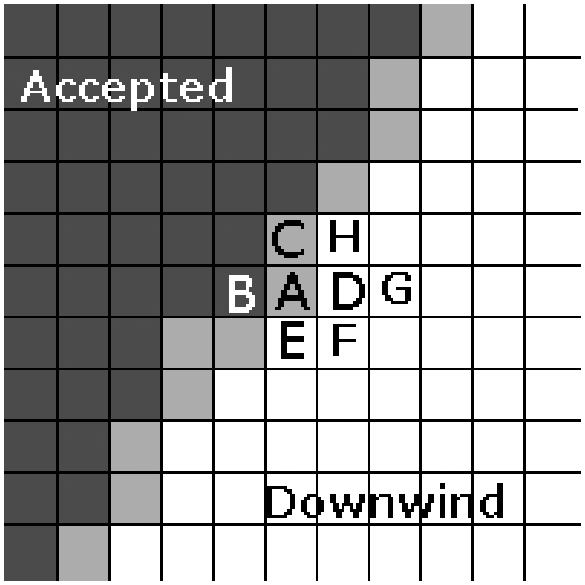
The fast marching method is essentially a discretized, computationally optimized version of the level set method. It introduces the constraint that a front may only expand monotonically, but we will see that this will have no negative impact on our other techniques. In exchange for this constraint, a very good approximation of our front's expansion can be calculated with a single pass over the data space. Supposing that $F(x, y)$ describes the monotonically expanding movement of the front at each

point it will cross, we can now also define a function $T(x, y)$ that defines the time at which the front arrives at a given point (x, y). It is then evident that

$$|\nabla T|\, F = 1, \tag{2}$$

or the gradient of the arrival time is inversely proportional to the speed of the front[15]. If we can now solve this equation for T through the entire 2-dimensional space, starting from the initial location of the front with $T = 0$, we will have a representation of the front's propagation according to F. The method by which this is done is illustrated in Figure 2.

Figure 2: Pixel by pixel expansion of a front using the fast marching method. The front can be assumed to have started propagating from the pixel in the top left corner. All of the dark gray points are taken to be 'accepted', in that their arrival times will no longer change. The 'downwind' white pixels have yet to be calculated, and so their arrival times are infinity. The lighter gray points are in the process of being calculated, and these are what is referred to as the narrow band.



The algorithm works pixel by pixel, selecting the point in the narrow band (see Figure 2 caption) that has the lowest arrival time by current calculations. For the sake of our example, we suppose that this is the pixel indicated A. We then add A to the Accepted collection of points, and all points adjacent to A that are not already

Accepted (points C, D, and E) have their arrival times calculated, or recalculated, based on the current arrival times available. Suppose we are calculating T_D, the arrival time of D. If $T_x = min(T_G, T_A)$, $T_y = min(T_H, T_F)$, and F_D is the value of our force function at pixel D, the final value of T_D is calculated with one of the following equations.

$$T_D = T_x + 1/F_D \tag{3}$$

$$T_D = T_y + 1/F_D \tag{4}$$

$$(T_D - T_x)^2 + (T_D - T_y)^2 = 1/F_D^2 \tag{5}$$

We use Equation 3 if $T_x < T_y - 1/F_D$ and we use Equation 4 if $T_y < T_x - 1/F_D$. Otherwise, T_D is the largest root of Equation 5. To complete processing of A, these calculations are repeated for pixels C and E, and pixels D and E are added to the narrow band. For a more rigorous proof that these equations accomplish our goal of solving Equation 2, see [22].

Once this operation has been performed across an entire image, we have a very good approximation of the the growth of the front with respect to time. It is worth noting that this is only an approximation because the discretization of the data introduces a margin of error; if the input data is discretized, then there is no reason to consider the results inaccurate. Assuming that time goes on to infinity, however, it is apparent that we still need a mechanism to prevent the expanding front from swalling the entire image, presumably using some stopping criteria. This is a significant difficulty with this method, which is why it is suggested in [15] that the force function be designed to minimize the fast marching overshoot effect, then the results can be used as a starting point for the full levelset method. Since the levelset method

does not require a monotonically advancing force function, a more complex function can be designed to draw the front back toward a boundary rather than blowing past it. However, this introduces the extra computational effort of running the levelset method, and engineering such a force function can be very difficult if it is to be used for different structures. The next section will explain our attempt at solving the stopping criteria problem without such a mechanism.

One flexibility we gain by not requiring such a complex mechanism is that our method may be enhanced by introducing new factors into the force function as long as the resulting function is monotonically increasing. For all examples in this paper, we will use the very simple force function discussed in [15],

$$F(x) = e^{-\tau(|\nabla I(x)|)}, \tag{6}$$

where x is some point in the image, $|\nabla I(x)|$ is the magnitude of the image gradient at that point, and τ is a positive parameter to adjust the strength of the function. This equation provides the simple behavior of slowing the front down as it approaches a high image gradient, and speeding up the front in uniform areas. When images have a significant amount of noise or more detail than is desired, it can also be beneficial to use an edge-preserving smoothing filter on the image before calculating I.

3 Method

Most of the programming for this thesis was performed in Matlab. A small library was written in C++ to quickly locate the minimum t-value in the narrow band so the fast marching method could be performed efficiently. C++ was also used to tabulate and keep track of certain statistics. With the exception of simple filtering, no Matlab image processing libraries were used; all algorithms were implemented from scratch to allow flexibility and experimentation.

We start by explaining the core motivation of our method; the use of competitive region growth has potential to produce better results than existing methods of image segmentation. Next, we explore different methods to automatically locate starting points from which regions will grow. After expanding these regions with the fast marching method, a scheme is needed to combine the regions into the larger structures we seek through segmentation; our method of doing this will be described. Finally, since it is inevitable that a user will in some cases want to modify the outcome of the automatic mechanism, we introduce changes to our algorithm that allow region growth and joining to be coerced into meeting a user's specifications.

Competition

Existing studies of the fast marching method concentrate on locating a single entity in the context of all other entities in the image data. This translates to the algorithm repeatedly answering a fundamental question of whether or not some pixel is part of a particular region. Unfortunately the data available to give a definitive answer to this question using current techniques is limited; essentially, where the discretized front stops is the determining factor. While the force function can be modified to allow for more complex criteria to find the solution, the implementation still comes down to image gradient, shaping factors such as curvature, and user-entered or ontological data

that pushes the algorithm to the correct conclusion. The last of these factors is very powerful, but some types of user input can be difficult to work into an algorithm, and the more such information required by an algorithm, the more impractical it becomes from a usability standpoint.

With competition we make a few assumptions to reformulate the segmentation question into something more easily answered with the data available. First, we acknowledge that every pixel in an image belongs to some region that could be sought. This is a self-evident statement, so long as we make clear that a region is any semi-uniform, contiguous series of one or more pixels; in an MR slice of the brain, the uniform area that lies outside of the subject's head, for example, can still be considered a region even if we are not particularly interested in it. Having accepted this as a postulate, we can then go further to state more directly that the entire image is made up of regions, and that any region we might be interested in is entirely bordered by other regions. Further still, the borders of any region can also be defined by the borders of its neighboring regions and the space outside of their boundaries that they share but do not consume. From this reasoning, we go forward with our reformulation of a slightly different boundary search question.

By attacking not only a region of interest but any regions around it as well, we give an algorithm much more to work with. Instead of having to determine if a pixel is an element in some region or not (or, stated differently, which of two sets it lies in, one of which may be highly heterogeneous), we can ask instead which of all available regions the pixel belongs to. When each region has its own set of characteristics that help to accept or reject a particular pixel, the primary task of determining whether or not a given pixel is part of the structure of interest can be performed in a much more informed way.

The competition is performed indirectly by using the relative speed of multiple fronts in the image. A number of fronts are expanded throughout the image according

to the usual fast marching mechanism. If some front A reaches a point before another front B does, then that points belongs to region A. As two regions meet each other, they no longer expand toward one another. It is worth noting, however, that in order for this to truly implement a competitive approach, there must be a sufficiently high number of regions that competition for points will take place before any region overexpands.

We hope to gain two concrete advantages from this method. First, a stopping criterion is no longer necessary, since our goal is to have all our regions fill the entire image. This is a significant benefit, since timestep or gradient based stopping criteria we might otherwise develop would need to be region specific, which reduces the usefulness of the algorithm. With the competitive method, the stopping criterion is to stop when no more computation can be done; either all pixels have been consumed by some region, or the maximum timestep value has been reached, so any remaining pixels will never be reached using the current parameters.

The second and more fundamental advantage is our ability to control the growth of the region of interest beyond its actual biological boundaries. When 'bleeding' of a region beyond the edges of the anatomical structure it represents might occur under the conventional fast marching method, competition makes it likely that points outside of that anatomical structure will be absorbed by more appropriate regions before the front extends too far. It is also worth noting that while a non-competitive fast marching method could be enhanced to counteract this by providing a more robust force function, the competitive method can also be further enhanced in this same way, since the mechanism is independent of the force function being used.

Automatic Initialization

The manual selection of starting points is one characteristic that is common to nearly every front propagation and image segmentation algorithm in prior work. A successful

technique to automatically choose starting points would likely receive wide use, and the need for such a method is made more severe by the previously discussed concept of searching for the whole image simultaneously. manually putting starting points into all the entities in an image is both laborous and prone to accidental omission.

Our automatic initialization methods have been created based around a few high level goals in how starting points should be placed. First, front propagation should start in the most uniform areas of the image. This is reasonable, as a front will spread across a region more effectively if it moves uniformely in all directions, so starting at a point or set of points far from its edges is the best technique. Next, we want to be sure we have sufficient starting points to capture all detail in the image. Small regions which do not have starting points in them will be omitted from the results, or will be swallowed up by neighboring regions whose boundaries are not greatly pronounced. Finally, since a finite amount of computing power is available to evaluate the algorithm, fewer starting points are preferable to more starting points. The remainder of this section attempts to attack these goals with two different methods.

Grid Initialization

The grid initialization method is based on initializing the image with a simple grid of starting points. The image is divided into squares of equal size, and a starting point is placed in the center of each square. The size of the squares is parameterized such that it can be chosen by the user, as different levels of spacing are helpful depending on the level of detail in the image. Since this set of points is initially chosen with no regard for the actual image data, it must be adjusted to more adequately satisfy our goal of starting in uniform areas. This can be done in two ways, the first being the omission of points, and the second the adjustment of points.

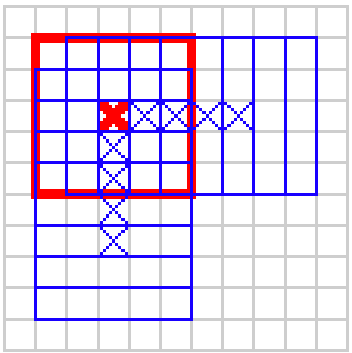
To determine which points should be kept and which should be eliminated, we use the equation

$$\sigma_s^2 < \alpha\sigma_I^2, \tag{7}$$

where σ_s^2 is the variance of the grid square surrounding the starting point, α is a user-selected constant, and σ_I^2 is the variance of the entire image. This means that we keep only starting points whose surrounding pixels are more uniform than the image as a whole.

In images with a higher level of detail where exclusion of points could result in loss of regions, we may shift starting points instead of eliminating them. Equation 7 is still used to determine whether or not a given point should be shifted. If so, the point is shifted right or down, one point at a time, so that it covers the width and height of the square that contains it. At each new point, the variance of the surrounding square is calculated. The center of the square with the smallest variance is then chosen as the starting points. If none of the calculated variances satisfies Equation 7, then the point is altogether discarded. This is illustrated in Figure 3.

Figure 3: Each small square in the grid shown represents a single pixel in the image data. The bold red 'X' represents the original position of the starting point and the surrounding bold red square surrounds the pixels whose variance is calculated to obtain σ_s^2 in Equation 7. If the equation is not satisfied, the variance of all pixels contained by each of the 8 overlapped blue squares will be calculated. Whichever of the blue squares produces the lowest variance, the blue 'X' in the center of that grid square will be the shifted location for the starting point. (Note that this example assumes a grid square size of 5x5 pixels. For any other size, the process would be similar.



Division Initialization

In images with varying levels of detail, grid-based approaches of initialization may prove impractical. An extremely fine grid may be necessary to bring out fine details in the image, but this can significantly increase the required computation resources, and introduces unnecessary risk of errors in areas of the image with low levels of detail. For these situations, we provide the division approach. The method is essentially a recursive cutting of the image into smaller chunks until all of the pieces have sufficiently low variance.

The image is cut into 4 equal chunks and the variance of each chunk is used as σ_s^2 in Equation 7. If the equation is satisfied, a starting point is placed in the center of the chunk and it is not processed futher. Chunks that do not satisfy the equation are cut into 4 smaller chunks and the process is repeated. This goes on until all chunks satisfy Equation 7 or a user-defined minimum chunk size is reached. The minimum chunk size is necessary because continuous cutting will inevitably lead to chunks of 1 pixel, which will result in an impractical number of starting points in locations of high detail or significant image noise.

One additional calculation is performed for chunks that have reached the minimum size. Since many rasterized boundaries may appear as mostly horizontal or vertical when looking at a very small section, it is possible that significant benefit can be had by cutting the chunk once more into two pieces so that there can be a starting point on opposite sides of the boundary. We attempt to take advantage of this by determining whether to cut the chunk horizontally or vertically for the greatest impact. To make this decision we use the equation

$$\epsilon = (\sigma_T^2 + \sigma_B^2) - (\sigma_L^2 + \sigma_R^2), \tag{8}$$

where σ_T^2, σ_B^2, σ_L^2, and σ_R^2 are the variances of the top, bottom, left, and right halves

of the chunk, respectively. If ϵ is positive, we cut the chunk into top and bottom halves and put a starting point in the center of each half. If ϵ is negative, we create starting points in the centers of the left and right halves. If ϵ is 0, then we presume that no benefit can be had from further splitting, and we simply put a starting point in the center of the chunk.

In many cases the division approach should presumably be more effective than our grid-based approaches at fulfilling our goals for initialization. Use of Equation 7 should help locate points in positions of low variance. Additionally, the analysis of different portions of the image adds a local dynamic to the algorithm, ensuring that there will be significantly more starting points in areas of high variance, while very few points will be used in large, uniform areas, allowing us to minimize unnecessary computation.

Region Joining

Automatic initialization of starting points in our image introduces a problem. While user interaction is significantly decreased, the number and location of regions is determined with no regard for the number and locations of biological entities that are actually present in the data. The result of this is a sort of stained-glass effect, where there are a number of small regions which show the boundaries that we are interested in, but also divide biological entities further into small pieces with additional boundaries. We will solve this problem by performing intelligent joining of the regions produced by our method with the aim of having our regions be recognizable as anatomical structures.

As the fronts expand and meet each other, which regions contact one another is recorded. When the regions are all fully expanded (either the timestep has reached infinity or all pixels in the data are consumed by some region), we will perform a statistical evaluation of each contacting pair of regions to determine whether or

not they should become a single region. Another possibility would be to perform the joins immediately as the regions come in contact, which might have an impact on their propagation if the fast marching method's force function had statistical dependencies. In this study, we do not pursue this approach, as our force function has no such dependencies, and recalculating the region statistics adds unnecessary computational expense.

To determine whether two touching regions A and B should be joined together, we will use the relatively simple solution of checking that the sample intensity means are within some distance of one another, or

$$\lambda_m > abs(\mu_A - \mu_B), \tag{9}$$

where λ_m is some number close to 0.

Since we are determining whether or not to join regions based on a statistical value, there is another question in the implementation of the joining algorithm. When two regions are joined together, should their statistics also be combined for the purpose of further joining, or should they stay separate? To deal with this we have another user-defined setting which we will call connectivity. When connectivity is turned on, two regions A and B that are joined will have their statistics combined, and will be treated as a new larger region as the joining algorithm continues to run. If connectivity is turned off, the regions will remain separate for the purpose of joining, but will be recognized as a single region when the algorithm is complete. We will see in the results that this setting can have a significant effect in regions where the image gradient tapers off to nearly matching the intensity of other nearby regions.

User Coercion of Front Propagation

Since the combination of the above methods is largely aimed at automating the segmentation process, there are bound to be situations in our biological data where the user wants to give the algorithm a push in the right direction toward finding the correct boundaries of a particular entity, specifically one with an unusual shape. We address this problem by providing a mechanism for the user to specify a series of points that should be attached together when the algorithm is completed.

After automatic initialization has been performed, the starting points are displayed and the user can select a series of them that should be attached together when the front propagation algorithm has run its course. This information affects the algorithm in two ways. First, it sets a flag to indicate that if the regions created by any two of those points come in contact with one another, they should be joined together regardless of their intensity statistics. Second, bias factors are applied to the force function for the selected points to make the regions selected grow more quickly in the relevant directions. The user is given the opportunity to do this for more than one group of points, though this detail is not critical to the functionality of the algorithm.

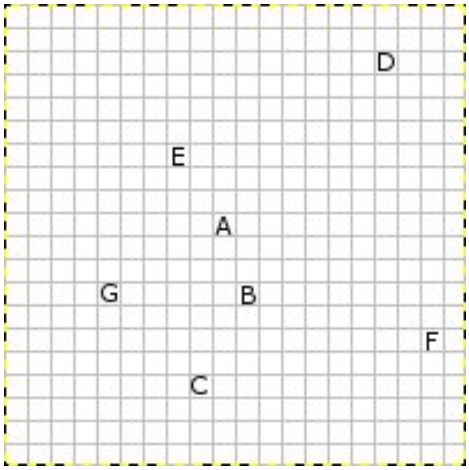
In our four-connected grid (working in two dimensions), we will have four separate bias factors for each starting point, one in each direction. Each of these factors will be determined by the point in the user-specified grouping that is farthest away from the relevant starting point. This is illustrated in Figure 4.

Using these points, the bias factors in each direction are determined by the equation

$$\beta_d^P = \delta_d \rho, \tag{10}$$

where β_d^P is the bias factor for starting point P growing in direction d (where d can be up, down, left, or right), ρ is a constant chosen to determine the strength of coercion,

Figure 4: Suppose all indicated pixels in the figure are user-selected starting points from which regions will grow. In determining the right, left, up, and down bias factors for point A, points F, G, D, and C will be used respectively. Point B and E will play no part, as they are not the extremes.



and δ_d is the distance between point P and the point in the user-selected grouping that is farthest in direction d. Note that δ_d is not the Euclidean distance, but the horizontal or vertical component of the Euclidean distance.

As a region grows from point P, the bias factors will be used to modify the force at the points on the outer edge of the region. This is done separately for each edge point as the front moves outward, and is described in the equations

$$F'_E = F_E\sqrt{\phi_V^2 + \phi_H^2}, \tag{11}$$

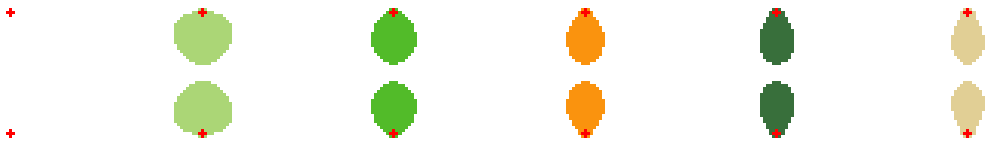
$$\phi_H = \beta_H^P \left(\frac{\delta_H}{\sqrt{\delta_H^2 + \delta_V^2}} \right)^{\nu} + \sqrt{\frac{1}{2}}, and \tag{12}$$

$$\phi_V = \beta_V^P \left(\frac{\delta_V}{\sqrt{\delta_H^2 + \delta_V^2}} \right)^{\nu} + \sqrt{\frac{1}{2}}, \tag{13}$$

where F_E are F'_E are the edge point E's original and biased force values, respectively, β_V^P is the vertical bias factor determined by Equation 10 (β_{up}^P if point E is above the starting point P, or β_{down}^P if point E is below P), β_H^P is the horizontal bias factor (either β_{left}^P or β_{right}^P), and δ_H and δ_V are the horizontal and vertical components of

the Euclidean distance between points P and E. This amounts to a position-weighted bias of the original force value F_E. Adding $\sqrt{\frac{1}{2}}$ in Equations 12 and 13 ensures that if the bias factor is 0 for one or both components, the resulting factor in Equation 11 will never be less than 1. This makes sense, since we would never want the coercion to slow down front propagation instead of speeding it up.

We use ν, which we call the bias inhibitor, to make the effect of the bias more directional. Because F'_E will be used as the outward force in all directions from edge point E, using these equations without ν will result in the region quickly swelling in all directions as it draws away from the starting point rather than just in the directions we desire. Since ν's operand will always be between 0 and 1, larger ν values will bring ϕ_H very close to $\sqrt{\frac{1}{2}}$ for edge points whose horizontal distance from the starting point is close to 0. The effect is the same for ϕ_V. This is illustrated in Figure 5.



(a) Start (b) $\nu = 1$ (c) $\nu = 3$ (d) $\nu = 5$ (e) $\nu = 7$ (f) $\nu = 9$

Figure 5: Result of the bias inhibitor. The two starting points have been selected for coercion such that their force functions are biased toward one another. Each image shows the points expanded on a blank background with different values for ν. All other expansion parameters are held constant, and all images show the same time step. As the value of the bias inhibitor increases, the regions grow more directly toward one another's starting points.

There are a few issues worth noting about this mechanism. First, the distance away from the starting point is not a factor in how the front expands. This means it is theoretically possible for the front to propagate farther in that direction than might have been intended. However, because the formula is intended to draw near

other starting points, in most cases the fronts for other points in the region should collide before expanding beyond the intended boundaries. Another issue is that the algorithm is designed to bias in straight lines, favoring the shortest distance between two points. This limits its usefulness for particularly complex shapes, but it can still be quite effective for elongated shapes or single structures that have varying intensities, where the algorithm might otherwise be led to identify them as multiple structures. Finally, this mechanism does not force two regions to come together, but simply modifies the growth of the regions so it is more likely that they will contact, allowing them to be joined. This is unfortunate in that two points that the user has specified to be joined may not do so in some situations, but if the regions were forced to come together, the boundaries of even moderately curved shapes might be plowed over by the evolving front if doing so were the most direct route to other destination points.

4 Results

To isolate the different characteristics and requirements of our algorithm, we divide the results into several sections. First, we look at the performance of our initialization algorithms. The algorithm's ability to locate boundaries and regions with different intensity and geometric characteristics is then discussed. We will then look at the ability of our method to react to bleeding, a common problem when using front propagation methods. This section also covers the ability of our user coercion scheme to induce a desired behavior, which can include the prevention of bleeding. Finally, because medical images can sometimes be subject to significant amounts of noise, we examine our algorithm's performance in the presence of significant image noise.

For much of our results, we perform our analysis using synthetic images unrelated to medical imaging. This allows us to focus on specific image and algorithm characteristics instead of evaluating the algorithm based only on what a few medical images may have to offer. We do use a few MR data slices to obtain a more realistic look at our algorithm's functionality in the bleeding and noise sections.

Initialization

Since initialization is essential to the start of our algorithm, it is natural to begin our analysis here. We will focus on three images that introduce different kinds of boundaries into our data; we will call these Lena, Fingerprint, and Rings. Each of the images is shown with initialization performed using a selection of method and parameter combinations. While each starting point is only a single pixel, they will be indicated on the images by a '+' symbol for greater visibility. The actual starting point is the center of the symbol.

In evaluating our initialization, we will revisit our goals stated earlier. We should generate sufficient points to capture detail and which avoid boundaries in the data.

(a) Lena Test Image (b) Fingerprint Test Image (c) Rings Test Image

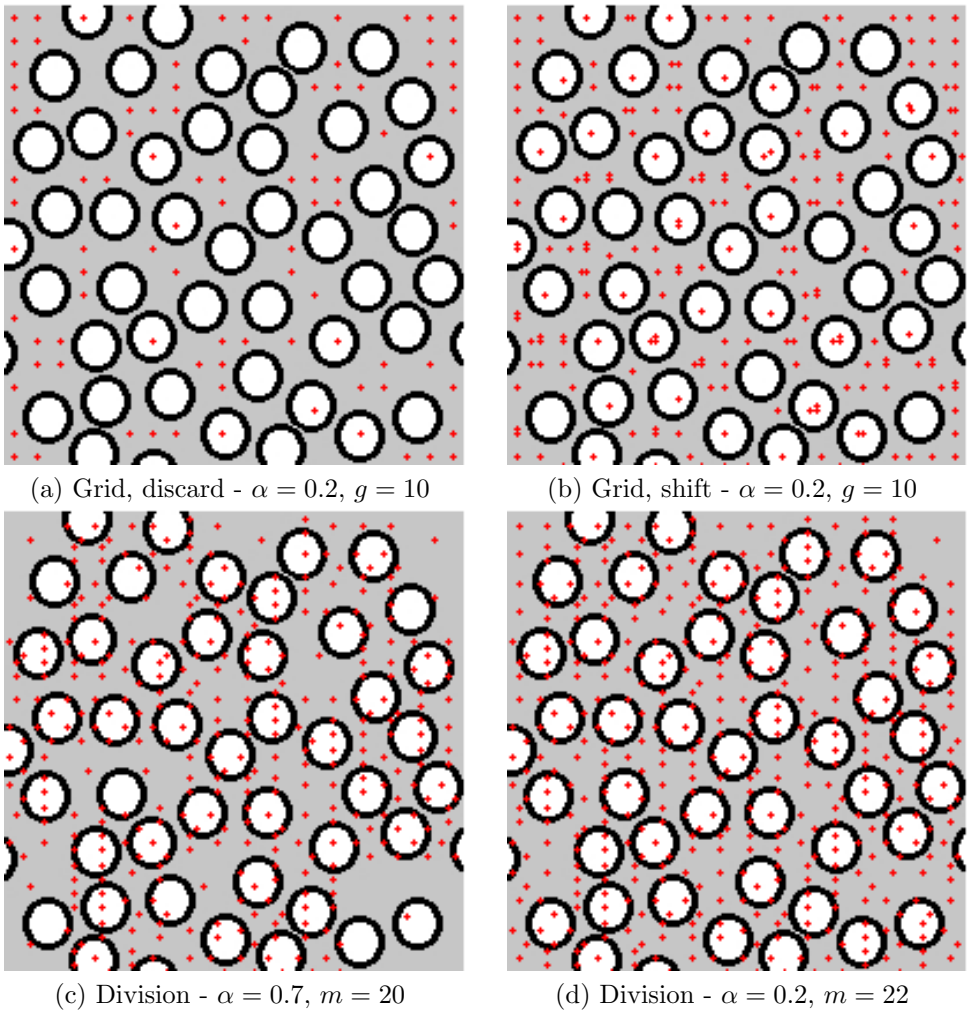
Figure 6: Initialization Test Images

Also, we must minimize computation by not producing substantially more points than are necessary. Between our two methods we have a few parameters that can be modified. For both the grid and division initialization, we can adjust α from Equation 7. For the grid method only, we can adjust and the width of our grid squares in pixels, which we will call g, and we can also choose to shift or discard points which do not meet the requirement of Equation 7. For the division method only, we can adjust the size in pixels of the minimum chunk of the image that is allowed to be split, which we will call m.

We will start with Rings. This appears to be the simplest of our images, but there is a difficult challenge in that the thin outer edge of the rings might also be considered separate structures, so we need starting points in them as well as the white circles and background. Various initialization results are shown in Figure 7. One iteration of the grid method is shown in 7a, and it is very clear that when discarding points that do not satisfy Equation 7, we will not have suitable results. Several of the circles have no starting points inside, and none of the rings do. We could attempt to get around this by making the grid size smaller than 10 pixels, but this would result in a huge amount of unnecessary points on the background, so we will avoid it. We use the same parameters with shifting in 7b and the results are much better. Most of the circles have starting points, but we are still left with a few empty ones and none of

the rings are addressed.

The division method is used in 7c, and we see much more encouraging results. There are fewer unnecessary starting points in the background, and many of the edges are addressed. By tweaking the parameters in 7d, we get a starting state that appears to meet our requirements.



(a) Grid, discard - $\alpha = 0.2$, $g = 10$

(b) Grid, shift - $\alpha = 0.2$, $g = 10$

(c) Division - $\alpha = 0.7$, $m = 20$

(d) Division - $\alpha = 0.2$, $m = 22$

Figure 7: Grid and Division Initialization - Rings

Figure 8 shows the initialization methods used on the Lena image, which tests our methods with a high level of detail and relatively strong boundaries. Again, the challenge here is mostly in getting starting points in the small details. The grid method in 8a is inadequate; the eyes are neglected entirely, as are the small details in

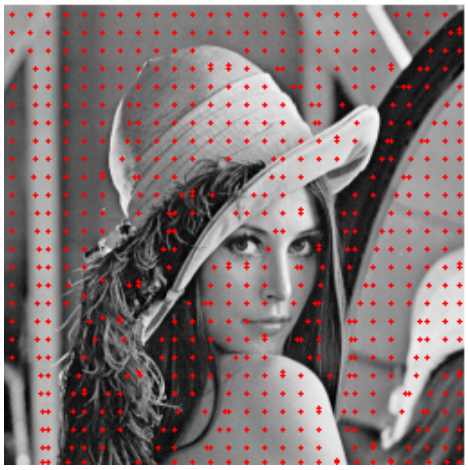
bottom left of the background. Attempting to remedy this by halving the grid size in 8b helps, but there are many unnecessary points in the background that would impede our efficiency. We see in 8c that using our original grid size and making the algorithm more tolerant of variance still neglects details.

The last three images in Figure 8 use the division method, and all appear to be more suitable than the grid images. For computation efficiency, 8f is the leader by far, but there are a few places where more points might be desired, such as the top of the hat, lips, and a small section of her shoulder which is cut off by hair. Both of the other two images address these places; either should be suitable for segmentation.

Fingerprint appears at first glance to be the most difficult image to initialize, and the one whose boundaries most closely mirror that of real MR data. The first three images in Figure 9a, all of those processed with the grid algorithm, appear woefully inadequate. Adjusting the parameters and using shifting obtains seemingly satisfactory results in some parts of 9c, but there are entire sections of the fingerprint left with no coverage.

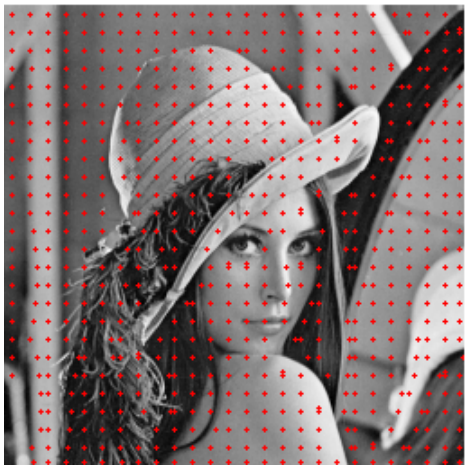
The division method used in 9d, however, is surprisingly effective. The image is well covered, and the points all appear to be in reasonable locations. Also, we can see the very significant effect of our algorithm's handling of the final split of each chunk, particularly just below and to the left of the center; the points are spread farther apart horizontal than vertically. This is because the algorithm took note of the high degree of variance in the vertical direction and split the chunks into top/bottom halves instead of left/right halves to result in better initialization of the region.

The common theme among our results is that the division algorithm outperforms our grid method significantly, regardless of the level of detail in the image or the level of contrast around the boundaries. We can attribute this to the overall approach of the algorithms to solving problem areas. While the grid method simply discards or moves a point when its target location is unsuitable, the division method attempts



(a) Grid, shift - $\alpha = 0.2$, $g = 10$

(b) Grid, shift - $\alpha = 0.2$, $g = 5$



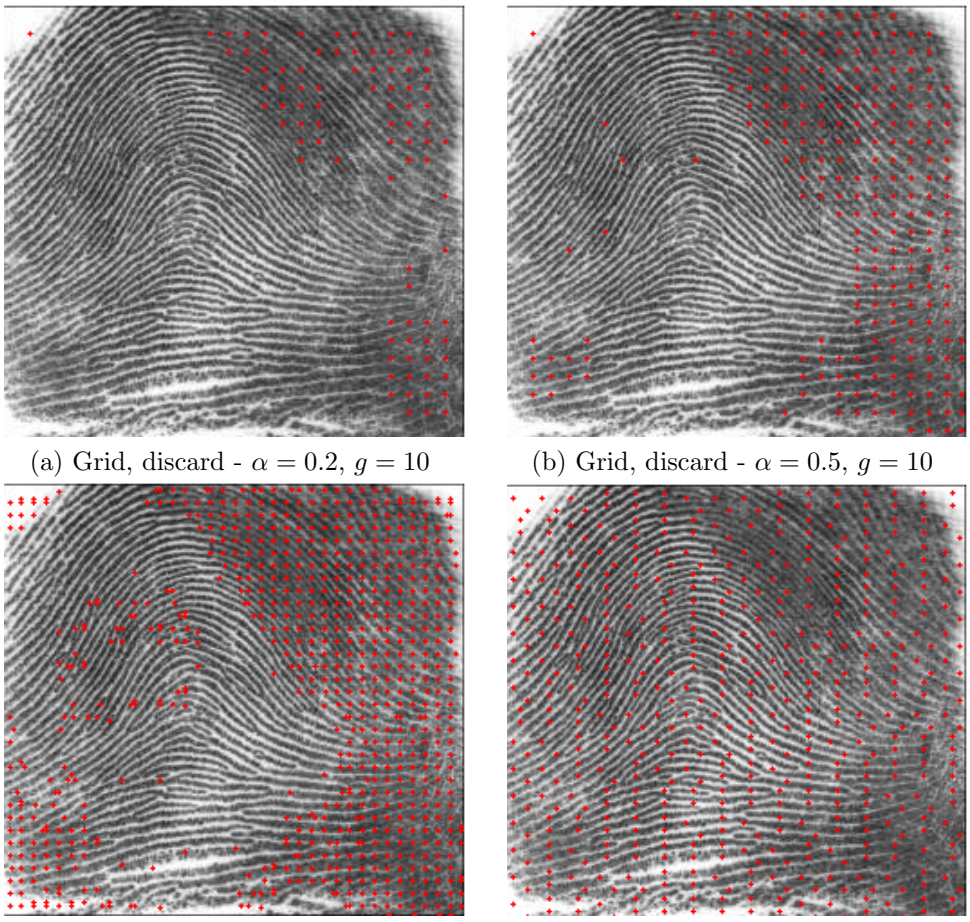
(c) Grid, shift - $\alpha = 0.4$, $g = 10$

(d) Division - $\alpha = 0.2$, $m = 10$

(e) Division - $\alpha = 0.1$, $m = 10$

(f) Division - $\alpha = 0.5$, $m = 10$

Figure 8: Grid and Division Initialization - Lena



(a) Grid, discard - $\alpha = 0.2$, $g = 10$

(b) Grid, discard - $\alpha = 0.5$, $g = 10$

(c) Grid, shift - $\alpha = 0.5$, $g = 7$

(d) Division - $\alpha = 0.1$, $m = 20$

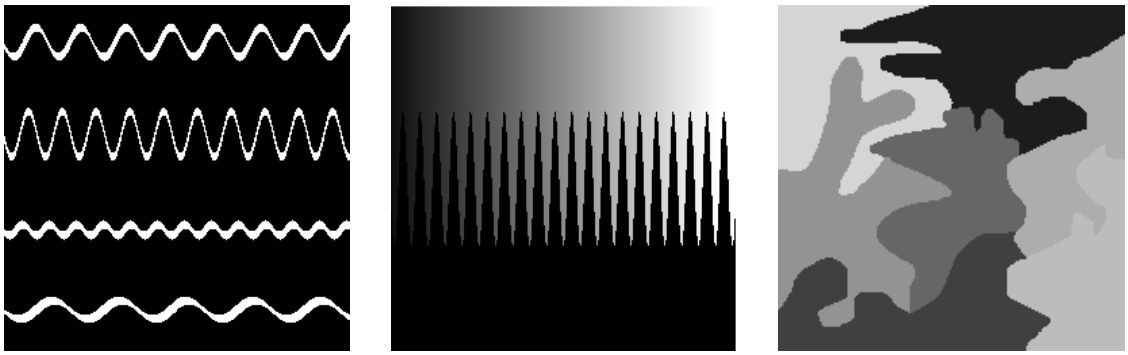
Figure 9: Grid and Division Initialization - Fingerprint

to go to a higher level of detail and add more points. This is interesting also because it reflects opposite applications of α in the same equation; when α is lowered in the grid method, fewer points are created because the criterion is tighter, while lowering α in the division method will likely result in more point locations as the algorithm tries to satisfy the stronger restriction.

Boundary and Gradient Performance

Next we test some images to see how our algorithm reacts to complex edges and slowly changing gradients within a single structure. Figure 10 shows the images we will use. 10a is a combination of out-of-phase sine wave pairs of different frequencies. This is

essentially 4 images in one, and will give us a good sampling of how the algorithm can perform when a structure is particularly narrow and bending. Another severe case is shown in 10b using a very steep sine wave with a gradient fill in the top. The bottom half of the image is fully black, while the top is fully white on the right side and on the left side is 5% of the grayscale range away from being completely black. This allows us to see how the algorithm reacts to various intensity gradients across complex borders, and we will be challenging the algorithm to recognize the top and bottom as two distinct structures. Next, since the novelty of our algorithm is its ability to locate multiple regions at once, we have an image with several oddly shaped regions in varying shades of gray in 10c.



(a) Sines Test Image (b) Sine Gradient Test Image (c) Sections Test Image

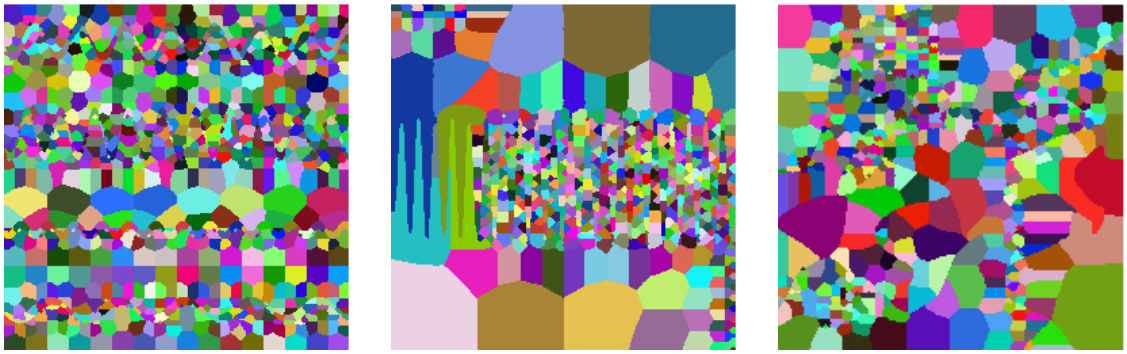
Figure 10: Boundary Complexity Test Images

Because our algorithm is performed in a number of stages, we will continue analyzing the results stage by stage. First, we will look at the growth of individual regions. Once we have seen how our initial starting points have expanded out to cover the image, we will then look at the different methods by which those regions can be joined together to create the structures we are seeking.

Note that for the remainder of this section, all three images have been initialized with the same method unless stated otherwise. Division initialization was used, with a minimum split size of $m = 10$ and with $\alpha = 0.1$ in Equation 7. The force function variable value from Equation 6 was $\tau = 10$ unless stated otherwise.

Region Growth

Figure 11 shows the test images from Figure 10 after a region expansion has taken place. Each colored section is a different region that has expanded until its borders are completely limited by its surrounding regions. Two regions with the same color have no special significance; the colors have been selected randomly.



(a) Sines Test Image (b) Sine Gradient Test Image (c) Sections Test Image

Figure 11: Test Images After Region Expansion

With the regions fully expanded, it is difficult to see where the boundaries are. Nonetheless, we can obtain some insight into the performance of the algorithm by analyzing the way the regions have grown. First, we immediately notice that areas with a higher concentration of edges have more and smaller regions than smooth, uneventful areas. This is evident in Figure 11a, where there are clearly four horizontal strips of highly concentrated, small regions where the sine waves were, with a similar effect on the borders of Figure 11c. This result is mostly due to the initialization planting more starting points in these areas, but it also allows us to see how the competitive nature of the algorithm significantly alters the way the regions grow. For example, look at the aqua and blue sections on the left side of 11b, where the top and bottom halves meet. There is a very clear sinusoidal boundary here. Had we done this segmentation seeking only a single region, with a starting point in the bottom left, for example, it is very likely that a portion of the top left would have been wrongly swallowed by the growing front before it reached the bottom right, where it

apparently belongs. This is shown in Figure 15 in the next section. Because we have searched for many regions at once, each region travels a smaller distance, reducing the chance of any region going too far. This benefit is more necessary in areas of greater complexity, which is why increasing the number of regions in high contrast areas is effective. Additionally, such an algorithm would have required us to devise a stopping criterion to determine when the front had gone far enough, which here we have avoided altogether.

Now that the regions are fully expanded, we need to address bringing regions together into larger structures, so they more closely resemble the segmentation results we are seeking.

Region Joining

Looking first at our simplest image, we immediately uncover a problem. In Figure 12, we see that after performing region joining there are some artifacts on the boundaries between the regions. From left to right, the value of λ_m is decreasing, and this causes the number and size of artifacts to decrease slightly. However, we also lose distinction between our regions, namely between the two rightmost regions; revisit the original image in Figure 10c and you will see that the boundary between these two structures was the least defined boundary in the image.

It turns out that this is caused by starting points that are very close to (within a pixel of) a boundary. When each starting point is initialized, each of the neighboring points will have its arrival time calculated based its neighbors' arrival times and the gradient at the starting point. Since all surrounding points other than the starting points are infinite (essentially unassigned as far as the calculation is concerned), the arrival times of all four neighboring points are based solely on the gradient at the starting point. If one of these neighboring points is on the other side of an intensity boundary, then the region will start expanding in that direction as quickly as it does

in other directions, resulting in a sizeable region that crosses even the most obvious of intensity boundaries. Since it spans two notably different intensities, Equation 9 is not satisfied enough for it to be absorbed into either structure. Possible solutions to this problem are discussed later.

(a) $\lambda_m = 0.01$ (b) $\lambda_m = 0.05$ (c) $\lambda_m = 0.1$

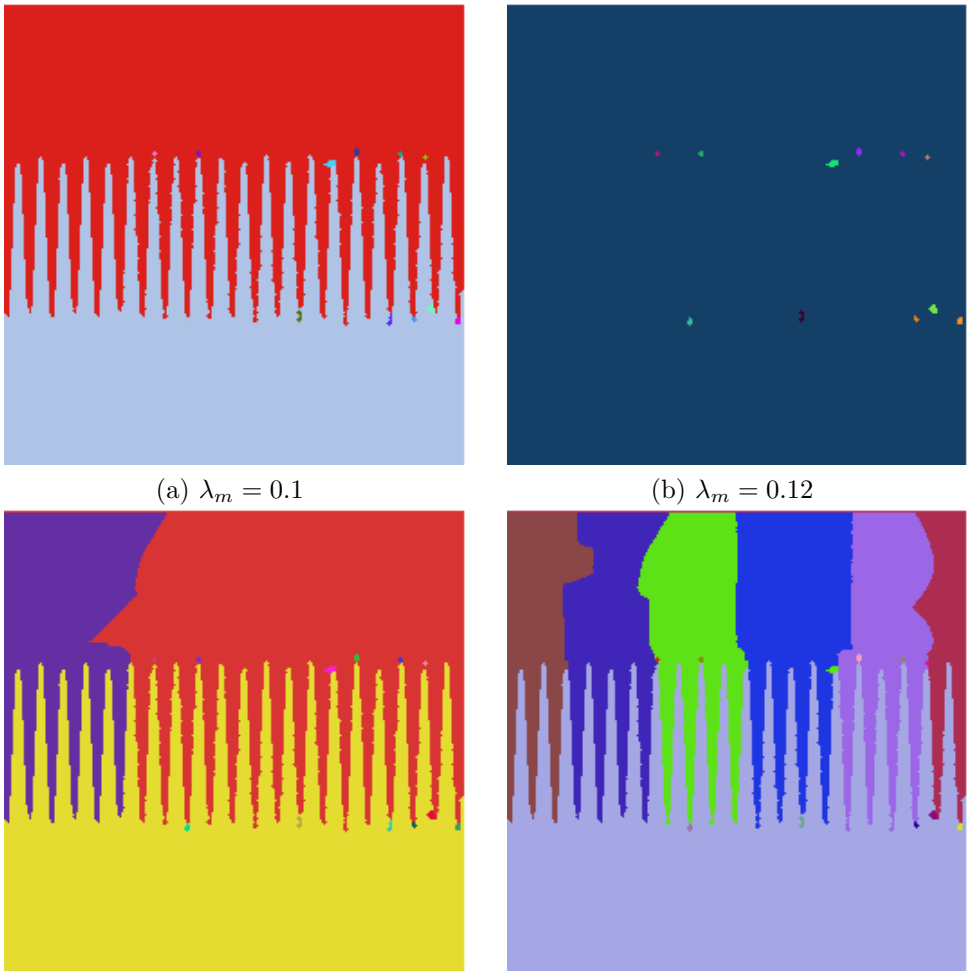
Figure 12: Sections test image after region joining. For each image, the value of λ_m used in Equation 9 is given.

Setting aside the artifacts, the algorithm has segmented the image as we expect in figures 12a and 12b. The majority of each section of the image has been grouped as a single region by the algorithm, even where the two regions were nearly the same shade on the right side. Moving on to a more difficult image, we see the results are similarly promising.

For our sine wave gradient image, 13a largely meets our expectations. As in Figure 12, there are some very small artifacts, but the entire gradient range across the top half of the image has been found to be a single entity, and the detail of every point of the sine wave is quite clear. The other images show us the effects on the results as the parameters of our algorithm change. When λ_m is increased, loosening the requirements for two regions to be combined, the entire image becomes a single entity. In Figure 13c we make our join requirement more stringent and get the opposite effect; the darkest part of the top half is determined to be a separate structure.

This effect is taken to an extreme when we enable region connectivity in 13d.

As regions are joined together, their means are combined before being compared to other neighbors for possible joining. Because of the gradient across the structure, the algorithm attempts to smooth it out, statistically ignoring the varying left to right intensity. The result is that we get a number of striped regions instead of the single continous region we were seeking. These results underscore the somewhat destructive effect of connectivity on our algorithm when dealing with structures that do not have highly uniform intensities.



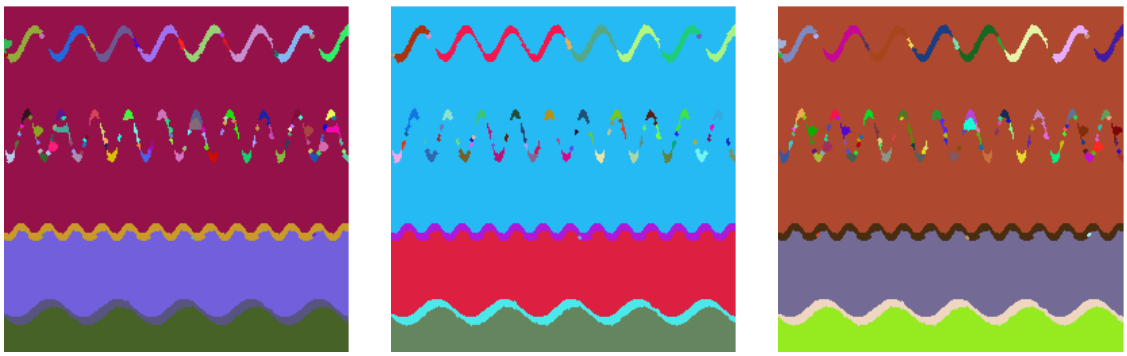
(a) $\lambda_m = 0.1$ (b) $\lambda_m = 0.12$

(c) $\lambda_m = 0.08$ (d) $\lambda_m = 0.1$ with join connectivity

Figure 13: Sine Gradient test image after region joining. For each image, the value of λ_m used in Equation 9 is given.

Finally, we have the results for our Sines image in Figure 14, where we clearly see that the width of a path has a substantial effect on the output. The wider, lower

sine waves have been filled in and segmented properly in all three images, but the narrower top two prove challenging. In 14a, the top wave has been filled out well, but the thinner sections at the center of each period have prevented the algorithm from joining the fragments into one whole structure. Also, the color of the background above and below the curve is the same, indicating that at least one of the regions that should have belonged to the sine wave has been incorrectly designated as a part of the background. The results for the narrower curve have the same problems but with greater severity. Raising λ_m in 14b to relax the joining criterion helps slightly in the top curve, where 3 periods have become connected, but otherwise worsens the problem by bolstering joins with the background.



(a) $\lambda_m = 0.1, \tau = 10$ (b) $\lambda_m = 0.3, \tau = 10$ (c) $\lambda_m = 0.1, \tau = 12$

Figure 14: Sines test image after region joining. For each image, the value of λ_m used in Equation 9 is given. The values of τ in Equation 6 are also given.

These problems are evidently caused by the region growth expanding beyond the strictly black or white area they start in by a few pixels. When the means are then calculated for each region, they are sufficiently dilluted to prevent the joining from working as desired. In an attempt to mediate this, we try raising the value of τ in Figure 14c, which should cause the growth of the regions to avoid the black/white boundary more vigorously. However, this seems to only result in the regions of very thin sections hardly growing at all, which of course does not produce the desired result. It is worth noting that in this case, the very narrow paths are particularly problematic because the path is diagonal; since the region expansion occurs in a four-

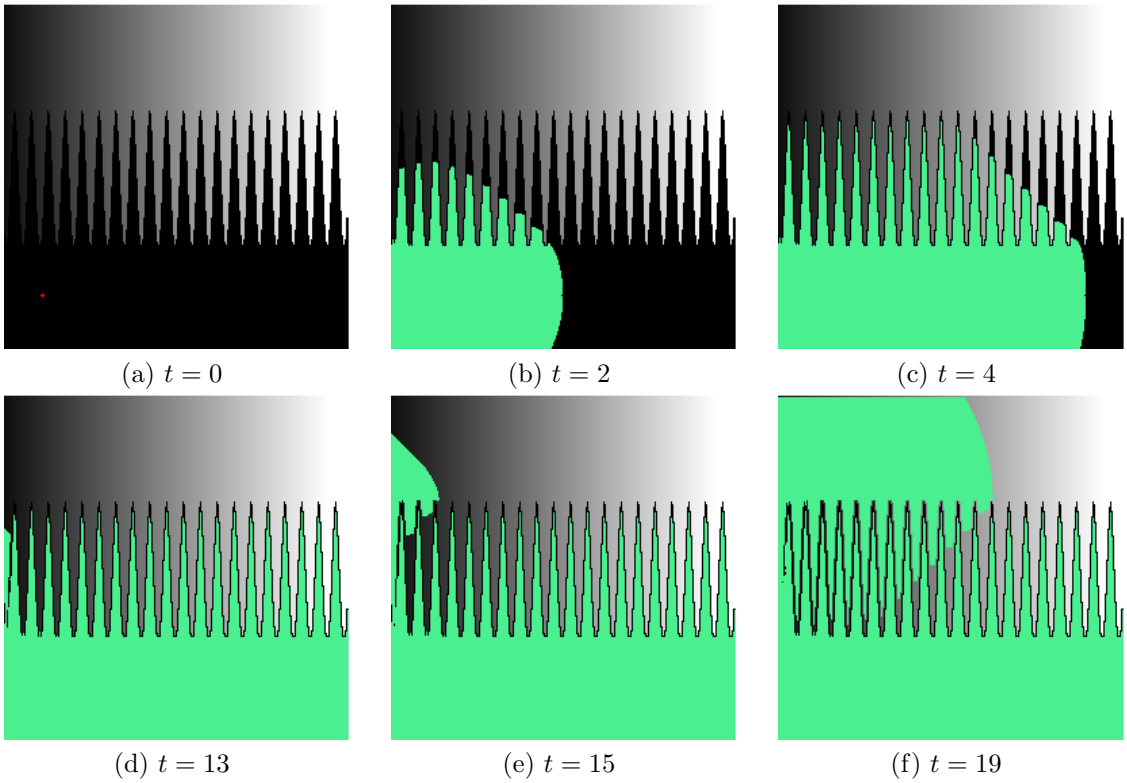
connected way, this is likely to be a much smaller issue when the paths are mostly horizontal or vertical.

Bleeding and Coercion

We specifically address the ability of our algorithm to stop bleeding because it is a fundamental problem with using the classical fast marching method by itself. Without an effective stopping criterion, which must often be tailored for a specific piece of data, fast marching fronts will always expand beyond their intended boundaries if left to run. We have already seen a rudimentary example of our algorithm combating this in Figure 11b.

Figure 15 shows an example of how the Sine Gradient image might be segmented according to the classical fast marching method. The results are very good at first, but it is clear that without a stopping criterion to stop the region's expansion between figures 15c and 15d, the results become unusuable. Of course, such a criterion would be simple to design, but would be usable only on images that are similar to this one. Figure 13 shows our algorithm's ability to suppress this overgrowth without image-specific formulas.

For a more realistic look at the performance of our algorithm with respect to bleeding, we will use some real medical data, as it is difficult to contrive robust test images to test bleeding. Figure 16 shows the progressions of regions over time using the conventional fast marching method in an MRI image of the third ventricle of the brain. The results are reasonable until time step 68 in 16f, when the lower right region begins to bleed beyond its boundary. The problem is made more severe in 16h when bleeding out the top becomes noticeable. This could be remedied by a robust stopping criterion that could halt front propagation between 16e and 16f, but then we would have to deal with an additional issue. Since the algorithm has not been allowed to expand to $t = \infty$, there are significant gaps in the structure that have not



(a) $t = 0$ (b) $t = 2$ (c) $t = 4$

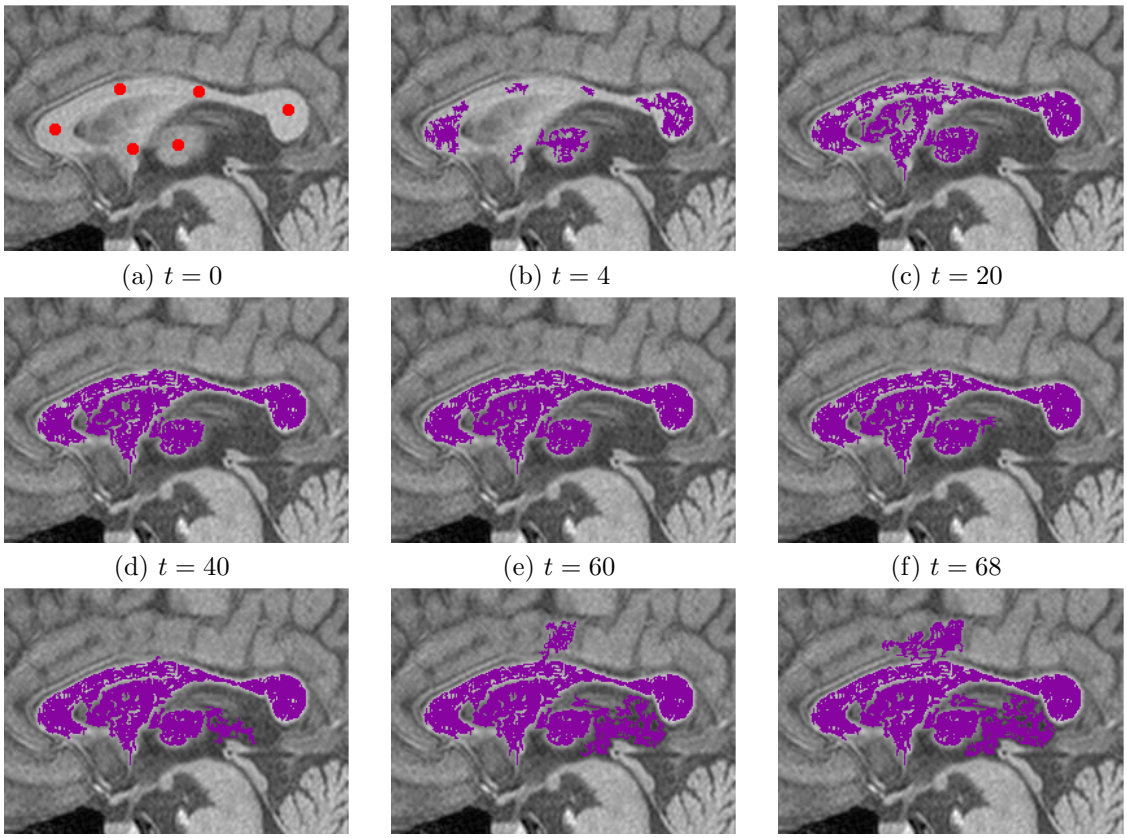
(d) $t = 13$ (e) $t = 15$ (f) $t = 19$

Figure 15: Bleeding in the Gradient Sine image. These images show the growth of a single region and the associated time step for the level of growth. The first image 15a shows just the starting point at $t = 0$

been filled when the borders are reached; these are noticeable even after significant bleeding has occurred in 16i. An additional pass would have to be used to fill in these gaps to make the resulting boundary easily recognizable for further processing, and just doing this could require another complex algorithm.

Note further that having all regions be displayed in the same color in Figure 16 has nothing to do with our algorithm. As is usually the case with the conventional fast marching method, it was the user's responsibility to ensure all starting points represented parts of the same structure.

Performing the same segmentation with our method, we see very different results with regard to bleeding. In general, the propagation appears much faster because there is a greater number of starting points, and by $t = 4$ the areas surrounding the third ventricle are drawing near to its edges. At $t = 68$, when the results from



(a) $t = 0$ (b) $t = 4$ (c) $t = 20$

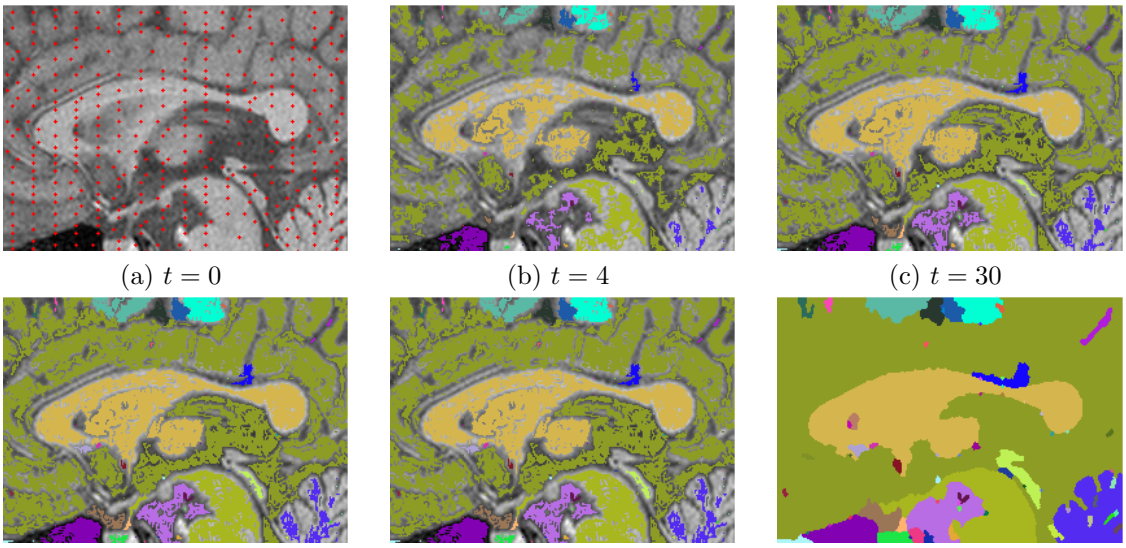
(d) $t = 40$ (e) $t = 60$ (f) $t = 68$

(g) $t = 85$ (h) $t = 100$ (i) $t = 110$

Figure 16: Bleeding in the third ventricle of the brain using the standard fast marching method. These images show the growth of user selected starting points and the associated time step for the level of growth. The first image 15a shows just the starting points at $t = 0$. Starting points have been enhanced to increase visibility. For all of these images, $\tau = 10$ in Equation 6.

the conventional fast marching method began to break down, any risk of bleeding is virtually gone because of the competition of surrounding structures. Competition also allows us to go on to $t = \infty$, shown in 17f, where we achieve quite accurate results and have resolved the problem of unfilled gaps within the regions.

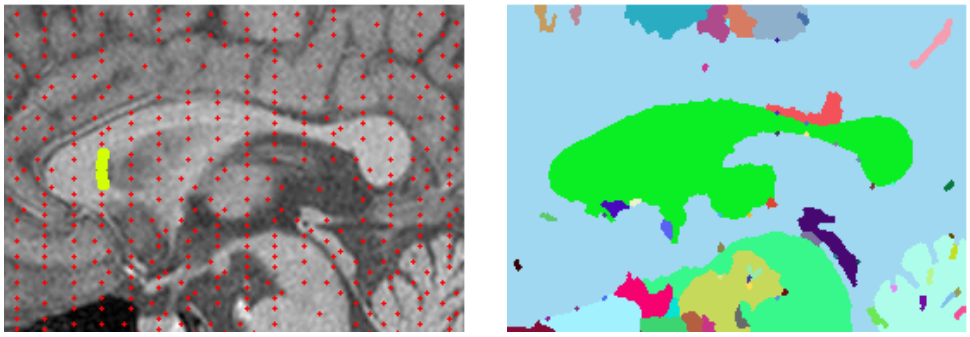
We do see a small problem with our results, however. Toward the left side of third ventricle (the front, biologically speaking), we see a small spot that has not been interpreted as part of the larger structure. This is due to the particularly dark spot of the third ventricle that we can see in 17a. Our coercion mechanism can force this to be interpreted correctly, as shown in Figure 18.



(a) $t = 0$ (b) $t = 4$ (c) $t = 30$

(d) $t = 68$ (e) $t = 76$ (f) $t = \infty$

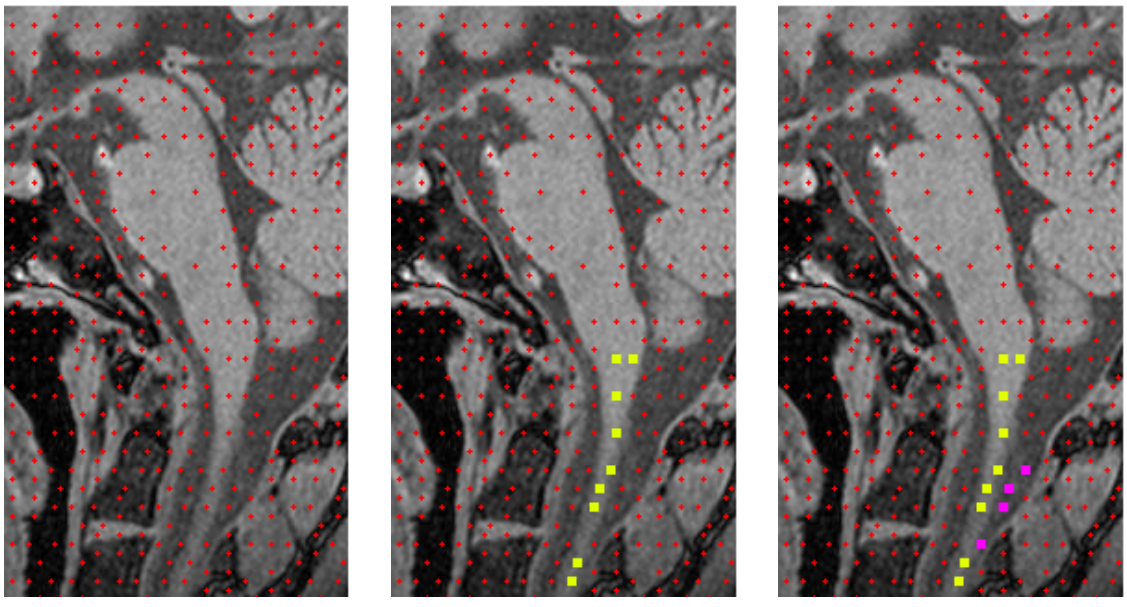
Figure 17: Growth of the third ventricle using our method. Starting points have not been enhanced, as there are too many to do so without impairing visibility. In these figures, $\alpha = 0.2$, $\lambda_m = 0.07$, and $\tau = 10$ in Equations 7, 9 and 6. The minimum split size $m = 20$ for initialization.



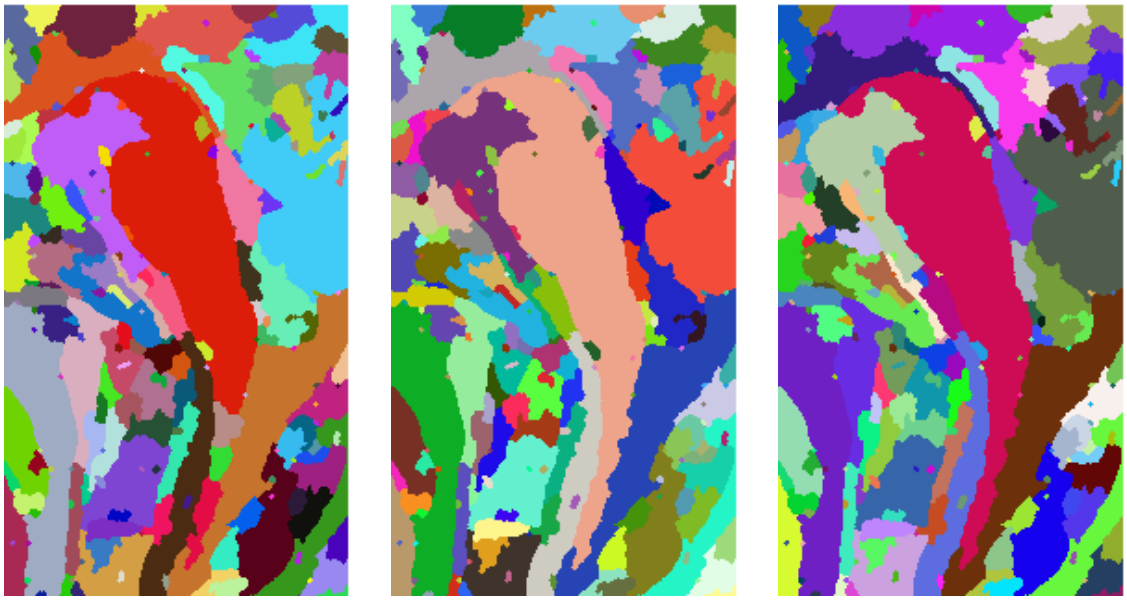
(a) $t = 0$ (b) $t = \infty$

Figure 18: Growth of the third ventricle with coercion in use. 18a shows the image at $t = 0$ with the four user-selected points enhanced for greater visibility. The algorithm will interpret these four points as being from the same region, forcing them to join and eliminating in Figure 18b the artifact that was visible in Figure 17f. Variable values mentioned in Figure 17 are the same here. In these figures, $\rho = 0.2$ and $\nu = 3.0$ in Equations 10, 12, and 13.

This shows the functionality of our coercion mechanism for forcing two adjacent regions to join, but does not demonstrate our ability to influence region growth. For a better analysis of this, we look at another section of the same source image, the spinal cord.



(a) Original no coercion (b) Original, 1 coercion (c) Original, 2 coercions



(d) Segmented no coercion (e) Segmented, 1 coercion (f) Segmented, 2 coercions

Figure 19: Segmentation of the spinal cord with coercion. In 19b and 19c the starting points for coersed groups have been enhanced for better visibility. To produce these results, $\tau = 15$, $\lambda_m = 0.03$, $\alpha = 0.15$, $\rho = 0.2$, and $\nu = 3.0$ in Equations 6, 9, 7, 10, 12, and 13. The minimum split size for division initialization is 15.

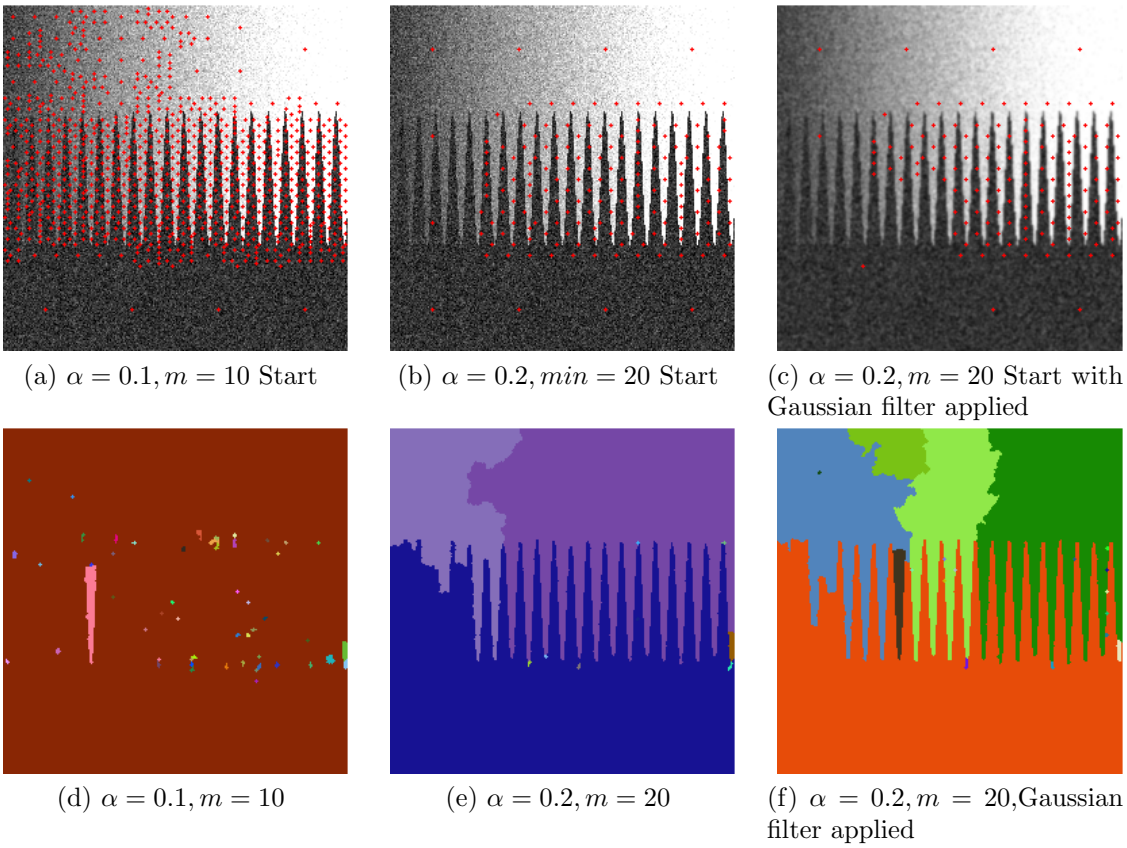
Figure 19 shows attempts to influence the spinal cord segmentation with coercion. We see the problem on the lower end of the spinal cord in 19d; this narrow portion has been split into several pieces, and the last part of it has been joined with another

structure to the right (to the rear, biologically) of the spinal cord. By identifying a group of points that belong together in 19b, we successfully designate all but one of the small sections as part of the spinal cord in 19d. However, the very last section is still being mistaken for part of another structure. In a second attempt, an additional group is added in 19c aimed at causing a faster expansion of the lowest point in the second group. This unfortunately does not succeed; the results shown in 19f are no better than the previous ones.

Noise Performance

To evaluate our algorithm's performance with high amounts of noise, we start with the Sine Gradient image. This is a good image to use, since it has relatively high amounts of detail and intensity values at both extremes of grayscale. Figures 20, 21, and 22 show this image with Gaussian noise applied and the results of the algorithm under various conditions. All algorithm parameters not explicitly mentioned are the same as those used to produce Figure 13a. The variance of the noise applied in all figures is 0.01.

Figure 20d shows the image segmented with the parameters that were most successful when no noise was present, and the results are unfortunately poor; the entire image has essentially become a single region with only artifacts remaining of the original image. If we look at 20a, we see that the number of starting points has been driven very high by the extra image variation created by the noise, which may be producing our problem. We attempt to remedy this in 20c by manipulating the parameters to reduce the number of starting points, and this pays off in 20f. Most of the boundary is successfully located, but the more obscure portion is lost, and the portion of the gradient that has an intensity closer to the mean of the noise is viewed as a separate region. We attempt to mediate this further in the last two images by applying a 5x5 Gaussian smoothing filter with a standard deviation of 0.8, but this



(a) $\alpha = 0.1, m = 10$ Start

(b) $\alpha = 0.2, min = 20$ Start

(c) $\alpha = 0.2, m = 20$ Start with Gaussian filter applied

(d) $\alpha = 0.1, m = 10$

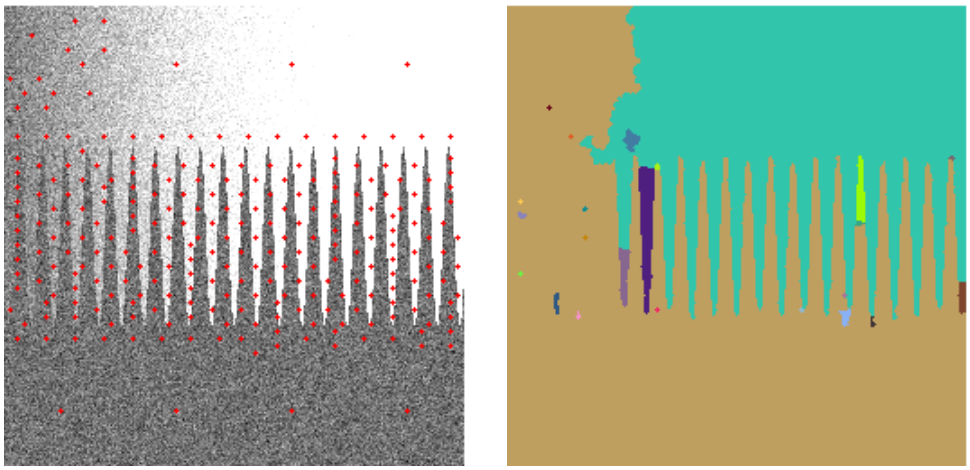
(e) $\alpha = 0.2, m = 20$

(f) $\alpha = 0.2, m = 20$,Gaussian filter applied

Figure 20: Sine Gradient with Gaussian noise applied with a mean of 0.2. The top row of images show the beginning state of the algorithm with starting points, and the bottom row shows the corresponding segmented image. For all images, α refers to Equation 7 and m is the minimum split value for division starting point initialization.

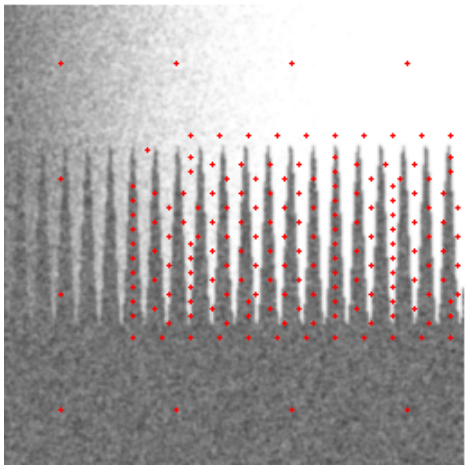
appears only to make matters worse as the gradient portion of the image is separated into more regions and the border appears slightly rougher.

The results produced in Figure 21 show significantly better results, presumably because the mean of the noise is at the center of the intensity spectrum rather than near one end. Results using the same parameters from 20e show similar performance in 21b, albeit with a few artifacts. We see in 21d that using the same smoothing filter we used earlier produces a much better effect here; most of the boundary is very clear. The less obvious portion of the boundary has been missed, and this is to be expected. As the noise is smoothed out by the Gaussian filter, the top and bottom of the left end would made to have more similar intensities where the intensities were



(a) No filter, Start

(b) No filter, Result



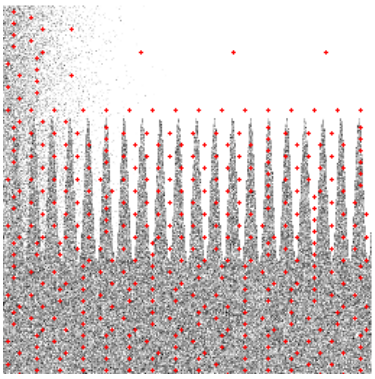
(c) Filter applied, Start

(d) Filter applied, Result

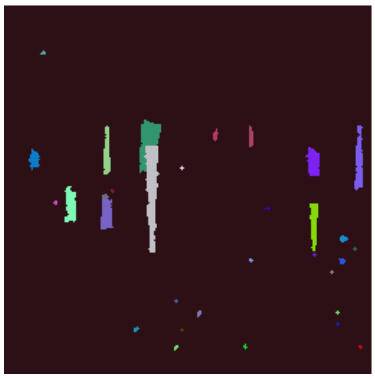
Figure 21: Sine Gradient with Gaussian noise applied with a mean of 0.5. The left images show the beginning state of the algorithm with starting points and the images are the corresponding segmented results. For all images, $\alpha = 0.2$ in Equation 7 and the minimum split value for division starting point initialization is $m = 20$.

very close to begin with.

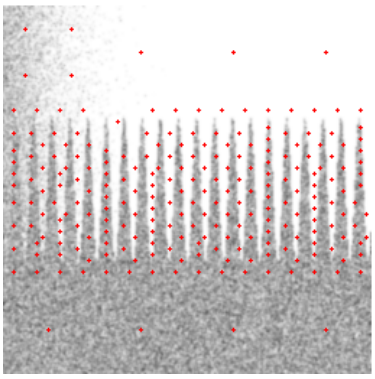
Finally, Figure 22 shows results on the opposite end of the intensity spectrum from Figure 20. Our initial set of results in 22b is very poor, with just some artifacts and small portions of the boundary made visible by stray regions. The Gaussian filter does a good job of smoothing out the result in 22d, but unfortunately leads to the entire image being recognized as a single region. By narrowing the region joining criterion in 22e, the results are much improved. Most of the boundary is visible and the top gradient has been segmented better than in our other results, though there



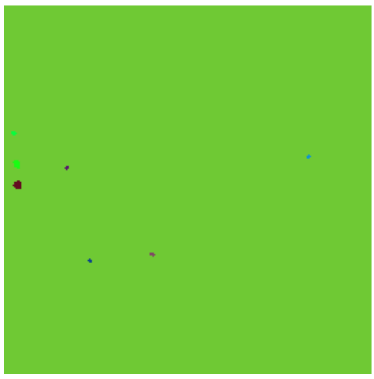
(a) No filter, Start



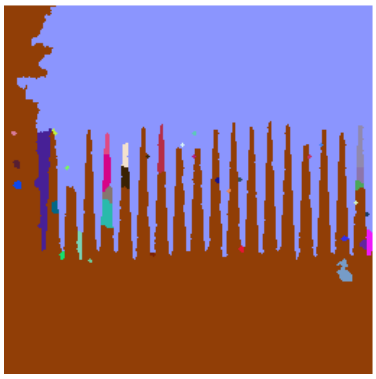
(b) No filter, Result



(c) Filter applied, Start



(d) Filter applied, Result

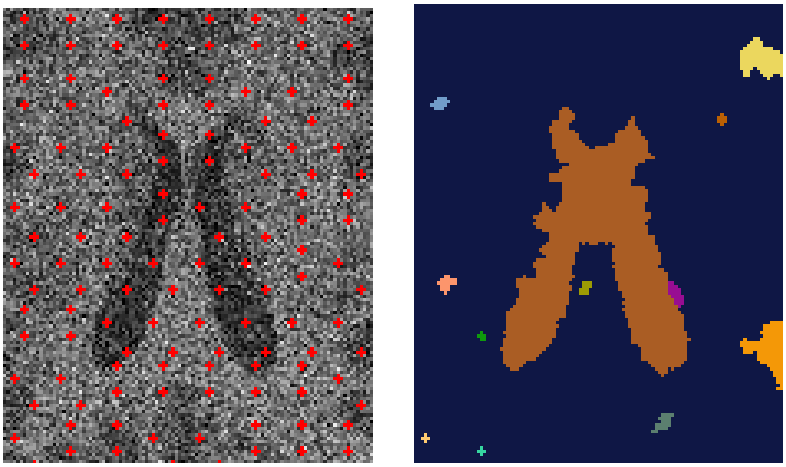


(e) Filter applied, Result with $\lambda_m = 0.05$

Figure 22: Sine Gradient with Gaussian noise applied with a mean of 0.8. The left images show the beginning state of the algorithm with starting points and the images are the corresponding segmented results. For all images, $\alpha = 0.2$ in Equation 7 and the minimum split value for division starting point initialization is $m = 20$. In 22e, λ_m refers to Equation 9

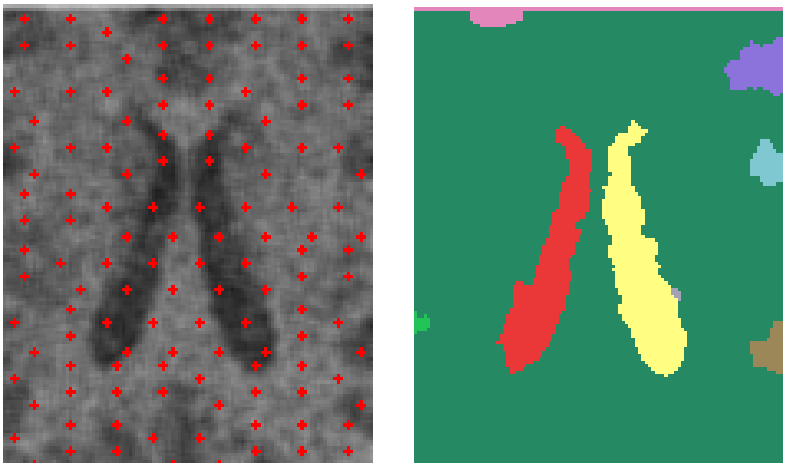
are significantly more artifacts than we saw in 20e and 21d. The gradient section has been recognized as a single region very successfully, presumably because the filter has lightened the entire top half, causing it to appear more uniform to the algorithm.

To get a more realistic look at performance under noise, we briefly examine an MR data slice with a high amount of instrument-produced noise.



(a) Original image with starting points

(b) Segmented image



(c) Original image with filter applied and starting points

(d) Filtered image, segmented

Figure 23: Portion of a top view MR slice of the head. The filter applied in 23c is a 4x4 Gaussian filter with standard deviation $\sigma^2 = 5$. The parameters used to produce the results are $\tau = 20$, $\lambda_m = 0.07$, and $\alpha = 0.15$ from Equations 6, 9, and 7. The division initialization minimum split size is $m = 20$.

The image and segmented results are showing in Figure 23. The image comes from a top view of the head, cropped to narrow our view to the target structures, the two small elongated sections in the center. Using the original image, the algorithm is able to segment out the two structures, but there are several artifacts, as well as very rough edges, and the two structures have been incorrectly joined by the algorithm. A

strong Gaussian filter is applied in an attempt to even it out. Segmenting the filtered image produces Figure 23d, which clearly shows the two structures we were seeking, as well as some of the outlying darker structures on the edge of the image. The edges of the elongated structures appear quite rough at first, but with a closer look at 23c, we see that many of the rough characteristics are actually present in the smoothed input image.

While our performance on noisy data is imperfect, we have shown that noise does not impair the algorithm's ability to function effectively. Even in the very noisy scenarios we evaluated for the Sine Gradient image, much of the boundary was still located after parameters were adjusted slightly. Our results with real MR data with the application of an appropriate smoothing filter.

5 Conclusions

We have discussed a competitive fast marching method for medical image segmentation. In the course of determining the success of the various parts of our method, we have shown that some of the developed techniques were not particularly effective, namely the coercion mechanism and the grid-based starting point initialization. However, these did not prevent the algorithm as a whole from functioning successfully. Most of the components of the algorithm have functioned successfully in operating on various images, and in some cases did so with significant amounts of noise. Our test images had diverse characteristics which exposed various fine points of behavior exhibited by the algorithm.

We showed at least two examples where our performance in segmenting medical images was very good, particularly in warding off bleeding problems that can be quite common with the gradual intensity changes often found in biological data. A third medical image produced reasonable, if imperfect, results which may because usable with future enhancements to our algorithm. This verifies that our method has merit as a realistic segmentation technique, particularly when used with medical data.

6 Future Work

Some promising results with our competitive algorithm open the door to significant exploration. For one, it would be valuable to see how extension into the third dimension would affect performance. Since regions in 3 dimensions would have a much more diverse statistical makeup than those in just 2 dimensions, this has potential to significantly change the performance of our algorithm for better or for worse.

Since the algorithm is controlled by a relatively small set of parameters, higher quality results might also be produced by introducing some a priori knowledge into the algorithm. For example, when seeking the boundaries of a particular organ in the body, it may be possible to show through repeated experience that a given set of parameters are particularly good at locating that specific structure. This may undermine the generality of the algorithm to some degree, but if performance can be significantly improved or the results can be made more reproducable, it might have important implications for eventual use in patient diagnosis and treatment.

One of the strengths of our use of the Fast Marching Method is that the core concepts of the our algorithm are independent of the speed and direction with which the fronts propagate. There are numerous papers discussing manipulations of the level-set force function (curvature is a particularly popular factor, as in [15]), and integrating some such changes could significantly improve the results produced. These changes could be particularly effective if our algorithm were integrated with the full level set method, since the requirement that the force function be monotonically increasing would be lifted.

Allowing fronts that can move backward could also add a new dynamic to the competitive aspect of our method. Competition could be made much more explicit, with regions statistically competing for pixels that lay on their boundaries, rather than simply racing the other regions to those pixels. In addition to producing more robust results in the general case, this would be one possible way of correcting the

artifact problem that we saw in Figure 12. Regions that initially expand over very obvious boundaries could be pushed back, preventing their statistical means from becoming too dilluted to join either major region. The artifact problem might also be solved by a more intelligent force function, or an initialization method that more robustly avoids boundaries.

There are several images throughout our results that suggest improved region joining criteria could be helpful. While Equation 9 has proven successful in many of our results, a more complex criterion involving statistics other than the mean might produce better results in some circumstances. This goes hand in hand with improvements to the coercion mechanism; our results showed that this feature was largely ineffective, and enhancements could allow the user better ability to influence results. One possible improvement might be to modify the propagation equations to be more effective in all directions, as their functionality is currently someone limited when not affecting propagation that is directly up, down, left, or right. It could also be very helpful to allow the user to add their own starting points for coercion, rather than having to select from those which were produced automatically.

7 Appendix - Implementation Details

All algorithms were implemented in Matlab 7.3.0.267 (R2006b) with the exception of optimizations to the Fast Marching Method. An unmanaged C++ library, compiled in MS Visual Studio .NET 2003, was used to store the narrow band values in a heap so that the pixel with the minimum t-value could be easily retrieved. This library was also used to store lists of pixels belonging to each region, so that they could be easily marked with updated region markers when region joining was performed. The functions were exported from the library as standard C functions, which Matlab then linked to. The Matlab compiler was not used.

The platform used was Microsoft Windows XP with Service Pack 2. The benchmarks that follow were run on a Toshiba Satellite laptop with a 1.6Ghz Intel Centrino processor, 1 GB of RAM, and an 80 GB‘parallel ATA hard drive. In general, parameters chosen and number of starting points have no impact on performance assuming that memory is not severely exhausted. Also, this time should be taken to reflect the completion of the fast marching method expansion. The point initialization and the region joining stages take a negligible amount of time, generally less than 1 second.

Figure #	Resolution	Time to run
23d	113x140	15 seconds
18b	267x190	45 seconds
19d	187x320	48 seconds
13a	256x256	1 minute, 8 seconds

Table 1: Algorithm Runtime Statistics

References

[1] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118:269–277, 1995.

[2] V. Caselles. Geometric models for active contours. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 9–12, 1995.

[3] Amit Chakraborty and James S. Duncan. Game-theoretic integration for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(1):12–30, 1999.

[4] Laurent D. Cohen. Multiple contour finding and perceptual grouping using minimal paths. *J. Math. Imaging Vis.*, 14(3):225–236, 2001.

[5] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models–their training and application. *Comput. Vis. Image Underst.*, 61(1):38–59, 1995.

[6] Timothy F. Cootes, A. Hill, Christopher J. Taylor, and J. Haslam. The use of active shape models for locating structures in medical images. In *IPMI*, pages 33–47, 1993.

[7] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994.

[8] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[9] Sarang C. Joshi, Stephen M. Pizer, P. Thomas Fletcher, Andrew Thall, and Gregg Tracton. Multi-scale 3-d deformable model segmentation based on medial description. In *IPMI '01: Proceedings of the 17th International Conference on Information Processing in Medical Imaging*, pages 64–77, London, UK, 2001. Springer-Verlag.

[10] Yootai Kim, Raghu Machiraju, and David Thompson. Rough interface reconstruction using the level set method. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 251–258, Washington, DC, USA, 2004. IEEE Computer Society.

[11] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature-sensitive surface extraction from volume data. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 57–66. ACM Press / ACM SIGGRAPH, 2001.

[12] Ioannis Kompatsiaris and Michael G. Strintzis. Spatiotemporal segmentation and tracking of objects for visualization of videoconference image sequences. In *ICMCS, Vol. 1*, pages 709–713, 1999.

[13] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM.

[14] R. Malladi, J. A. Sethian, and B. C. Vemuri. A fast level set based algorithm for topology-independent shape modeling. *J. Math. Imaging and Vision*, 6(2/3):269–290, 1996.

[15] Ravi Malladi and James A. Sethian. A real-time algorithm for medical shape recovery. In *ICCV*, pages 304–310, 1998.

[16] Ravi Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[17] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[18] V. Mezaris, I. Kompatsiaris, and M. Strintzis. A framework for the efficient segmentation of large-format color images, 2002.

[19] Reza Momenan, Daniel Hommer, Robert Rawlings, Urs Ruttimann, Michael Kerich, and Daniel Rio. Intensity-adaptive segmentation of single-echo t1-weighted magnetic resonance images. *Human Brain Mapping*, 5(3):194–205, 1997.

[20] Frederic Cao Pablo Muse and Frederic Sur. Extracting meaningful curves from images. *Journal of Mathematical Imaging and Vision*, 22:159–181, 2005.

[21] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[22] J. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci.*, pages 1591–1595, 1996.

[23] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry*. Cambridge University Press, 1998.

[24] J. A. Sethian. Curvature and the evolution of fronts. *Communications in Mathematical Physics*, 101:487–499, 1985.

[25] E. Sifakis, C. Garcia, and G. Tziritas. Bayesian level sets for image segmentation. *Visual Communication and Image Representation*, 2000.

[26] Stephen M. Smith. Fast robust automated brain extraction. *Human Brain Mapping*, 17(3):143–155, 2002.

[27] J. Suri, K. Liu, S. Singh, S. Laxminarayana, and L. Reden. Shape recovery algorithms using level sets in 2-d/3-d medical imagery: A state-of-the-art review, 2001.

[28] L. Vincent and P. Sollie. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[29] W. M. Wells III, W. E. L. Grimson, R. Kikinis, and F. A. Jolesz. Adaptive segmentation of MRI data. In Nicholas Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine*. Springer-Verlag, 1995.

[30] Paul A. Yushkevich, Joseph Piven, Heather Cody, Sean Ho, and Guido Gerig. Geodesic snakes for user-guided segmentation of 3-d anatomical objects: Significantly improved efficiency and reliability, 2005.

[31] Song Chun Zhu and Alan L. Yuille. Region competition: Unifying snakes, region growing, and bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996.